IMSL®

IMSL Fortran Library User's Guide
STAT/LIBRARY Volume 2 of 2

Statistical Functions in Fortran

Trusted For Over 30 Years

IMSL®

IMSL Fortran Library User's Guide
STAT/LIBRARY Volume 2 of 2

Statistical Functions in Fortran

Visual Numerics

P/N 7697

[www.vni.com]

Visual Numerics, Inc. – United States
Corporate Headquarters
2000 Crow Canyon Place, Suite 270
San Ramon, CA 94583
PHONE: 925-807-0138
FAX: 925-807-0145
e-mail: info@vni.com
Westminster, CO
PHONE: 303-379-3040

Houston, TX
PHONE: 713-784-3131

Visual Numerics International Ltd.
Sussex House
6 The Forbury
Reading, Berkshire RGI 3EJ
UNITED KINGDOM

PHONE: +44-1-189 25-3370
FAX: +44 –1-189-25-3371
e-mail: info@vniuk.co.uk
Support: support@vniuk.co.uk

Visual Numerics SARL
Immeuble le Wilson 1
70, avenue due General de Gaulle
F-92058 PARIS LA DEFENSE, Cedex
FRANCE

PHONE: +33-1-46-93-94-20
FAX: +33-1-46-93-94-39
e-mail: info@vni.paris.fr

Visual Numerics S. A. de C. V.
Florencia 57 Piso 10-01
Col. Juarez
Mexico D. F. C. P. 06600
MEXICO

PHONE: +52-55-514-9730 or 9628
FAX: +52-55-514-4873

Visual Numerics International GmbH
Zettachring 10
D-70567Stuttgart
GERMANY

PHONE: +49-711-13287-0
FAX: +49-711-13287-99
e-mail: vni@visual-numerics.de

Visual Numerics Japan, Inc.
GOBANCHO HIKARI BLDG. 4TH Floor
14 GOBAN-CHO CHIYODA-KU
TOKYO, JAPAN 102

PHONE: +81-3-5211-7760
FAX: +81-3-5211-7769
e-mail: vnijapan@vnij.co.jp

Visual Numerics, Inc.
7/F, #510, Sect. 5
Chung Hsiao E. Road
Taipei, Taiwan 110
ROC

PHONE: +(886) 2-2727-2255
FAX: +(886) 2-2727-6798
e-mail: info@vni.com.tw

Visual Numerics Korea, Inc.
HANSHIN BLDG. Room 801
136-1, MAPO-DONG, MAPO-GU
SEOUL, 121-050
KOREA SOUTH

PHONE: +82-2-3273-2632 or 2633
FAX: +82-2-3273--2634
e-mail: info@vni.co.kr

World Wide Web site: http://www.vni.com

IMSL Fortran, C, and Java
Application Development Tools

Contents

Chapter 8: Time Series Analysis and Forecasting

Routines

Usage Notes

The name of a time series routine is a combination of three to four sets of one to four letters. The first set specifies the type of model or method. The second set identifies the particular approach. If the name uses four sets of letters, then both the second and third sets are used to identify the particular approach. The final set always specifies the general procedure. The table below summarizes the naming convention of the time series analysis and forecasting routines.

The names and meanings of arguments are consistent within a set of routines pertaining to a particular topic. For example, `XCNTR` corresponds to the constant used to center the time series `X` in all of the spectral analysis routines. Note that `IPRINT` always represents the printing option, the values and possible choices of output necessarily depend on the given routine. An option argument always begins with the letter "`I`," and a leading dimension argument always begins with "`LD`."

The routines in this chapter assume the time series does not contain any missing observations. If missing values are present, they should be set to NaN (see the routine `AMACH`, in the Reference Material section), and the routine will return an appropriate error message. To enable fitting of the model, the missing values must be replaced by appropriate estimates.

Naming Conventions in Chapter 8

Meaning	Abbreviation
Bayesian	BAY*
Nonseasonal ARMA	NS*
Transfer Function	TF*
Maximum Likelihood	MAX*
Multichannel	M*
Periodogram	P*
Cross Periodogram	CP*
Spectral Density	S*
Automatic Model Selection	AUTO*
Cross-Spectral Density	CS*
Preliminary	*P*
Univariate	*UNI*
Multivariate	*MUL*
Final Prediction Error	*FPE*
Method of Moments	*MM*
Least-Squares	*LS*
Box-Jenkins	*BJ*
Spectral Window	*SW*
Weights	*WE*
Autoregressive	*AR
Autoregressive, Moving Average	*ARMA
Seasonal Modeling	*SEA
Estimation	*E
Forecast	*F
Fast Fourier Transform	*FFT
Periodogram	*P
Data	*D

The "*" represents one or more letters.

General Methodology

A major component of the model identification step concerns determining if a given time series is stationary. The sample correlation functions computed by routines ACF (page 697), PACF (page 702), CCF (page 706), and MCCF (page 711) may be used to diagnose the presence of nonstationarity in the data, as well as to indicate the type of transformation require to induce stationarity. The family of power transformations provided by routine BCTR (page 689) coupled with the ability to difference the transformed data using routine DIFF (page 692) affords a convenient method of transforming a wide class of nonstationary time series to stationarity.

The "raw" data, transformed data, and sample correlation functions also provide insight into the nature of the underlying model. Typically, this information is displayed in graphical form via time series plots, plots of the lagged data, and various correlation function plots. The routines in Chapter 16, "Line Printer Graphics," provide the necessary tools to produce the visual displays of this quantitative information.

The observed time series may also be compared with time series generated from various theoretical models to help identify possible candidates for model fitting. The routine RNARM in Chapter 18, "Random Number Generation" may be used to generate a time series according to a specified autoregressive moving average model.

Time Domain Methodology

Once the data are transformed to stationarity, a tentative model in the time domain is often proposed and parameter estimation, diagnostic checking and forecasting are performed.

Autoregressive Moving Average Model

A parsimonious, yet comprehensive, class of stationary time series models consists of the nonseasonal autoregressive moving average (ARMA) processes defined by

$$\phi(B)(W_t - \mu) = \theta(B)A_t \qquad t \in ZZ$$

where

$$\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$$

denotes the set of integers, B is the backward shift operator defined by $B^k W_t = W_{t-k}$, μ is the mean of W_t,

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p \qquad p \geq 0$$

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q \qquad q \geq 0$$

The model is of order (p, q) and is referred to as an ARMA(p, q) model.

An equivalent version of the ARMA(p, q) model is given by

$$\phi(B)W_t = \theta_0 + \theta(B)A_t \qquad t \in ZZ$$

where θ_0 is an overall constant defined by

$$\theta_0 = \mu\left(1 - \sum_{i=1}^{p} \phi_i\right)$$

See Box and Jenkins (1976, pages 92–93) for a discussion of the meaning and usefulness of the overall constant. The coefficients in the ARMA model can be estimated using MAX_ARMA (page 741).

Parameter estimates for ARMA processes can also be obtained using the MAX_ARMA routine. This routine uses the maximum likelihood method to obtain estimates for the moving average and autoregressive parameters in an ARMA model. This routine also requires initial parameter estimates and further requires that these initial values represent a stationary time series. If they are not stationary, MAX_ARMA replaces these estimates with initial estimates that are stationary. However these may be far away from the values selected to initially describe this series.

Moreover, the method of maximum likelihood for estimating ARMA parameters may not converge to stationary estimates. In this case, MAX_ARMA will display a warning message and sets its convergence parameter ICONV to zero.

If the "raw" data $\{Z_t\}$ are homogeneous nonstationary, then differencing induces stationarity and the model is called autoregressive *integrated* moving average (ARIMA). Parameter estimation is performed on the stationary time series

$$W_t = \nabla^d Z_t$$

where

$$\nabla^d = (1 - B)^d$$

is the backward difference operator with period 1 and order d, $d > 0$.

Typically, routine NSPE (page 727) is first applied to the transformed data to provide preliminary parameter estimates. These estimates are used as initial values in an estimation procedure. In particular, routine NSLSE (page 733) may be used to compute conditional or unconditional least-squares estimates of the parameters, depending on the choice of the backcasting length. Parameter estimates from either NSPE or NSLSE may be input to routine NSBJF (page 752) to produce forecasts with associated probability limits. The routines for preliminary parameter estimation, least squares parameter estimation, and forecasting follow the approach of Box and Jenkins (1976, programs 2–4, pages 498–509).

Transfer Function Model

Define $\{x_t\}$ and $\{y_t\}$ by

$$x_t = \begin{cases} X_t - \hat{\mu}_X & d = 0 \\ \nabla^d X_t & d > 0 \end{cases}$$

and

$$y_t = \begin{cases} Y_t - \hat{\mu}_Y & d = 0 \\ \nabla^d Y_t & d > 0 \end{cases}$$

where $\{X_t\}$ and $\{Y_t\}$ for $t = (-d + 1), \ldots, n$ represent the undifferenced input and undifferenced output series with

$$\hat{\mu}_X \text{ and } \hat{\mu}_Y$$

estimates of their respective means. The differenced input and differenced output series may be obtained using the routine DIFF (page 692) following any preliminary transformation of the data.

The transfer function model is defined by

$$Y_t = \delta^{-1}(B)\omega(B)X_{t-b} + n_t$$

or equivalently,

$$y_t = \delta^{-1}(B)\omega(B)x_{t-b} + n_t$$

where $n_t = \nabla^d N_t$ for $d \geq 0$, and the left-hand side and right-hand side transfer function polynomial operators are, respectively,

$$\delta(B) = 1 - \delta_1 B - \delta_2 B^2 - \ldots - \delta_r B^r$$

$$\omega(B) = \omega_0 - \omega_1 B - \omega_2 B^2 - \ldots - \omega_s B^s$$

with $r \geq 0$, $s \geq 0$, and $b \geq 0$. The noise process $\{N_t\}$ and the input process $\{X_t\}$ are assumed to be independent, with the noise process given by the ARIMA model

$$\phi(B)n_t = \theta(B)A_t$$

where

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p$$

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q$$

with $p \geq 0$ and $q \geq 0$.

The impulse response weights $\{v_k\}$ of the transfer function

$$\nu(B) = \delta^{-1}(B)\omega(B) = \nu_0 + \nu_1 B + \nu_2 B^2 + \ldots$$

and the differenced noise series $\{n_t\}$ are estimated using routine IRNSE (page 758). Preliminary estimates of the transfer function parameters and noise model parameters are computed by routine TFPE (page 762).

Multivariate Autoregressive Time Series

A multivariate autoregressive time series can be expressed as:

$$X_t = \sum_{i=1}^{p} A_i \cdot X_{t-i} + U_t,$$

where

$$X_t = (x_{1,t}, x_{2,t}, \ldots, x_{m,t})'$$

is a column vector containing the values for the m univariate time series at time = t.

$$A_1, A_2, \ldots, A_p$$

are the m x m matrices containing the autoregressive parameters for lags 1, 2, …, p.

$$U_t = (\varepsilon_{1,t}, \varepsilon_{2,t}, \ldots, \varepsilon_{m,t})'$$

is a column vector containing the values for the m white noise values for each time series at time = t.

Akaike's Information Criterion (AIC) can be used to identify the optimum number of lags. For a multivariate autoregressive time series,

$$AIC = (N-p)\cdot \ln(\left\|\hat{\Sigma}_p\right\|) + 2\cdot K + (N-p)(\ln(2\pi)+1),$$

where

N = number of observations in each of the m univariate time series,

K = number of non-zero autoregressive coefficients in the parameter matrices,

p = maximum number of lags used in calculating AIC, and

$\left\|\hat{\Sigma}_p\right\|$ = determinate of the estimated m by m covariance matrix for the white noise, U_t.

Routine AUTO_MUL_AR calculates AIC for selected lags and returns parameter estimates for the optimum lag.

Multichannel Time Series

A multichannel time series X is simply a multivariate time series whose channels correspond to interrelated univariate time series. In this setting, the model-building process is a logical extension of the procedures used to identify, estimate, and forecast univariate time series. In particular, the multichannel cross-correlation function computed by routine MCCF (page 711) may help identify a tentative model. A particular regression model may be fit using routine MLSE (page 768), with the Wiener filter estimated using routine MWFE (page 774). The Wiener forecast function for a single channel may be obtained by routine SPWF (page 749). The state space approach to fitting many time domain models is available through routine KALMN (page 780).

Automatic Model Selection

There are two popular criteria for comparing autoregressive (AR) models with different lags:

- FPE – Final Prediction Error
- AIC – Akaike's Information Criterion.

These are defined for both univariate and multivariate time series. FPE for an autoregressive univariate model with p lags is calculated using the formula:

$$FPE_p = \frac{(N+p+1)}{(N-p-1)}\hat{\sigma}_p^2,$$

where

N = number of observations in the time series, and $\hat{\sigma}_p^2$ = the estimate for the variance of the white noise term in an AR model of order p. The fit with the smallest FPE is considered best.

Similarly, AIC for an AR univariate series is calculated by:

$$AIC = -2\cdot\ln(L)+2p,$$

where

L = the value for the maximum likelihood function for the fitted model, and

p = number of lags for the AR model. In some routines this is approximated by

$$AIC = (N-p)\cdot\ln(\hat{\sigma}_p^2)+2\cdot K+\left(N-p\right)\left(\ln\left(2\pi\right)+1\right),$$

where K= number of nonzero parameters in the model.

Similar to FPE, the fit with the smallest AIC is considered best.

A formula also exist for the final prediction error of a multivariate time series:

$$MFPE_p = \frac{\left(1+\frac{pm+1}{N}\right)^m}{\left(1-\frac{pm+1}{N}\right)^m}\left\|\hat{\Sigma}_p\right\|$$

where m = number of univariate time series, and p = maximum number of lags in the AR model. The equivalent multivariate AIC calculation is

$$AIC = (N-p)\cdot\ln(\left\|\hat{\Sigma}_p\right\|)+2\cdot K+\left(N-p\right)\left(\ln\left(2\pi\right)+1\right)$$

The table below summarizes the four routines for automatic AR fitting categorized as either univariate, or multivariate, and minimum AIC or FPE.

Routine	Variables	Criterion
AUTO_UNI_AR	Univariate	AIC
AUTO_MUL_AR	Multivariate	AIC
AUTO_FPE_UNI_AR	Univariate	FPE
AUTO_UNI__MUL_AR	Multivariate	FPE

Frequency Domain Methodology

An alternative method of time series analysis with much less emphasis on the form of the model may be performed in the frequency domain.

Spectral Analysis

Let $\{X(t)\}$ denote a continuous-parameter stationary process with mean

$$\mu = E[X(t)]$$

and autocovariance function

$$\sigma(k) = \text{cov}\{X(t), X(t+k)\} = \text{E}\{[X(t) - \mu][X(t+k) - \mu]\} \quad k \in \mathbf{R}$$

Similarly, let $\{X_t\}$ denote a discrete-parameter stationary process with mean

$$\mu = E[X_t]$$

and autocovariance function

$$\sigma(k) = \text{cov}\{X_t, X_{t+k}\} = \text{E}\{[X_t - \mu][X_{t+k} - \mu]\} \qquad k \in ZZ$$

Note that $\sigma(0) = \sigma^2$ is the variance of the process.

The routines for the spectral analysis of time series are concerned with the estimation of the spectral density of a stationary process given a finite realization $\{X_t\}$ for $t = 1, \ldots, n$ where $n =$ `NOBS`. This realization consists of values sampled at equally spaced time intervals in the continuous-parameter case or of values observed consecutively in the discrete-parameter case. Hence, we need only develop methodology concerned with the spectral analysis of discrete-parameter stationary processes and later account for the time sampling in the continuous-parameter model.

The nonnormalized spectral density $h(\omega)$ and the autocovariance function $\sigma(k)$ of the stationary process form a Fourier transform pair. The relationship in the continuous-parameter case is given by

$$h(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \sigma(k) e^{-i\omega k} dk$$

$$\sigma(k) = \int_{-\pi}^{\pi} h(\omega) e^{i\omega k} d\omega$$

Similarly, the normalized spectral density $f(\omega)$ and the autocorrelation function $\rho(k) = \sigma(k)/\sigma(0)$ of the stationary process form a Fourier transform pair. The relationship in the continuous-parameter case is given by

$$f(\omega) = \frac{h(\omega)}{\sigma(0)}$$

$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} \rho(k) e^{-i\omega k} dk$$

$$\rho(k) = \int_{-\pi}^{\pi} f(\omega) e^{i\omega k} d\omega$$

The discrete-parameter analogs of the above equations involve summation over *k* instead of integration over *dk*. Also, the normalized spectral density $f(\omega)$ satisfies

$$\int_{-\pi}^{\pi} f(\omega)d\omega = 1$$

Discrete Fourier Transform. The discrete Fourier transform of the sequence $\{Z_t\}$ for $t = 1, \ldots, N$ is defined by

over the discrete set of frequencies

$$\omega_p = \frac{2\pi p}{N} \quad p = 0, \pm 1, \ldots, \pm \lfloor N/2 \rfloor$$

where the function $\lfloor r \rfloor$ determines the greatest integer less than or equal to *r*. An alternative representation of $\zeta(\omega_p)$ in terms of cosine and sine transforms is

$$\zeta(\omega_p) = \alpha(\omega_p) - i\beta(\omega_p)$$

where

$$\alpha(\omega_p) = \sum_{t=1}^{N} Z_t \cos(\omega_p t)$$

$$\beta(\omega_p) = \sum_{t=1}^{N} Z_t \sin(\omega_p t)$$

The fast Fourier transform algorithm implemented in the IMSL MATH/LIBRARY routine `FFTCF` is used to compute the discrete Fourier transform. All of the frequency domain routines that output a periodogram utilize the fast Fourier transform algorithm.

Centering and Padding. Consider the centered and padded realization

for $t = 1, \ldots, N$ defined by

$$\tilde{X}_t = \begin{cases} X_t - \hat{\mu} & t = 1, \ldots, n \\ 0 & t = (n+1), \ldots, N \end{cases} \qquad (1)$$

where $N = (n + n_0)$ and

$$\hat{\mu} = \text{XCNTR}$$

is

$$\hat{\mu} = \begin{cases} \mu & \mu \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} X_t & \mu \text{ unknown} \end{cases} \qquad (2)$$

Centering the data simplifies the formulas for estimation of the periodogram and spectral density. The addition of n_0 = NPAD zeros to the end of the data is called *padding*. This procedure increases the effective length of the data from n to N in an effort to:

- increase the computational efficiency of the Fourier transformation of the series by providing a more suitable series length N (Priestley 1981, page 577).
- obtain the periodogram ordinates required to give the exact expression of the sample autocovariances in terms of the inverse Fourier transformation of the periodogram (Priestley 1981, page 579).
- produce periodogram ordinates over a more refined range of frequencies ω_p.

Any desired filtering, prewhitening, or data tapering should be performed prior to estimating the spectral density. The resulting estimate may be adjusted accordingly.

Periodogram. The periodogram of the sample sequence $\{X_t\}$, $t = 1, \ldots, n$ computed with the centered and padded sequence

$$\{\tilde{X}_t\}, \quad t = 1, \ldots, N$$

is defined by

$$I_{n,N,\tilde{X}}(\omega_p) = K\left|\sum_{t=1}^{N} \tilde{X}_t e^{-i\omega_p t}\right|^2 = K\left|\zeta_{\tilde{X}}(\omega_p)\right|^2$$

where K is the scale factor

$$K = \begin{cases} \dfrac{2}{n} & \text{for the usual periodogram} \\ \dfrac{1}{2\pi n} & \text{for the modified periodogram} \end{cases}$$

The scale factor of the usual periodogram relates the ordinates to the sum of squares of

$$X_t - \hat{\mu}$$

(Fuller 1976, pages 276–277). If the first ordinate (corresponding to $p = 0$) is replaced by one-half of its value, then if N is odd, the sum of the $\lfloor N/2 \rfloor + 1$ ordinates corresponding to $p = 0, 1, \ldots, \lfloor N/2 \rfloor$ is

$$\frac{N}{n}\sum_{t=1}^{n}(X_t - \hat{\mu})^2$$

The modified periodogram is an asymptotically unbiased estimate of the nonnormalized spectral density function at each frequency ω_p (Priestley 1981, page 417). The argument IPVER is used to specify the version of the periodogram.

Spectral Density. The relationship between the sample autocovariance function and estimate of the nonnormalized spectral density function is similar to the theoretical situation previously discussed.

Define the sample autocovariance function of the X_t process by

$$\hat{\sigma}(k) = \frac{1}{n}\sum_{t=1}^{n-|k|}\left\{[X_t - \hat{\mu}][X_{t+|k|} - \hat{\mu}]\right\} \quad k = 0, \pm 1, \ldots, \pm(n-1)$$

where

$$\hat{\mu}$$

is given by Equation 2. Note that

$$\hat{\sigma}(0) = \hat{\sigma}^2$$

is the sample variance. The nonnormalized spectral density may be estimated directly from the sample autocovariances by

$$\hat{h}(\omega) = \frac{1}{2\pi}\sum_{k=-(n-1)}^{(n-1)} \lambda_n(k)\hat{\sigma}(k)e^{-i\omega k}$$

The sequence of weights $\{\lambda_n(k)\}$ called the *lag window* decreases at a rate appropriate for consistent estimation of $h(\omega)$.

An algebraically equivalent method of estimating $h(\omega)$ consists of locally smoothing the modified periodogram in a neighborhood of ω. Let

$$I_{n,N,\tilde{X}}$$

denote the modified periodogram of the centered and padded realization

$$\{\tilde{X}_t\}$$

defined in Equation 1. Then, an estimate of the nonnormalized spectral density is given by

$$\hat{h}(\omega) = \frac{2\pi}{N}\sum_{p=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} I_{n,N,\tilde{X}}(\omega_p)W_n(\omega - \omega_p) \qquad (3)$$

where

$$W_n(\theta) = \frac{1}{2\pi}\sum_{k=-(n-1)}^{(n-1)} \lambda_n(k)e^{-i\theta k}$$

The *spectral window* $W_n(\theta)$ is the discrete Fourier transform of the lag window $\lambda_n(k)$. We note that for $N = 2n - 1$, the modified periodogram and autocovariances,

$$I_{n,2n-1,\tilde{X}}\left(\omega_p\right) \text{ and } \hat{\sigma}\left(k\right)$$

form the discrete Fourier transform pair

$$I_{n,N,\tilde{X}}(\omega_p) = \frac{1}{2\pi}\sum_{k=-(n-1)}^{n-1} \hat{\sigma}(k)e^{-i\omega_p k}, \qquad p = 0, \pm 1, \ldots, \pm(n-1)$$

$$\hat{\sigma}(k) = \frac{2\pi}{N}\sum_{p=-(n-1)}^{n-1} I_{n,N,\tilde{X}}(\omega_p)e^{i\omega_p k}, \quad k = 0, \pm 1, \ldots, \pm(n-1)$$

This relationship is exact and recovers the $(n-1)$ sample autocovariances only when $n_0 = (n-1)$ zeros are padded, since then $\lfloor N/2 \rfloor = (n-1)$.

Another method of estimating $h(\omega)$ is given by

$$\hat{h}(\omega) = \sum_j w_j I_{n,N,\tilde{X}}(\omega_{p,j}) \qquad (4)$$

where

$$\omega_{p,j} = \frac{2\pi\{p(\omega)+j\}}{N}$$

and $p(\omega)$ is the integer such that $\omega_{p,0}$ is closest to ω. The sequence of m weights $\{w_j\}$ for $j = -[m/2], \ldots, (m - [m/2] - 1)$ is fixed in the sense that they do not depend on the frequency, ω, and satisfy $\sum_j w_j = 1$. Priestley (1981, page 581) notes that if we write

$$w_j = \frac{2\pi}{N} W_n(\omega - \omega_{p,j})$$

then Equation 4 and Equation 3 are quite similar except that the weights $\{w_j\}$ depend on ω. In fact, if $p(\omega) = 0$ and $m = N$, these equations are equivalent.

Given estimates

$$\hat{h}(\omega) \text{ and } \hat{\sigma}(0)$$

the estimate of the normalized spectral density is given by

$$\hat{f}(\omega) = \frac{\hat{h}(\omega)}{\hat{\sigma}(0)}$$

This follows directly from the definition of $f(\omega)$.

Spectral Window. The following spectral windows $W_n(\theta)$ are available in routines containing the argument `ISWVER`.

Modified Bartlett

$$W_n(\theta) = \frac{1}{2\pi M}\left\{\frac{\sin(M\theta/2)}{\sin(\theta/2)}\right\}^2 = F_M(\theta)$$

where $F_M(\theta)$ corresponds to the Fejér kernel of order M.

Daniell

$$W_n(\theta) = \begin{cases} M/2\pi & -\pi/M \le \theta \le \pi/M \\ 0 & \text{otherwise} \end{cases}$$

Tukey

$$W_n(\theta) = aD_M\left(\theta - \frac{\pi}{M}\right) + (1-2a)D_M(\theta) + aD_M\left(\theta + \frac{\pi}{M}\right)$$

for $0 < a \le 0.25$, where $D_M(\theta)$ represents the Dirichlet kernel. The Tukey-Hanning window is obtained when $a = 0.23$, and the Tukey-Hamming window is obtained when $a = 0.25$.

Parzen

$$W_n(\theta) = \frac{6\pi}{M}\left[F_{M/2}(\theta)\right]^2\left\{1 - \frac{2}{3}\sin^2(\theta/2)\right\}$$

where M is even. If M is odd, then $M + 1$ is used instead of M in the above formula.

Bartlett-Priestley

$$W_n(\theta) = \begin{cases} \frac{3M}{4\pi}\left\{1 - \left(\frac{M\theta}{\pi}\right)^2\right\} & |\theta| \le \pi/M \\ 0 & |\theta| > \pi/M \end{cases}$$

The *window parameter M* is inversely proportional to the *bandwidth* of the spectral window. Priestley (1981, pages 520–522) discusses a number of definitions of bandwidth and concludes that the particular definition adopted is of little significance. The choice of spectral window bandwidth, and hence, the choice of M, is a more important problem. One practical choice for M is the last lag at which the estimated autocorrelation function

$$\hat{\rho}(k)$$

is significantly different from zero, i.e.,

$$\hat{\rho}(k) \approx 0 \text{ for } k > M$$

The estimated autocorrelations and their associated estimated standard errors can be computed using routine ACF (page 697). See Priestley (1981, pages 528–556) for alternative strategies of determining the window parameter M.

Since the spectral window is the Fourier transform of the lag window, we estimate the spectral density function by application of a particular spectral window to the periodogram. Note that M is directly related to the rate of decay of the lag window.

Time Interval. Consider the continuous-parameter stationary process $\{X(t)\}$ and let $\{X_t\}$ denote a realization of this process sampled at equal time intervals Δt = TINT. Although the spectral density of $X(t)$ extends over the frequency range $(-\pi, \pi)$, the spectral density of X_t is unique over the restricted frequency range $(-\pi/\Delta t, \pi/\Delta t)$. This problem of aliasing or *spectrum folding* is inherent to spectral analysis, see Blackman and Tukey (1958) and Priestley (1981) for further discussion.

In practice, the $\{X_t\}$ realization is treated as a discrete parameter process with spectral density

$$h_X^{\dagger}(\omega)$$

defined over the frequency range $(-\pi, \pi)$. This corresponds to setting $\Delta t = 1$. The transformation of the spectral density to the restricted frequency range $(-\pi/\Delta t, \pi/\Delta t)$ is given by

$$h_X(\omega) = \Delta t \, h_X^{\dagger}(\omega \Delta t) \qquad |\omega| \le \pi / \Delta t$$

Priestley (1981, pages 507–508) considers a method of choosing Δt. A similar transformation is performed for the estimated spectral density.

Frequency Scale. The argument IFSCAL is used to specify the scale of the frequencies at which to estimate the spectral density. The NF frequencies are contained in the argument F.

Approximate Confidence Intervals for Spectral Ordinates. An approximate $(1 - \alpha)100\%$ confidence interval for the value of the nonnormalized spectral density function $h(\omega)$ at a particular frequency ω is given by the formula (Priestley 1981, page 468)

$$\left(\frac{\mathrm{DF} \times \hat{h}(\omega)}{\chi^2_{\mathrm{DF},1-\alpha/2}}, \frac{\mathrm{DF} \times \hat{h}(\omega)}{\chi^2_{\mathrm{DF},\alpha/2}} \right)$$

Routine CHIIN (page 1132) using argument P = $1 - \alpha/2$ and P = $\alpha/2$ can be used to compute the percentage point

$$\chi^2_{\mathrm{DF,P}}$$

Also, routine CHIIN should be used with degrees of freedom (DF), which depend upon the version of the spectral window (ISWVER), as given in the following table (Priestley 1981, page 467).

ISWVER	Window	DF
1	Modified Bartlett	$3n/M$
2	Daniell	$2n/M$
3	Tukey-Hamming	$2.5164n/M$
4	Tukey-Hanning	$2\ 2/3n/M$
5	Parzen	$3.708614n/M$
6	Bartlett-Priestley	$1.4n/M$

If one of the windows above is not specified and the user provides relative weights, such as with routine SWED (page 845), the weights are normalized to sum to one in the actual computations. Given all m (m odd) normalized weights w_j, then for $2\pi\lfloor m/2 \rfloor/n < \omega < \pi(1 - 2\lfloor m/2 \rfloor/n)$ the degrees of freedom for a confidence interval on $h(\omega)$ are given by Fuller (1976, page 296)

$$\mathrm{DF} = \frac{2}{\sum_{j=-\lfloor m/2 \rfloor}^{\lfloor m/2 \rfloor} w_j^2}$$

Frequently, confidence intervals on the $\ln h(\omega)$ are suggested because this produces fixed width intervals. The interval is

$$\left(\ln \hat{h}(\omega) + \ln\left[\frac{\mathrm{DF}}{\chi^2_{\mathrm{DF},1-\alpha/2}} \right], \ln \hat{h}(\omega) + \ln\left[\frac{\mathrm{DF}}{\chi^2_{\mathrm{DF},a/2}} \right] \right)$$

Cross-Spectral Analysis

The routines for cross-spectral analysis are concerned with the estimation of the crossspectral density of two jointly stationary processes given finite realizations $\{X_t\}$ and $\{Y_t\}$ for $t = 1, \ldots, n$. These realizations consist of values sampled at equally spaced time intervals in the continuous-parameter case or of values observed consecutively in the discreteparameter case. Again, we develop methodology concerned with the cross-spectral analysis of discrete-parameter stationary processes and later account for the time sampling in the continuous-parameter model.

Let μ_X and $\sigma_{XX}(k)$ denote the mean and autocovariance function of the X_t process; similarly, define μ_Y and $\sigma_{YY}(k)$, with respect to the Y_t process. Define the cross-covariance function between X_t and Y_t by

$$\sigma_{XY}(k) = \text{cov}\{[X_t - \mu_X][Y_{t+k} - \mu_Y]\} \qquad k \in ZZ$$

Then, the nonnormalized cross-spectral density $h_{XY}(\omega)$ and the cross-covariance function $\sigma_{XY}(k)$ form a Fourier transform pair. The relationship in the continuous-parameter case is given by

$$h_{XY}(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \sigma_{XY}(k)e^{-i\omega k}\,dk$$

$$\sigma_{XY}(k) = \int_{-\pi}^{\pi} h_{XY}(\omega)e^{i\omega k}\,d\omega$$

Similarly, the normalized cross-spectral density $f_{XY}(\omega)$ and the cross-correlation function $\rho_{XY}(k) = \sigma_{XY}(k)/[\sigma_{XX}(0)\sigma_{YY}(0)]$ form a Fourier transform pair. The relationship in the continuous-parameter case is given by

$$f_{XY}(\omega) = \frac{h_{XY}(\omega)}{\sigma_{XX}(0)\sigma_{YY}(0)} \qquad = \frac{1}{2\pi}\int_{-\infty}^{\infty} \rho_{XY}(k)e^{-i\omega k}\,dk$$

$$\rho_{XY}(k) = \int_{-\pi}^{\pi} f_{XY}(\omega)e^{i\omega k}\,d\omega$$

The discrete-parameter analogs of the above equations involve summation over k instead of integration over dk.

The cross-spectral density function is often written in terms of real and imaginary components, since in general, the function is complex-valued. In particular,

$$h_{XY}(\omega) = c_{XY}(\omega) - iq_{XY}(\omega)$$

where the *cospectrum* and *quadrature spectrum* of the X_t and Y_t process are respectively defined by

$$c_{XY}(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \tfrac{1}{2}[\sigma_{XY}(k) + \sigma_{XY}(-k)]\cos k\omega\,dk$$

$$q_{XY}(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \tfrac{1}{2}[\sigma_{XY}(k) - \sigma_{XY}(-k)]\sin k\omega\,dk$$

The *polar* form of $h_{XY}(\omega)$ is defined by

$$h_{XY}(\omega) = \alpha_{XY}(\omega)e^{i\phi_{XY}(\omega)}$$

where the *cross-amplitude spectrum* is

$$\alpha_{XY}(\omega) = \left|h_{XY}(\omega)\right| = \left[c_{XY}^2(\omega) + q_{XY}^2(\omega)\right]^{1/2}$$

and the *phase spectrum* is

$$\phi_{XY}(\omega) = \tan^{-1}[-q_{XY}(\omega)/c_{XY}(\omega)]$$

The *coherency spectrum* is defined by

$$w_{XY}(\omega) = \frac{h_{XY}(\omega)}{[h_{XX}(\omega)h_{YY}(\omega)]^{1/2}}$$

For a given frequency ω, the *coherency* $|w_{XY}(\omega)|$ lies between zero and one, inclusive, and reflects the linear relationship between the random coefficients. See Priestley (1981, pages 654–661) for additional information concerning the interpretation of the components of the cross-spectral density.

Centering and Padding. The centered and padded realizations

$$\left\{\tilde{X}_t\right\} \text{ and } \left\{\tilde{Y}_t\right\}$$

are defined as in Equation 1 with centering constants

$$\hat{\mu}_X = \text{XCNTR} \text{ and } \hat{\mu}_Y = \text{YCNTR}$$

Any desired filtering, prewhitening, or data tapering should be performed prior to estimating the crossspectral density. The resulting estimate may be adjusted accordingly.

Cross Periodogram. The cross periodogram of the sample sequences $\{X_t\}$ and $\{Y_t\}$, $t = 1, \ldots, n$ computed with the padded sequences

$$\left\{\tilde{X}_t\right\} \text{ and } \left\{\tilde{Y}_t\right\}$$

$t = 1, \ldots, N$ is defined by

$$I_{n,N,\tilde{X}\tilde{Y}}(\omega_p) = \left(\sum_{t=1}^{N}\tilde{X}_t e^{-i\omega_p t}\right)\left(\sum_{t=1}^{N}\tilde{Y}_t e^{i\omega_p t}\right) = K\zeta_{\tilde{X}}(\omega_p)\zeta_{\tilde{Y}}^*(\omega_p)$$

where K is the scale factor

$$K = \begin{cases} \dfrac{2}{n} & \text{for the usual cross periodogram} \\ \dfrac{1}{2\pi n} & \text{for the modified cross periodogram} \end{cases}$$

The scale factor option is maintained for compatibility with the spectral routines. The argument `IPVER` is used to specify the version of the periodogram used to compute the cross periodogram.

Cross-Spectral Density Estimation. The relationship between the sample cross-covariance function and estimate of the nonnormalized cross-spectral density function is similar to the theoretical situation previously discussed.

Define the sample cross-covariance function between the X_t and Y_t process by

$$\tilde{\sigma}_{XY}(k) = \begin{cases} \frac{1}{n}\sum_{t=1}^{n-k} \left\{\left[X_t - \hat{\mu}_X\right]\left[Y_{t+k} - \hat{\mu}_Y\right]\right\} & k = 0,1,\ldots,(n-1) \\ \frac{1}{n}\sum_{t=1-k}^{n} \left\{\left[X_t - \hat{\mu}_X\right]\left[Y_{t+k} - \hat{\mu}_Y\right]\right\} & k = -1,-2,\ldots,-(n-1) \end{cases}$$

The nonnormalized cross-spectral density may be estimated directly from the sample cross-covariances by

$$\tilde{h}_{XY}(\omega) = \frac{1}{2\pi}\sum_{k=-(n-1)}^{(n-1)} \lambda_n(k)\hat{\sigma}_{XY}(k)e^{-i\omega k}$$

The sequence of weights $\{\lambda_n(k)\}$ called the *lag window* decreases at a rate appropriate for consistent estimation of $h_{XY}(\omega)$.

An algebraically equivalent method of estimating $h_{XY}(\omega)$. consists of locally smoothing the modified cross periodogram in a neighborhood of ω. Let

$$I_{n,N,\tilde{X}\tilde{Y}}$$

denote the modified cross periodogram of the centered and padded realizations

$$\{\tilde{X}_t\} \text{ and } \{\tilde{Y}_t\}$$

Then, an estimate of the nonnormalized cross-spectral density is given by

$$\hat{h}_{XY}(\omega) = \frac{2\pi}{N}\sum_{p=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} I_{n,N,\tilde{X}\tilde{Y}}(\omega_p)W_n(\omega - \omega_p) \quad (5)$$

where $W_n(\theta)$ is the spectral window.

Another method of estimating $h_{XY}(\omega)$ is given by

$$\hat{h}_{XY}(\omega) = \sum_j w_j I_{n,N,\tilde{X}\tilde{Y}}(\omega_{p,j}) \quad (6)$$

where $\omega_{p,j}$, $p(\omega)$, and the weights $\{w_j\}$ are as defined in the univariate setting.

Given estimates

$$\hat{h}_{XY}(\omega), \hat{\sigma}_{XX}(0), \text{and } \hat{\sigma}_{YY}(0)$$

the estimate of the normalized cross-spectral density is given by

$$\hat{f}_{XY}(\omega) = \frac{\hat{h}_{XY}(\omega)}{\hat{\sigma}_{XX}(0)\hat{\sigma}_{YY}(0)}$$

This follows directly from the definition of $f_{XY}(\omega)$.

BCTR

Performs a forward or an inverse Box-Cox (power) transformation.

Required Arguments

Z — Vector of length NOBS containing the data. (Input)

POWER — Exponent parameter in the power transformation. (Input)

SHIFT — Shift parameter in the power transformation. (Input)
SHIFT must satisfy the relation min(Z(*i*)) + SHIFT > 0.

X — Vector of length NOBS containing the transformed data. (Output)
If Z is not needed, then X and Z can occupy the same storage locations. In this case, IPRINT = 1 will print two identical vectors.

Optional Arguments

NOBS — Number of observations in Z. (Input)
NOBS must be greater than or equal to one.
Default: NOBS = size (Z,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints Z and the transformed data, X.

IDIR — Direction of transformation option. (Input)
Default: IDIR = 0.

IDIR	Action
0	Forward transformation.
1	Inverse transformation.

FORTRAN 90 Interface

Generic: `CALL BCTR (Z, POWER, SHIFT, X [,…])`

Specific: The specific interface names are `S_BCTR` and `D_BCTR`.

FORTRAN 77 Interface

Single: `CALL BCTR (NOBS, Z, IPRINT, IDIR, POWER, SHIFT, X)`

Double: The double precision name is `DBCTR`.

Example 1

Consider the Airline Data (Box and Jenkins 1976, page 531) consisting of the monthly total number of international airline passengers from January 1949 through December 1960. Routine `BCTR` is used to compute a forward Box-Cox transformation of the first 12 observations. In the transformation `SHIFT` and `POWER` are each set to zero, which corresponds to taking natural logarithms of the data.

```
      USE GDATA_INT
      USE BCTR_INT
      INTEGER IPRINT, NOBS
      PARAMETER (IPRINT=1, NOBS=12)
!
      INTEGER    NCOL, NROW
      REAL       POWER, SHIFT, X(NOBS), Z(144, 1)
!                                Airline Data
      CALL GDATA (4, Z, NROW, NCOL)
!                                Forward direction
!                                Transformation parameters
      POWER = 0.0
      SHIFT = 0.0
!                                Compute natural logarithms of
!                                first 12 observations in Z
      CALL BCTR (Z(:,1), POWER, SHIFT, X, NOBS=NOBS, IPRINT=IPRINT)
!
      END
```

Output

```
Output from BCTR

 I        Z           X

 1      112.00      4.7185
 2      118.00      4.7707
 3      132.00      4.8828
 4      129.00      4.8598
 5      121.00      4.7958
 6      135.00      4.9053
 7      148.00      4.9972
 8      148.00      4.9972
```

```
 9      136.00      4.9127
10      119.00      4.7791
11      104.00      4.6444
12      118.00      4.7707
```

Comments

1. Informational errors

Type	Code	
4	1	For the specified forward transformation, the minimum element of X will underflow.
4	2	For the specified forward transformation, the maximum element of X will overflow.
4	3	For the specified inverse transformation, the maximum element of X will overflow.
4	4	For the specified inverse transformation, the minimum element of X will underflow.

2. The forward transformation is performed prior to fitting a model. Differencing of the data is done after the data are transformed.

3. The inverse transformation is performed on results such as forecasts and their corresponding probability limits.

Description

Routine BCTR performs a forward or inverse Box-Cox transformation of the n = NOBS observations $\{Z_t\}$ for $t = 1, 2, \ldots, n$.

The forward transformation is useful in the analysis of linear models or models with nonnormal errors or nonconstant variance (Draper and Smith 1981, page 222). In the time series setting, application of the appropriate transformation and subsequent differencing of a series may enable model identification and parameter estimation in the class of homogeneous stationary autoregressive-moving average models. The inverse transformation may later be applied to certain results of the analysis, such as forecasts and probability limits of forecasts, in order to express the results in the scale of the original data. A brief note concerning the choice of transformations in ARIMA models is given in Box and Jenkins (1976, page 328). The class of power transformations discussed by Box and Cox (1964) is defined by

$$X_t = \begin{cases} \dfrac{(Z_t + \xi)^{\lambda} - 1}{\lambda} & \lambda \neq 0 \\ \ln(Z_t + \xi) & \lambda = 0 \end{cases}$$

where $Z_t + \xi > 0$ for all t. Since

$$\lim_{\lambda \to 0} \frac{(Z_t + \xi)^{\lambda} - 1}{\lambda} = \ln(Z_t + \xi)$$

the family of power transformations is continuous.

Let λ = POWER and ξ = SHIFT; then, the computational formula utilized by routine BCTR is given by

$$X_t = \begin{cases} (Z_t + \xi)^{\lambda} & \lambda \neq 0 \\ \ln(Z_t + \xi) & \lambda = 0 \end{cases}$$

where $Z_t + \xi > 0$ for all t. The computational and Box-Cox formulas differ only in the scale and the origin of the transformed data. Consequently, the general analysis of the data is unaffected (Draper and Smith 1981, page 225).

The inverse transformation is computed by

$$X_t = \begin{cases} Z_t^{1/\lambda} - \xi & \lambda \neq 0 \\ \exp(Z_t) - \xi & \lambda = 0 \end{cases}$$

where $\{Z_t\}$ now represents the result computed by BCTR for a forward transformation of the original data using parameters λ and ξ.

Example 2

The estimated standard errors of forecasts (lead times 1 through 12 at origin July 1957) using the transformed Airline Data (Box and Jenkins 1976, page 311) may be converted back to their original scale using routine BCTR. The backward Box-Cox transformation with SHIFT and POWER each set to zero corresponds to using the exponential function.

```
      USE BCTR_INT
      INTEGER    NOBS
      PARAMETER  (NOBS=12)
!
      INTEGER    IDIR, IPRINT
      REAL       POWER, SD(NOBS), SHIFT, X(NOBS)
!                                Standard errors of forecasts
      DATA SD/3.7, 4.3, 4.8, 5.3, 5.8, 6.2, 6.6, 6.9, 7.2, 7.6, 8.0, &
          8.2/
!
      SD=SD * 1.0E-2
!                                Backward direction
      IDIR = 1
!                                Transformation parameters
      POWER = 0.0
      SHIFT = 0.0
!                                Transform standard errors from
!                                log scale to original scale
      IPRINT = 1
      CALL BCTR (SD, POWER, SHIFT, X, IPRINT=IPRINT, IDIR=IDIR)
!
      END
```

Output

```
 Output from BCTR

 I        Z           X
```

```
 1    0.037000      1.0377
 2    0.043000      1.0439
 3    0.048000      1.0492
 4    0.053000      1.0544
 5    0.058000      1.0597
 6    0.062000      1.0640
 7    0.066000      1.0682
 8    0.069000      1.0714
 9    0.072000      1.0747
10    0.076000      1.0790
11    0.080000      1.0833
12    0.082000      1.0855
```

DIFF

Differences a time series.

Required Arguments

Z — Vector of length NOBSZ containing the time series. (Input)

IPER — Vector of length NDIFF containing the periods at which Z is to be differenced. (Input)
The elements of IPER must be greater than or equal to one.

IORD — Vector of length NDIFF containing the order of each difference given in IPER. (Input)
The elements of IORD must be greater than or equal to zero.

NOBSX — Number of observations in the differenced series X. (Output)
NOBSX = NOBSZ − IMISS * NLOST.

X — Vector of length NOBSX containing the differenced series. (Output)

Optional Arguments

NOBSZ — Number of observations in the time series Z. (Input)
NOBSZ must be greater than or equal to one.
Default: NOBSZ = size (Z,1).

NDIFF — Number of differences to perform. (Input)
NDIFF must be greater than or equal to one.
Default: NDIFF = size (IPER,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the number of observations lost because of differencing Z, the number of observations in the differenced series X, and the differenced series X.

IMISS — Missing value option. (Input)
Default: IMISS = 0.

IMISS	Action
0	Include missing values in X.
1	Exclude missing values from X.

NLOST — Number of observations lost because of differencing the time series Z. (Output)
NLOST = IPER(1) * IORD(1) + … + IPER(NDIFF) * IORD(NDIFF).

FORTRAN 90 Interface

Generic: `CALL DIFF (Z, IPER, IORD, NOBSX, X [,…])`

Specific: The specific interface names are `S_DIFF` and `D_DIFF`.

FORTRAN 77 Interface

Single:
```
CALL DIFF (NOBSZ, Z, NDIFF, IPER, IORD, IPRINT,IMISS,
NLOST, NOBSX, X)
```

Double: The double precision name is `DDIFF`.

Example

Consider the Airline Data (Box and Jenkins 1976, page 531) consisting of the monthly total number of international airline passengers from January 1949 through December 1960. Routine DIFF is used to compute

$$X_t = \nabla_1 \nabla_{12} X_t = (Z_t - Z_{t-12}) - (Z_{t-1} - Z_{t-13})$$

For the first invocation of DIFF with IMISS = 0, $X_1, X_2, \ldots, X_{13}$ are set to the missing value code (NaN) and the equation is applied for $t = 14, 15, \ldots, 24$. For the second invocation of DIFF with IMISS = 1, the missing values are excluded from the output array containing the differenced series.

```
USE GDATA_INT
USE DIFF_INT
INTEGER    IPRINT, NDIFF, NOBSZ
```

```
      PARAMETER  (IPRINT=1, NDIFF=2, NOBSZ=24)
!
      INTEGER    IMISS, IORD(NDIFF), IPER(NDIFF), NCOL, NLOST, NOBSX, &
                 NROW
      REAL       X(NOBSZ), Z(144, 1)
!                                Periods of differencing
      DATA IPER/1, 12/
!                                Orders of differencing
      DATA IORD/1, 1/
!                                Airline Data
      CALL GDATA (4, Z, NROW, NCOL)
!                                 Nonseasonal and seasonal difference
!                                 first 24 observations in Z
!
!                                 Include missing values in result X
!                                 USE Default IMISS = 0
      CALL DIFF (Z(:, 1), IPER, IORD, NOBSX, X, NOBSZ=NOBSZ, IPRINT=IPRINT)
!                                 Exclude missing values in result X
      IMISS = 1
      CALL DIFF (Z(:, 1), IPER, IORD, NOBSX, X, IPRINT=IPRINT, &
                 NOBSZ=NOBSZ, IMISS=IMISS)
!
      END
```

Output

```
Output from DIFF/D2FF

NLOST = 13
NOBSX = 24

 I     Z(I)     X(I)
 1   112.00      NaN
 2   118.00      NaN
 3   132.00      NaN
 4   129.00      NaN
 5   121.00      NaN
 6   135.00      NaN
 7   148.00      NaN
 8   148.00      NaN
 9   136.00      NaN
10   119.00      NaN
11   104.00      NaN
12   118.00      NaN
13   115.00      NaN
14   126.00    5.000
15   141.00    1.000
16   135.00   -3.000
17   125.00   -2.000
18   149.00   10.000
19   170.00    8.000
20   170.00    0.000
21   158.00    0.000
22   133.00   -8.000
23   114.00   -4.000
24   140.00   12.000
```

```
Output from DIFF/D2FF

NLOST = 13
NOBSX = 11

 I     Z(I)      X(I)
 1   112.00     5.000
 2   118.00     1.000
 3   132.00    -3.000
 4   129.00    -2.000
 5   121.00    10.000
 6   135.00     8.000
 7   148.00     0.000
 8   148.00     0.000
 9   136.00    -8.00
10   119.00    -4.000
11   104.00    12.000
12   118.00
13   115.00
14   126.00
15   141.00
16   135.00
17   125.00
18   149.00
19   170.00
20   170.00
21   158.00
22   133.00
23   114.00
24   140.00
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `D2FF/DD2FF`. The reference is:

   ```
   CALL D2FF (NOBSZ, Z, NDIFF, IPER, IORD, IPRINT, IMISS, NLOST,
   NOBSX, X, XWK)
   ```

   The additional argument is:

   XWK — Work vector of length equal to `NOBSZ`.

2. A value is considered to be missing if it is not itself in the data set or if it is the result of an operation involving missing value(s). In differencing, missing values occur at the beginning of the differenced series since $X(i) = Z(i) - Z(i-k)$ is not defined for k greater than or equal to i.

Description

Routine DIFF performs m = NDIFF successive backward differences of period s_i= IPER(i) and order d_i = IORD(i) for $i = 1, \ldots, m$ on the n = NOBSZ observations $\{Z_t\}$ for $t = 1, 2, \ldots, n$.

Consider the backward shift operator B given by

$$B^k Z_t = Z_{t-k}, \text{ for all } k$$

Then, the *backward difference operator* with period s is defined by

$$\nabla_s Z_t = (1 - B^s) Z_t = Z_t - Z_{t-s}, \; s \geq 0$$

Note that $B^s Z_t$ and $\nabla_s Z_t$ are defined only for $t = (s + 1), \ldots, n$. Repeated differencing with period s is simply

$$\nabla_s^d Z_t = (1 - B^s)^d Z_t = \sum_{j=0}^{d} \frac{d!}{j!(d-j)!} (-1)^j B^{sj} Z_t$$

where $d \geq 0$ is the order of differencing. Note that

$$\nabla_s^d Z_t$$

is defined only for $t = (sd + 1), \ldots, n$.

The general difference formula used in routine DIFF is given by

$$X_t = \begin{cases} \text{NaN} & t = 1, \ldots, n_L \\ \nabla_{s_1}^{d_1} \nabla_{s_2}^{d_2} \cdots \nabla_{s_m}^{d_m} Z_t & t = n_L + 1, \ldots, n \end{cases}$$

where n_L = NLOST represents the number of observations "lost" because of differencing and NaN (not a number) represents the missing value code. See the routine AMACH (see the Reference Material section of this manual) in the "Machine-Dependent Constants" section of the Reference Material. Note that $n_L = \sum_j s_j d_j$.

A homogeneous stationary time series may be arrived at by appropriately differencing a homogeneous nonstationary time series (Box and Jenkins 1976, page 85). Preliminary application of an appropriate transformation followed by differencing of a series may enable model identification and parameter estimation in the class of homogeneous stationary autoregressive-moving average models.

ACF

Computes the sample autocorrelation function of a stationary time series.

Required Arguments

X — Vector of length NOBS containing the time series. (Input)

MAXLAG — Maximum lag of autocovariances, autocorrelations, and standard errors of autocorrelations to be computed. (Input)
MAXLAG must be greater than or equal to one and less than NOBS.

AC — Vector of length MAXLAG + 1 containing the autocorrelations of the time series X. (Output)
AC(0) = 1. AC(*k*) contains the autocorrelation of lag *k* where *k* = 1, …, MAXLAG.

Optional Arguments

NOBS — Number of observations in the time series X. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the mean and variance.
2	Prints the mean, variance, and autocovariances.
3	Prints the mean, variance, autocovariances, autocorrelations, and standard errors of autocorrelations.

ISEOPT — Option for computing standard errors of autocorrelations. (Input)
Default: ISEOPT = 0.

ISEOPT	Action
0	No standard errors of autocorrelations are computed.
1	Compute standard errors of autocorrelations using Bartlett's formula.
2	Compute standard errors of autocorrelations using Moran's formula.

IMEAN — Option for computing the mean. (Input)
Default: IMEAN = 1.

IMEAN	Action
0	XMEAN is user specified.
1	XMEAN is set to the arithmetic mean of X.

XMEAN — Estimate of the mean of time series X. (Input, if IMEAN = 0; output, if IMEAN = 1)

ACV — Vector of length MAXLAG + 1 containing the variance and autocovariances of the time series X. (Output)
ACV(0) contains the variance of the series X. ACV(k) contains the autocovariance of lag k where k = 1, …, MAXLAG.

SEAC — Vector of length MAXLAG containing the standard errors of the autocorrelations of the time series X. (Output)

The standard error of AC(k) is SEAC(k) where k = 1, …, MAXLAG. If ISEOPT = 0, then SEAC may be dimensioned of length 1.

FORTRAN 90 Interface

Generic: `CALL ACF (X, MAXLAG, AC [,…])`

Specific: The specific interface names are S_ACF and D_ACF.

FORTRAN 77 Interface

Single:
```
CALL ACF (NOBS, X, IPRINT, ISEOPT, IMEAN, XMEAN, MAXLAG,
ACV, AC, SEAC)
```

Double: The double precision name is DACF.

Example

Consider the Wolfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Routine ACF computes the estimated autocovariances, estimated autocorrelations, and estimated standard errors of the autocorrelations.

```
      USE GDATA_INT
      USE ACF_INT
      INTEGER    IPRINT, MAXLAG, NOBS
      PARAMETER  (IPRINT=3, MAXLAG=20, NOBS=100)
!
      INTEGER    IMEAN, ISEOPT, NCOL, NROW
      REAL       AC(0:MAXLAG), ACV(0:MAXLAG), RDATA(176,2), &
                 SEAC(MAXLAG), X(NOBS), XMEAN
!
      EQUIVALENCE (X(1), RDATA(22,2))
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Compute standard errors
      ISEOPT = 1
```

```
!                                  Center on arithmetic mean
!                                  USE DEFAULT IMEAN = 1
!                                  Compute sample ACF
      CALL ACF (X, MAXLAG, AC, IPRINT=IPRINT, ISEOPT=ISEOPT)
!
      END
```

Output

```
Output from ACF/A2F

Mean     =     46.976
Variance =     1382.9
```

Lag	ACV	AC	SEAC
0	1382.9	1.00000	
1	1115.0	0.80629	0.03478
2	592.0	0.42809	0.09624
3	95.3	0.06891	0.15678
4	-236.0	-0.17062	0.20577
5	-370.0	-0.26756	0.23096
6	-294.3	-0.21278	0.22899
7	-60.4	-0.04371	0.20862
8	227.6	0.16460	0.17848
9	458.4	0.33146	0.14573
10	567.8	0.41061	0.13441
11	546.1	0.39491	0.15068
12	398.9	0.28848	0.17435
13	197.8	0.14300	0.19062
14	26.9	0.01945	0.19549
15	-77.3	-0.05588	0.19589
16	-143.7	-0.10394	0.19629
17	-202.0	-0.14610	0.19602
18	-245.4	-0.17743	0.19872
19	-230.8	-0.16691	0.20536
20	-142.9	-0.10332	0.20939

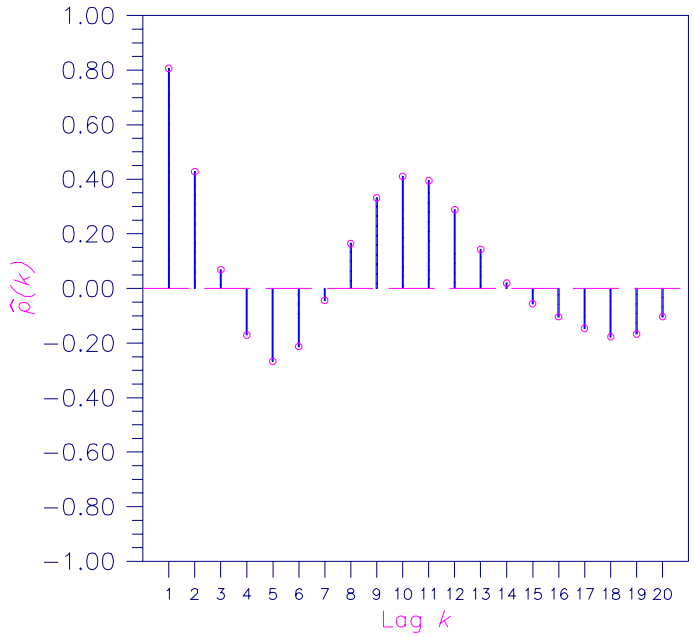


Figure 8-1 Sample Autocorrelation Function

Description

Routine ACF estimates the autocorrelation function of a stationary time series given a sample of n = NOBS observations $\{X_t\}$ for $t = 1, 2, \ldots, n$.

Let

$$\hat{\mu} = \text{XMEAN}$$

be the estimate of the mean μ of the time series $\{X_t\}$ where

$$\hat{\mu} = \begin{cases} \mu, & \mu \text{ known} \\ \dfrac{1}{n}\displaystyle\sum_{t=1}^{n} X_t & \mu \text{ unknown} \end{cases}$$

The autocovariance function $\sigma(k)$ is estimated by

$$\hat{\sigma}(k) = \frac{1}{n}\sum_{t=1}^{n-k}(X_t - \hat{\mu})(X_{t+k} - \hat{\mu}), \quad k = 0,1,\ldots,K$$

where K = MAXLAG. Note that

$$\hat{\sigma}(0)$$

is an estimate of the sample variance. The autocorrelation function $\rho(k)$ is estimated by

$$\hat{\rho}(k) = \frac{\hat{\sigma}(k)}{\hat{\sigma}(0)}, \qquad k = 0,1, \ldots, K$$

Note that

$$\hat{\rho}(0) \equiv 1$$

by definition.

The standard errors of the sample autocorrelations may be optionally computed according to argument ISEOPT. One method (Bartlett 1946) is based on a general asymptotic expression for the variance of the sample autocorrelation coefficient of a stationary time series with independent, identically distributed normal errors. The theoretical formula is

$$\text{var}\{\hat{\rho}(\text{k})\} = \frac{1}{n}\sum_{i=-\infty}^{\infty}\left[\rho^2(i) + \rho(i-k)\rho(i+k) - 4\rho(i)\rho(k)\rho(i-k) + 2\rho^2(i)\rho^2(k)\right]$$

where

$$\hat{\rho}(k)$$

assumes μ is unknown. For computational purposes, the autocorrelations $\rho(k)$ are replaced by their estimates

$$\hat{\rho}(k)$$

for $|k| \le K$, and the limits of summation are bounded because of the assumption that $\rho(k) = 0$ for all k such that $|k| > K$.

A second method (Moran 1947) utilizes an exact formula for the variance of the sample autocorrelation coefficient of a random process with independent, identically distributed normal errors. The theoretical formula is

$$\text{var}\{\hat{\rho}(k)\} = \frac{n-k}{n(n+2)}$$

where μ is assumed to be equal to zero. Note that this formula does not depend on the autocorrelation function.

PACF

Computes the sample partial autocorrelation function of a stationary time series.

Required Arguments

MAXLAG — Maximum lag of partial autocorrelations to be computed. (Input)

AC — Vector of length MAXLAG+ 1 containing the autocorrelations of the time series X. (Input)
AC(0) = 1. AC(k) contains the autocorrelation of lag k where k = 1, …, MAXLAG.

PAC — Vector of length MAXLAG containing the partial autocorrelations of the time series X. (Output)
The partial autocorrelation of lag k corresponds to PAC(k) where $k = 1, \ldots,$ MAXLAG.

FORTRAN 90 Interface

Generic: `CALL PACF (MAXLAG, AC, PAC)`

Specific: The specific interface names are S_PACF and D_PACF.

FORTRAN 77 Interface

Single: `CALL PACF (MAXLAG, AC, PAC)`

Double: The double precision name is DPACF.

Example

Consider the Wolfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Routine PACF to used to compute the estimated partial autocorrelations.

```
      USE GDATA_INT
      USE ACF_INT
      USE PACF_INT
      USE WRRRL_INT
      INTEGER    IMEAN, IPRINT, ISEOPT, MAXLAG, NOBS
      PARAMETER  (IMEAN=1, IPRINT=0, ISEOPT=0, MAXLAG=20, NOBS=100)
!
      INTEGER    NCOL, NROW
      REAL       AC(0:MAXLAG), ACV(0:MAXLAG), PAC(MAXLAG), &
                 RDATA(176,2), SEAC(1), X(NOBS), XMEAN
      CHARACTER  CLABEL(2)*4, RLABEL(1)*6
!
      EQUIVALENCE (X(1), RDATA(22,2))
!
      DATA RLABEL/'NUMBER'/, CLABEL/'Lag ', 'PACF'/
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Compute sample ACF
      CALL ACF (X, MAXLAG, AC)
!                                Compute sample PACF
      CALL PACF (MAXLAG, AC, PAC)
!                                Print results
      CALL WRRRL (' ', PAC, RLABEL, CLABEL, FMT= '(F8.3)')
!
      END
```

Output

```
 Lag       PACF
  1      0.806
  2     -0.635
  3      0.078
  4     -0.059
  5     -0.001
  6      0.172
  7      0.109
  8      0.110
  9      0.079
 10      0.079
 11      0.069
 12     -0.038
 13      0.081
 14      0.033
 15     -0.035
 16     -0.131
 17     -0.155
 18     -0.119
 19     -0.016
 20     -0.004
```

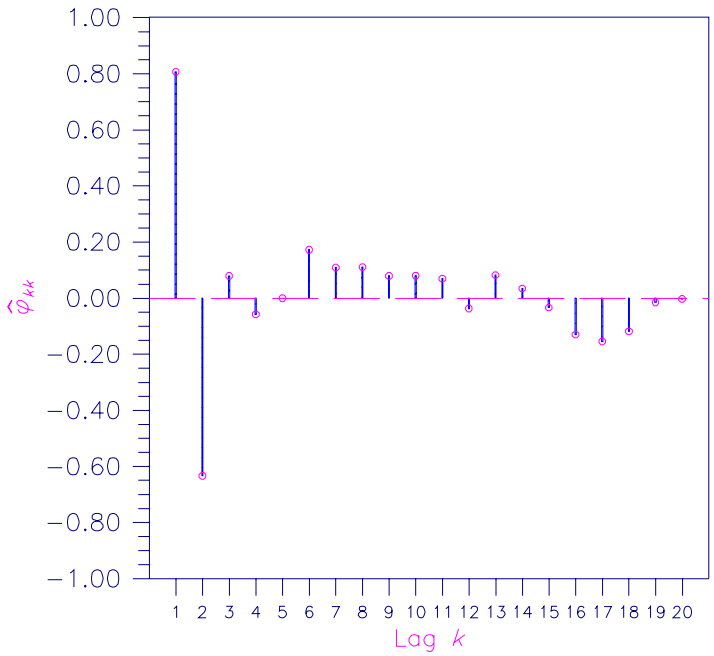


Figure 8-2 Sample Partial Autocorrelation Function

Comments

Workspace may be explicitly provided, if desired, by use of P2CF/DP2CF. The reference is:

```
CALL P2CF (MAXLAG, AC, PAC, WK)
```

The additional argument is:

WK — Work vector of length 2 * MAXLAG.

Description

Routines PACF estimates the partial autocorrelations of a stationary time series given the K = MAXLAG sample autocorrelations

$$\hat{\rho}(k)$$

for $k = 0, 1, \ldots, K$. Consider the AR(k) process defined by

$$X_t = \phi_{k1} X_{t-1} + \phi_{k2} X_{t-2} + \cdots + \phi_{kk} X_{t-k} + A_t$$

where ϕ_{kj} denotes the j-th coefficient in the process. The set of estimates

$$\{\hat{\phi}_{kk}\}$$

for $k = 1, \ldots, K$ is the sample partial autocorrelation function. The autoregressive parameters

$$\{\hat{\phi}_{kj}\}$$

for $j = 1, \ldots, k$ are approximated by Yule-Walker estimates for successive AR(k) models where $k = 1, \ldots, K$. Based on the sample Yule-Walker equations

$$\hat{\rho}(j) = \hat{\phi}_{k1}\hat{\rho}(j-1) + \hat{\phi}_{k2}\hat{\rho}(j-2) + \cdots + \hat{\phi}_{kk}\hat{\rho}(j-k), \quad j = 1, 2, \ldots, k$$

a recursive relationship for $k = 1, \ldots, K$ was developed by Durbin (1960). The equations are given by

$$\hat{\phi}_{kk} = \begin{cases} \hat{\rho}(1) & k = 1 \\ \dfrac{\hat{\rho}(k) - \sum_{j=1}^{k-1} \hat{\phi}_{k-1,j}\hat{\rho}(k-j)}{1 - \sum_{j=1}^{k-1} \hat{\phi}_{k-1,j}\hat{\rho}(j)} & k = 2, \ldots, K \end{cases}$$

and

$$\hat{\phi}_{kj} = \begin{cases} \hat{\phi}_{k-1,j} - \hat{\phi}_{kk}\hat{\phi}_{k-1,k-j} & j = 1, 2, \ldots, k-1 \\ \hat{\phi}_{kk} & j = k \end{cases}$$

This procedure is sensitive to rounding error and should not be used if the parameters are near the nonstationarity boundary. A possible alternative would be to estimate $\{\phi_{kk}\}$ for successive AR(k) models using least squares (IMSL routine NSLSE, page 733) or maximum likelihood. Based on the hypothesis that the true process is AR(p), Box and Jenkins (1976, page 65) note

$$\text{var}\{\hat{\phi}_{kk}\} \simeq \frac{1}{n} \quad k \geq p+1$$

See Box and Jenkins (1976, pages 82–84) for more information concerning the partial autocorrelation function.

CCF

Computes the sample cross-correlation function of two stationary time series.

Required Arguments

X — Vector of length NOBS containing the first time series. (Input)
NOBS must be greater than or equal to two.

Y — Vector of length NOBS containing the second time series. (Input)

MAXLAG — Maximum lag of cross-covariances and cross-correlations to be computed. (Input)
MAXLAG must be greater than or equal to one and less than NOBS.

CC — Vector of length 2 * MAXLAG + 1 containing the cross-correlations between the time series X and Y. (Output)
The cross-correlation between X and Y at lag *k* corresponds to CC(*k*) where *k* = –MAXLAG, …, –1, 0, 1, …, MAXLAG.

Optional Arguments

XMEAN — Estimate of the mean of time series X. (Input, if IMEAN = 0; output, if IMEAN = 1)
Default: XMEAN = 0.0.

YMEAN — Estimate of the mean of time series Y. (Input, if IMEAN = 0; output, if IMEAN = 1)
Default: YMEAN = 0.0.

XVAR — Variance of the time series X. (Output)

YVAR — Variance of the time series Y. (Output)

CCV — Vector of length 2 * MAXLAG + 1 containing the cross-covariances between the time series X and Y. (Output)

The cross-covariance between X and Y at lag *k* corresponds to CCV(*k*) where *k* = –MAXLAG, …, –1, 0, 1, …, MAXLAG.

NOBS — Number of observations in each time series. (Input)
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the means and variances.
2	Prints the means, variances, and cross-covariances.
3	Prints the means, variances, cross-covariances, cross-correlations, and standard errors of cross-correlations.

ISEOPT — Option for computing standard errors of cross correlations. (Input)
Default: `ISEOPT` = 0.

ISEOPT	Action
0	No standard errors of cross-correlations are computed.
1	Compute standard errors of cross-correlations using Bartlett's formula.
2	Compute standard errors of cross-correlations using Bartlett's formula with the assumption of no cross-correlation.

IMEAN — Option for computing the mean. (Input)
Default: `IMEAN` = 1.

IMEAN	Action
0	`XMEAN` and `YMEAN` are user specified.
1	`XMEAN` and `YMEAN` are set to the arithmetic means of `X` and `Y`.

SECC — Vector of length 2 * `MAXLAG` + 1 containing the standard errors of the crosscorrelations between the time series `X` and `Y`. (Output)
The standard error of `CC`(k) is `SECC`(k) where k = –`MAXLAG`, …, –1, 0, 1, …, `MAXLAG`.

FORTRAN 90 Interface

Generic: `CALL CCF (X, Y, MAXLAG, CC[,…])`

Specific: The specific interface names are `S_CCF` and `D_CCF`.

FORTRAN 77 Interface

Single:
```
CALL CCF (NOBS, X, Y, MAXLAG, IPRINT, ISEOPT, IMEAN, XMEAN,
YMEAN, XVAR, YVAR, CCV, CC, SECC)
```

Double: The double precision name is `DCCF`.

Example

Consider the Gas Furnace Data (Box and Jenkins 1976, pages 532–533) where X is the input gas rate in cubic feet/minute and Y is the percent CO_2 in the outlet gas. Routine CCF is used to computed the cross-covariances and cross-correlations between time series X and Y with lags from –MAXLAG = –10 through lag MAXLAG = 10. In addition, the estimated standard errors of the estimated cross-correlations are computed. In the first invocation with ISEOPT = 1, the standard errors are based on the assumption that autocorrelations and cross-correlations for lags greater than MAXLAG or less than –MAXLAG are zero. In the second invocation with ISEOPT = 1, the standard errors are based on the additional assumption that all cross-correlations for X and Y are zero.

```
!
      USE GDATA_INT
      USE CCF_INT

      INTEGER    IPRINT, MAXLAG, NOBS
      PARAMETER  (IPRINT=3, MAXLAG=10, NOBS=296)
!
      INTEGER    IMEAN, ISEOPT, NCOL, NROW
      REAL       CC(-MAXLAG:MAXLAG), CCV(-MAXLAG:MAXLAG), &
                 RDATA(296,2), SECC(-MAXLAG:MAXLAG), X(NOBS), XMEAN, &
                 XVAR, Y(NOBS), YMEAN, YVAR
!
      EQUIVALENCE (X(1), RDATA(1,1)), (Y(1), RDATA(1,2))
!
      CALL GDATA (7, RDATA, NROW, NCOL)
!                                USE Default Option to estimate means.

!                                Bartlett's formula (general case)
      ISEOPT = 1
!                                Compute cross correlation function
      CALL CCF (X, Y, MAXLAG, CC, IPRINT=IPRINT, ISEOPT=ISEOPT)
!                                Bartlett's formula (independent case)
      ISEOPT = 2
!                                Compute cross correlation function
      CALL CCF (X, Y, MAXLAG, CC, IPRINT=IPRINT, ISEOPT=ISEOPT)
!
      END
```

Output

```
Output from CCF/C2F

Mean of series X     =  -0.056834
Variance of series X =     1.1469

Mean of series Y     =     53.509
Variance of series Y =     10.219

Lag          CCV           CC         SECC

-10      -0.40450     -0.11815     0.158148
 -9      -0.50849     -0.14853     0.155750
 -8      -0.61437     -0.17946     0.152735
```

```
 -7      -0.70548     -0.20607     0.149087
 -6      -0.77617     -0.22672     0.145055
 -5      -0.83147     -0.24287     0.141300
 -4      -0.89132     -0.26035     0.138421
 -3      -0.98060     -0.28643     0.136074
 -2      -1.12477     -0.32854     0.132159
 -1      -1.34704     -0.39347     0.123531
  0      -1.65853     -0.48445     0.107879
  1      -2.04865     -0.59841     0.087341
  2      -2.48217     -0.72503     0.064141
  3      -2.88541     -0.84282     0.046946
  4      -3.16536     -0.92459     0.044097
  5      -3.25344     -0.95032     0.048234
  6      -3.13113     -0.91459     0.049155
  7      -2.83919     -0.82932     0.047562
  8      -2.45302     -0.71652     0.053478
  9      -2.05269     -0.59958     0.071566
 10      -1.69466     -0.49500     0.093933

Output from CCF/C2F

Mean of series X     =  -0.056834
Variance of series X =     1.1469

Mean of series Y     =     53.509
Variance of series Y =     10.219

Lag          CCV           CC         SECC

-10      -0.40450     -0.11815      0.16275
 -9      -0.50849     -0.14853      0.16247
 -8      -0.61437     -0.17946      0.16219
 -7      -0.70548     -0.20607      0.16191
 -6      -0.77617     -0.22672      0.16163
 -5      -0.83147     -0.24287      0.16135
 -4      -0.89132     -0.26035      0.16107
 -3      -0.98060     -0.28643      0.16080
 -2      -1.12477     -0.32854      0.16052
 -1      -1.34704     -0.39347      0.16025
  0      -1.65853     -0.48445      0.15998
  1      -2.04865     -0.59841      0.16025
  2      -2.48217     -0.72503      0.16052
  3      -2.88541     -0.84282      0.16080
  4      -3.16536     -0.92459      0.16107
  5      -3.25344     -0.95032      0.16135
  6      -3.13113     -0.91459      0.16163
  7      -2.83919     -0.82932      0.16191
  8      -2.45302     -0.71652      0.16219
  9      -2.05269     -0.59958      0.16247
 10      -1.69466     -0.49500      0.16275
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `C2F/DC2F`. The reference is:

```
CALL C2F (NOBS, X, Y, MAXLAG, IPRINT, ISEOPT, IMEAN,  XMEAN,
YMEAN, XVAR, YVAR, CCV, CC, SECC,  ACX, ACY)
```

The additional arguments are as follows:

ACX — Work vector of length equal to MAXLAG + 1.

ACY — Work vector of length equal to MAXLAG + 1.

2. If ISEOPT = 0, then no workspace is needed and SECC, ACX, and ACY can be dimensioned with length 1.

3. Autocovariances, autocorrelations, and standard errors of autocorrelations may be obtained by setting the first and second time series equal.

Description

Routine CCF estimates the cross-correlation function of two jointly stationary time series given a sample of n = NOBS observations $\{X_t\}$ and $\{Y_t\}$ for $t = 1, 2, \ldots, n$.

Let

$$\hat{\mu}_x = \text{XMEAN}$$

be the estimate of the mean μ_X of the time series $\{X_t\}$ where

$$\hat{\mu}_X = \begin{cases} \mu_X & \mu_X \text{ known} \\ \dfrac{1}{n}\sum_{t=1}^{n} X_t & \mu_X \text{ unknown} \end{cases}$$

The autocovariance function of $\{X_t\}$, $\sigma_X(k)$, is estimated by

$$\hat{\sigma}_X(k) = \frac{1}{n}\sum_{t=1}^{n-k}(X_t - \hat{\mu}_X)(X_{t+k} - \hat{\mu}_X), \quad k = 0, 1, \ldots, K$$

where K = MAXLAG. Note that

$$\hat{\sigma}_X(0)$$

is equivalent to the sample variance XVAR. The autocorrelation function $\rho_X(k)$ is estimated by

$$\hat{\rho}_X(k) = \frac{\hat{\sigma}_X(k)}{\hat{\sigma}_X(0)} \quad k = 0,1, \ldots, K$$

Note that

$$\hat{\rho}_X(0) \equiv 1$$

by definition. Let

$$\hat{\mu}_Y = \text{YMEAN}, \hat{\sigma}_Y(k), \text{and } \hat{\rho}_Y(k)$$

be similarly defined.

The cross-covariance function $\sigma_{XY}(k)$ is estimated by

$$\hat{\sigma}_{XY}(k) = \begin{cases} \frac{1}{n}\sum_{t=1}^{n-k}(X_t - \hat{\mu}_X)(Y_{t+k} - \hat{\mu}_Y) & k = 0,1,\ldots,K \\ \frac{1}{n}\sum_{t=1-k}^{n}(X_t - \hat{\mu}_X)(Y_{t+k} - \hat{\mu}_Y) & k = -1,-2,\ldots,-K \end{cases}$$

The cross-correlation function $\rho_{XY}(k)$ is estimated by

$$\hat{\rho}_{XY}(k) = \frac{\hat{\sigma}_{XY}(k)}{\left[\hat{\sigma}_X(0)\hat{\sigma}_Y(0)\right]^{1/2}} \quad k = 0,\pm 1,\ldots,\pm K$$

The standard errors of the sample cross-correlations may be optionally computed according to argument ISEOPT. One method is based on a general asymptotic expression for the variance of the sample cross-correlation coefficient of two jointly stationary time series with independent, identically distributed normal errors given by Bartlett (1978, page 352). The theoretical formula is

$$\begin{aligned} \mathrm{var}\left\{\hat{\rho}_{XY}(k)\right\} &= \frac{1}{n-k}\sum_{i=-\infty}^{\infty}\left[\rho_X(i)\rho_Y(i) + \rho_{XY}(i-k)\rho_{XY}(i+k)\right. \\ &\quad -2\rho_{XY}(k)\left\{\rho_X(i)\rho_{XY}(i+k) + \rho_{XY}(-i)\rho_Y(i+k)\right\} \\ &\quad \left.+\rho_{XY}^2(k)\left\{\rho_X(i) + \frac{1}{2}\rho_X^2(i) + \frac{1}{2}\rho_Y^2(i)\right\}\right] \end{aligned}$$

For computational purposes, the autocorrelations $\rho_X(k)$ and $\rho_Y(k)$ and the cross-correlations $\rho_{XY}(k)$ are replaced by their corresponding estimates for $|k| \le K$, and the limits of summation are equal to zero for all k such that $|k| > K$.

A second method evaluates Bartlett's formula under the additional assumption that the two series have no cross-correlation. The theoretical formula is

$$\mathrm{var}\left\{\hat{\rho}_{XY}(k)\right\} = \frac{1}{n-k}\sum_{i=-\infty}^{\infty}\rho_X(i)\rho_Y(i) \quad k \ge 0$$

For additional special cases of Bartlett's formula, see Box and Jenkins (1976, page 377).

An important property of the cross-covariance coefficient is $\sigma_{XY}(k) = \sigma_{YX}(-k)$ for $k \ge 0$. This result is used in the computation of the standard error of the sample crosscorrelation for lag $k < 0$. In general, the cross-covariance function is not symmetric about zero so both positive and negative lags are of interest.

MCCF

Computes the multichannel cross-correlation function of two mutually stationary multichannel time series.

Required Arguments

X — NOBSX by NCHANX matrix containing the first time series. (Input)
Each row of X corresponds to an observation of a multivariate time series and each column of X corresponds to a univariate time series.

Y — NOBSY by NCHANY matrix containing the second time series. (Input)
Each row of Y corresponds to an observation of a multivariate time series and each column of Y corresponds to a univariate time series.

MAXLAG — Maximum lag of cross-covariances and cross-correlations to be computed. (Input)
MAXLAG must be greater than or equal to one and less than the minimum of NOBSX and NOBSY.

CC — Array of size NCHANX by NCHANY by 2 * MAXLAG + 1 containing the cross-correlations between the channels of X and Y. (Output)
The cross-correlation between channel *i* of the X series and channel *j* of the Y series at lag *k* corresponds to CC(*i*, *j*, *k*) where *i* = 1, …, NCHANX, *j* = 1, …, NCHANY, and *k* = –MAXLAG, …, –1, 0, 1, …, MAXLAG.

Optional Arguments

NOBSX — Number of observations in each channel of the first time series X. (Input)
NOBSX must be greater than or equal to two.
Default: NOBSX = size (X,1).

NCHANX — Number of channels in the first time series X. (Input)
NCHANX must be greater than or equal to one.
Default: NCHANX = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement of the calling program. (Input)
LDX must be greater than or equal to NOBSX.
Default: LDX = size (X,1).

NOBSY — Number of observations in each channel of the second time series Y. (Input)
NOBSY must be greater than or equal to two.
Default: NOBSY = size (Y,1).

NCHANY — Number of channels in the second time series Y. (Input)
NCHANY must be greater than or equal to one.
Default: NCHANY = size (Y,2).

LDY — Leading dimension of Y exactly as specified in the dimension statement of the calling program. (Input)
LDY must be greater than or equal to NOBSY.
Default: LDY = size (Y,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the means and variances.
2	Prints the means, variances, and cross-covariances.
3	Prints the means, variances, cross-covariances, and cross-correlations.

IMEAN — Option for computing the means. (Input)
Default: IMEAN = 1.

IMEAN	Action
0	XMEAN and YMEAN are user-specified.
1	XMEAN and YMEAN are set to the arithmetic means of their respective channels.

XMEAN — Vector of length NCHANX containing the means of the channels of X. (Input, if IMEAN = 0; output, if IMEAN = 1)

YMEAN — Vector of length NCHANY containing the means of the channels of Y. (Input, if IMEAN = 0; output, if IMEAN = 1)

XVAR — Vector of length NCHANX containing the variances of the channels of X. (Output)

YVAR — Vector of length NCHANY containing the variances of the channels of Y. (Output)

CCV — Array of size NCHANX by NCHANY by 2 * MAXLAG + 1 containing the cross-covariances between the channels of X and Y. (Output)
The cross-covariance between channel i of the X series and channel j of the Y series at lag k corresponds to CCV(i, j, k) where i = 1, …, NCHANX, j = 1, …, NCHANY, and k = –MAXLAG, …, –1, 0, 1, …, MAXLAG.

LDCCV — Leading dimension of CCV exactly as specified in the dimension statement in the calling program. (Input)
LDCCV must be greater than or equal to NCHANX.
Default: LDCCV = size (CCV,1).

MDCCV — Middle dimension of CCV exactly as specified in the dimension statement in the calling program. (Input)
MDCCV must be greater than or equal to NCHANY.
Default: MDCCV = size (CCV,2).

LDCC — Leading dimension of CC exactly as specified in the dimension statement in the calling program. (Input)
LDCC must be greater than or equal to NCHANX.
Default: LDCCV = size (CC,1).

MDCC — Middle dimension of CC exactly as specified in the dimension statement in the calling program. (Input)
MDCC must be greater than or equal to NCHANY.
Default: MDCCV = size (CC,2).

FORTRAN 90 Interface

Generic: `CALL MCCF (X, Y, MAXLAG, CC [,…])`

Specific: The specific interface names are S_MCCF and D_MCCF.

FORTRAN 77 Interface

Single:
```
CALL MCCF (NOBSX, NCHANX, X, LDX, NOBSY, NCHANY, Y, LDY,
MAXLAG, IPRINT, IMEAN, XMEAN, YMEAN, XVAR, YVAR, CCV,
LDCCV, MDCCV, CC, LDCC, MDCC)
```

Double: The double precision name is DMCCF.

Example

Consider the Wolfer Sunspot Data (Y) (Box and Jenkins 1976, page 530) along with data on northern light activity (X_1) and earthquake activity (X_2) (Robinson 1967, page 204) to be a three-channel time series. Routine MCCF is used to computed the cross-covariances and cross-correlations between X_1 and Y and between X_2 and Y with lags from $-$MAXLAG = -10 through lag MAXLAG = 10:

```
      USE GDATA_INT
      USE MCCF_INT
      INTEGER    IPRINT, LDCC, LDCCV, LDX, LDY, MAXLAG, MDCC, MDCCV, &
                 NCHANX, NCHANY, NOBSX, NOBSY
      PARAMETER  (IPRINT=3, MAXLAG=10, NCHANX=2, NCHANY=1, NOBSX=100, &
                 NOBSY=100, LDCC=NCHANX, LDCCV=NCHANX, LDX=NOBSX, &
                 LDY=NOBSY, MDCC=NCHANY, MDCCV=NCHANY)
!
      INTEGER    IMEAN, NCOL, NROW
      REAL       CC(LDCC,MDCC,-MAXLAG:MAXLAG), CCV(LDCCV,MDCCV,- &
                 MAXLAG:MAXLAG), RDATA(100,4), X(LDX,NCHANX), &
                 XMEAN(NCHANX), XVAR(NCHANX), Y(LDY,NCHANY), &
                 YMEAN(NCHANY), YVAR(NCHANY)
!
      EQUIVALENCE (X(1,1), RDATA(1,3)), (X(1,2), RDATA(1,4))
      EQUIVALENCE (Y(1,1), RDATA(1,2))
!
      CALL GDATA (8, RDATA, NROW, NCOL)
```

```
!                                  USE Default Option to estimate
!                                  channel means
!                                  Compute multichannel CCVF and CCF
      CALL MCCF (X, Y, MAXLAG, CC, IPRINT=IPRINT)
!
      END
```

Output

```
Channel means of X from MCCF
             1       2
         63.43   97.97

Channel variances of X
           1        2
     2643.7   1978.4

Channel means of Y from MCCF
             46.94

Channel variances of Y
          1383.8

Multichannel cross-covariance between X and Y from MCCF

Lag K =     -10
    1  -20.51
    2   70.71

Lag K =      -9
    1   65.02
    2   38.14

Lag K =      -8
    1   216.6
    2   135.6

Lag K =      -7
    1   246.8
    2   100.4

Lag K =      -6
    1   142.1
    2    45.0

Lag K =      -5
    1   50.70
    2  -11.81

Lag K =      -4
    1   72.68
    2   32.69

Lag K =      -3
    1   217.9
    2   -40.1
```

```
Lag K =      -2
    1   355.8
    2  -152.6

Lag K =      -1
    1   579.7
    2  -213.0

Lag K =       0
    1   821.6
    2  -104.8

Lag K =       1
    1   810.1
    2    55.2

Lag K =       2
    1   628.4
    2    84.8

Lag K =       3
    1   438.3
    2    76.0

Lag K =       4
    1   238.8
    2   200.4

Lag K =       5
    1   143.6
    2   283.0

Lag K =       6
    1   253.0
    2   234.4

Lag K =       7
    1   479.5
    2   223.0

Lag K =       8
    1   724.9
    2   124.5

Lag K =       9
    1   925.0
    2   -79.5

Lag K =      10
    1   922.8
    2  -279.3

Multichannel cross-correlation between X and Y from MCCF
```

```
Lag K =     -10
   1  -0.01072
   2   0.04274

Lag K =      -9
   1   0.03400
   2   0.02305

Lag K =      -8
   1   0.1133
   2   0.0819

Lag K =      -7
   1   0.1290
   2   0.0607

Lag K =      -6
   1   0.07431
   2   0.02718

Lag K =      -5
   1   0.02651
   2  -0.00714

Lag K =      -4
   1   0.03800
   2   0.01976

Lag K =      -3
  1   0.1139
  2  -0.0242

Lag K =      -2
  1   0.1860
  2  -0.0923

Lag K =      -1
  1   0.3031
  2  -0.1287

Lag K =       0
  1   0.4296
  2  -0.0633

Lag K =       1
  1   0.4236
  2   0.0333

Lag K =       2
  1   0.3285
  2   0.0512

Lag K =       3
  1   0.2291
  2   0.0459
```

```
Lag K =       4
  1   0.1248
  2   0.1211

Lag K =       5
  1   0.0751
  2   0.1710

Lag K =       6
  1   0.1323
  2   0.1417

Lag K =       7
  1   0.2507
  2   0.1348

Lag K =       8
  1   0.3790
  2   0.0752

Lag K =       9
  1   0.4836
  2  -0.0481

Lag K =      10
  1   0.4825
  2  -0.1688
```

Comments

1. For a given lag k, the multichannel cross-covariance coefficient is defined as the array of dimension NCHANX by NCHANY whose components are the single-channel cross-covariance coefficients CCV(i, j, k). A similar definition holds for the multichannel cross-correlation coefficient.

2. Multichannel autocovariances and autocorrelations may be obtained by setting the first and second time series equal.

Description

Routine MCCF estimates the multichannel cross-correlation function of two mutually stationary multichannel time series. Define the multichannel time series X by

$$X = (X_1, X_2, \ldots, X_p)$$

where

$$X_j = (X_{1j}, X_{2j}, \ldots, X_{nj})^T, \qquad j = 1, 2, \ldots, p$$

with n = NOBSX and p = NCHANX. Similarly, define the multichannel time series Y by

$$Y = (Y_1, Y_2, \ldots, Y_q)$$

where

$$Y_j = (Y_{1j}, Y_{2j}, \ldots, Y_{mj})^T, \qquad j = 1, 2, \ldots, q$$

with m = NOBSY and q = NCHANY. The columns of X and Y correspond to individual channels of multichannel time series and may be examined from a univariate perspective. The rows of X and Y correspond to observations of p-variate and q-variate time series, respectively, and may be examined from a multivariate perspective. Note that an alternative characterization of a multivariate time series X considers the columns to be observations of the multivariate time series while the rows contain univariate time series. For example, see Priestley (1981, page 692) and Fuller (1976, page 14).

Let

$$\hat{\mu}_X = \text{XMEAN}$$

be the row vector containing the means of the channels of X. In particular,

$$\hat{\mu}_X = \left(\hat{\mu}_{X_1}, \hat{\mu}_{X_2}, \ldots, \hat{\mu}_{X_p}\right)$$

where for $j = 1, 2, \ldots, p$

$$\hat{\mu}_{X_j} = \begin{cases} \mu_{X_j} & \mu_{X_j} \text{ known} \\ \dfrac{1}{n}\displaystyle\sum_{t=1}^{n} X_{tj} & \mu_{X_j} \text{ unknown} \end{cases}$$

Let

$$\hat{\mu}_Y = \text{YMEAN}$$

be similarly defined. The cross-covariance of lag k between channel i of X and channel j of Y is estimated by

$$\hat{\sigma}_{X_i Y_j}(k) = \begin{cases} \dfrac{1}{N}\displaystyle\sum_t (X_{ti} - \hat{\mu}_{X_i})(Y_{t+k,j} - \hat{\mu}_{Y_j}) & k = 0, 1, \ldots, K \\ \dfrac{1}{N}\displaystyle\sum_t (X_{ti} - \hat{\mu}_{X_i})(Y_{t+k,j} - \hat{\mu}_{Y_j}) & k = -1, -2, \ldots, -K \end{cases}$$

where $i = 1, \ldots, p, j = 1, \ldots, q$, and K = MAXLAG. The summation on t extends over all possible cross-products with N equal to the number of cross-products in the sum.

Let

$$\hat{\sigma}_X(0) = \text{XVAR}$$

be the row vector consisting of the estimated variances of the channels of X. In particular,

$$\hat{\sigma}_X(0) = (\hat{\sigma}_{X_1}(0), \hat{\sigma}_{X_2}(0), \ldots, \hat{\sigma}_{X_p}(0))$$

where

$$\hat{\sigma}_{X_j}(0) = \frac{1}{n}\sum_{t=1}^{n} X_{tj} - \hat{\mu}_{X_j})^2 \quad j = 1, 2, \ldots, p$$

Let

$$\hat{\sigma}_Y(0) = \text{YVAR}$$

be similarly defined. The cross-correlation of lag k between channel i of X and channel j of Y is estimated by

$$\hat{\rho}_{X_i Y_j}(k) = \frac{\hat{\sigma}_{X_i Y_j(k)}}{\left[\hat{\sigma}_{X_i}(0)\hat{\sigma}_{Y_j}(0)\right]^{1/2}} \quad k = 0, \pm 1, \ldots, \pm K$$

ARMME

Computes method of moments estimates of the autoregressive parameters of an ARMA model.

Required Arguments

MAXLAG — Maximum lag of the sample autocovariances of the time series *W*. (Input)
MAXLAG must be greater than or equal to NPAR + NPMA.

ACV — Vector of length MAXLAG + 1 containing the sample autocovariances of *W*. (Input)
The *k*-th sample autocovariance of *W* is denoted by ACV(*k*), *k* = 0, 1, …, MAXLAG.

NPMA — Number of moving average parameters. (Input)
NPMA must be greater than or equal to zero.

NPAR — Number of autoregressive parameters. (Input)
NPAR must be greater than or equal to one.

PAR — Vector of length NPAR containing the estimates of the autoregressive parameters. (Output)

Optional Arguments

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the estimates of the autoregressive parameters.

FORTRAN 90 Interface

Generic: `CALL ARMME (MAXLAG, ACV, NPMA, NPAR, PAR [,…])`

Specific: The specific interface names are `S_ARMME` and `D_ARMME`.

FORTRAN 77 Interface

Single: `CALL ARMME (MAXLAG, ACV, IPRINT, NPMA, NPAR, PAR)`

Double: The double precision name is `DARMME`.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Routine `ARMME` is invoked first to compute the method of moments estimates for the autoregressive parameters of an ARMA(2, 0) model given the sample autocovariances computed from routine `ACF` (page 697). Then, `ARMME` is invoked a second time to compute estimated autoregressive parameters for an ARMA(2, 1) model.

```
      USE UMACH_INT
      USE GDATA_INT
      USE ACF_INT
      USE ARMME_INT
      INTEGER    IMEAN, IPRINT, ISEOPT, MAXLAG, NOBS
      PARAMETER  (IMEAN=1, IPRINT=1, ISEOPT=0, MAXLAG=4, NOBS=100)
!
      INTEGER    NCOL, NOUT, NPAR, NPMA, NROW
      REAL       AC(0:MAXLAG), ACV(0:MAXLAG), PAR(2), RDATA(176,2), &
                 SEAC(1), W(100), WMEAN
!
      EQUIVALENCE (W(1), RDATA(22,2))
!
      CALL UMACH (2, NOUT)
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Compute sample ACV
      CALL ACF (W, MAXLAG, AC, ACV=ACV)
!                                Compute estimates of autoregressive
!                                parameters for ARMA(2,0) model
!                                (Box and Jenkins, page 83)
      WRITE (NOUT,*) 'ARMA(2,0) Model'
      NPAR = 2
      NPMA = 0
      CALL ARMME (MAXLAG, ACV, NPMA, NPAR, PAR, IPRINT=IPRINT)
!                                Compute estimates of autoregressive
!                                parameters for ARMA(2,1) model
      WRITE (NOUT,*) ' '
      WRITE (NOUT,*) 'ARMA(2,1) Model'
```

```
      NPMA = 1
      CALL ARMME (MAXLAG, ACV, NPMA, NPAR, PAR, IPRINT=IPRINT)
!
      END
```

Output

```
ARMA(2,0) Model

  Output PAR
    1       2
1.318  -0.635

ARMA(2,1) Model

  Output PAR
    1       2
1.244  -0.575
```

Comments

1. Workspace may be explicitly provided, if desired, by use of A2MME/DA2MME. The reference is:

   ```
   CALL A2MME (MAXLAG, ACV, IPRINT, NPMA, NPAR, PAR, A, FACT, IPVT,
   WK)
   ```

   The additional arguments are as follows:

   A — Work vector of length equal to NPAR^2.

   FACT — Work vector of length equal to NPAR^2.

   IPVT — Work vector of length equal to NPAR.

   WK — Work vector of length equal to NPAR.

2. Informational error

Type	Code	
4	1	The problem is ill-conditioned. Transformation of the data or increased precision in the calculations may be appropriate.

3. The sample autocovariance function may be obtained using the routine ACF (page 697).

4. The first element of ACV must be the sample variance of the time series.

Description

Routine ARMME determines the autoregressive parameters of an ARMA process using the extended Yule-Walker equations given the K = MAXLAG autocovariances $\sigma(k)$ for $k = 1, \ldots, K$.

Suppose the time series $\{W_t\}$ is generated by an ARMA(p, q) model

$$W_t = \theta_0 + \phi_1 W_{t-1} + \ldots + \phi_p W_{t-p} + A_t - \theta_1 A_{t-1} - \ldots - \theta_q A_{t-q}, \qquad t \in \{0, \pm 1, \pm 2, \ldots\}$$

where p = NPAR and q = NPMA. Since W_t depends only on the innovations A_t that have occurred up through time t, the p autoregressive parameters are related to the autocovariances of lags $k = q + 1, \ldots, q + p$ by the set of equations

$$\sigma(q+1) = \phi_1 \sigma(q) + \phi_2 \sigma(q-1) + \ldots + \phi_p \sigma(q-p+1)$$

$$\sigma(q+2) = \phi_1 \sigma(q+1) + \phi_2 \sigma(q) + \ldots + \phi_p \sigma(q-p+2)$$

$$\vdots$$

$$\sigma(q+p) = \phi_1 \sigma(q+p-1) + \phi_2 \sigma(q+p-2) + \ldots + \phi_p \sigma(q)$$

This general system of linear equations is called the extended Yule-Walker equations. For $q = 0$, the system is referred to as the Yule-Walker equations. The equivalent matrix version is given by

$$\Sigma\phi = \sigma$$

where

$$\phi = (\phi_1, \ldots, \phi_p)^T$$
$$\Sigma_{ij} = \sigma(|q + i - j|) \qquad i, j = 1, \ldots, p$$
$$\sigma_i = \sigma(q + i) \qquad i = 1, \ldots, p$$

The overall constant θ_0 is defined by

$$\theta_0 = \begin{cases} \mu & p = 0 \\ \mu\left(1 - \sum_{i=1}^{p} \phi_i\right) & p > 0 \end{cases}$$

where μ is the mean of W_t.

In practice, the autocovariance function is estimated by the sample autocovariances

$$\hat{\sigma}(k)$$

for $k = 1, \ldots, K$. The solution of the extended Yule-Walker equations using these sample moments yields the *method of moments* estimates of the autoregressive parameters. The overall constant may then be estimated given an estimate of μ. Note that the extended Yule-Walker equations may be analogously defined in terms of autocorrelations instead of autocovariances. See Box and Jenkins (1976, pages 189–191) for some comments concerning the initial estimation of autoregressive parameters using the Yule-Walker equations.

MAMME

Computes method of moments estimates of the moving average parameters of an ARMA model.

Required Arguments

MAXLAG — Maximum lag of the sample autocovariances of the time series *W*. (Input)
MAXLAG must be greater than or equal to NPAR + NPMA.

ACV — Vector of length MAXLAG + 1 containing the sample autocovariances of *W*. (Input)
The *k*-th sample autocovariance of *W* is denoted by ACV(*k*), *k* = 0, 1, …, MAXLAG.

PAR — Vector of length NPAR containing the estimates of the autoregressive parameters. (Input)

PMA — Vector of length NPMA containing the estimates of the moving average parameters. (Output)

Optional Arguments

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the estimates of the moving average parameters.

NPAR — Number of autoregressive parameters. (Input)
NPAR must be greater than or equal to zero.
Default: NPAR = size (PAR,1).

RELERR — Stopping criterion for use in the nonlinear equation solver. (Input)
If RELERR = 0.0, then the default value RELERR = 100.0 * AMACH(4) is used. See the documentation for routine AMACH (in the Reference Material section of this manual).
Default: RELERR = 0.0.

MAXIT — The maximum number of iterations allowed in the nonlinear equation solver. (Input)
If MAXIT = 0, then the default value MAXIT = 200 is used.
Default: MAXIT = 0.

NPMA — Number of moving average parameters. (Input)
NPMA must be greater than or equal to one.
Default: NPMA = size (PMA,1).

FORTRAN 90 Interface

Generic: `CALL MAMME (MAXLAG, ACV, PAR, PMA [,…])`

Specific: The specific interface names are S_MAMME and D_MAMME.

FORTRAN 77 Interface

Single: `CALL MAMME (MAXLAG, ACV, IPRINT, NPAR, PAR, RELERR, MAXIT, NPMA, PMA)`

Double: The double precision name is `DMAMME`.

Example

Consider the Wölfer Sunspot Data (Box and Jenkins 1976, page 530) consisting of the number of sunspots observed each year from 1770 through 1869. Routine `MAMME` is invoked to compute the method of moments estimates for the moving average parameter of an ARMA(2,1) model given the sample autocovariances computed from routine `ACF` (page 697) and given the estimated autoregressive parameters computed from routine `ARMME` (page 720).

```
      USE GDATA_INT
      USE ACF_INT
      USE ARMME_INT
      USE MAMME_INT
      INTEGER    IMEAN, IPRINT, ISEOPT, LDX, MAXLAG, NDX, NOBS, &
                 NOPRIN, NPAR, NPMA
      PARAMETER  (IMEAN=1, IPRINT=1, ISEOPT=0, LDX=176, MAXLAG=4, &
                 NDX=2, NOBS=100, NOPRIN=0, NPAR=2, NPMA=1)
!
      INTEGER    MAXIT, NCOL, NROW
      REAL       AC(0:MAXLAG), ACV(0:MAXLAG), PAR(2), PMA(1), &
                 RDATA(LDX,NDX), RELERR, SEAC(1), W(100), WMEAN
!
      EQUIVALENCE (W(1), RDATA(22,2))
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Compute sample ACV
      CALL ACF (W, MAXLAG, AC, ACV=ACV)
!                                Compute estimates of autoregressive
!                                parameters for ARMA(2,1) model
      CALL ARMME (MAXLAG, ACV, NPMA, NPAR, PAR)
!                                Convergence parameters
!                                Compute estimate of moving average
!                                parameter for ARMA(2,1) model
      CALL MAMME (MAXLAG, ACV, PAR, PMA, IPRINT=IPRINT)
!
      END
```

Output

```
Output PMA from MAMME/M2MME
-0.1241
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `M2MME/DM2MME`. The reference is:

```
CALL M2MME (MAXLAG, ACV, IPRINT, NPAR, PAR, RELERR, MAXIT, NPMA,
PMA, PARWK, ACVMOD, TAUINI, TAU, FVEC, FJAC, R, QTF, WKNLN)
```

The additional arguments are as follows:

PARWK — Work vector of length equal to NPAR + 1.

ACVMOD — Work vector of length equal to NPMA + 1.

TAUINI — Work vector of length equal to NPMA + 1.

TAU — Work vector of length equal to NPMA + 1.

FVEC — Work vector of length equal to NPMA + 1.

FJAC — Work vector of length equal to (NPMA + 1)^2.

R — Work vector of length equal to (NPMA + 1) * (NPMA + 2)/2.

QTF — Work vector of length equal to NPMA + 1.

WKNLN — Work vector of length equal to 5 * (NPMA + 1).

2. Informational error

Type	Code	
4	1	The nonlinear equation solver did not converge to RELERR within MAXIT iterations.

3. The sample autocovariance function may be computed using the routine ACF (page 697).

4. The autoregressive parameter estimates may be computed using the routine ARMME (page 720).

Description

Routine MAMME estimates the moving average parameters of an ARMA process based on a system of nonlinear equations given K = MAXLAG autocovariances $\sigma(k)$ for $k = 1, \ldots, K$ and p = NPAR autoregressive parameters ϕ_i for $i = 1, \ldots, p$.

Suppose the time series $\{W_t\}$ is generated by an ARMA(p,q) model

$$\phi(B)W_t = \theta_0 + \theta(B)A_t, \qquad t \in 2\ \{0, \pm1, \pm2, \ldots\}$$

where p = NPAR and q = NPMA Let

$$W_t^{'} = \phi(B)W_t$$

then the autocovariances of the *derived* moving average process $W_t = \theta(B)A_t$ are given by

$$\sigma'(k) = \begin{cases} \sigma(k) & p = 0 \\ \sum_{i=0}^{p} \sum_{j=0}^{p} \phi_i \phi_j \sigma\left(\left|k + i - j\right|\right) & p \geq 1, \phi_0 \equiv -1 \end{cases}$$

where $\sigma(k)$ denotes the autocovariance function of the original W_t process. The iterative procedure for determining the moving average parameters is based on the relation

$$\sigma'(k) = \begin{cases} (1 + \theta_1^2 + \cdots + \theta_q^2)\sigma_A^2 & k = 0 \\ (-\theta_k + \theta_1\theta_{k+1} + \cdots + \theta_{q-k}\theta_q)\sigma_A^2 & k \geq 1 \end{cases}$$

Let $\tau = (\tau_0, \tau_1, \ldots, \tau_q)^T$ and $f = (f_0, f_1, \ldots, f_q)^T$ where

$$\tau_j = \begin{cases} \sigma_A & j = 0 \\ -\theta_j / \tau_0 & j = 1, \ldots, q \end{cases}$$

and

$$f_j = \sum_{i=0}^{q-j} \tau_i \tau_{i+j} - \sigma'(j) \qquad j = 0, 1, \ldots, q$$

Then, the value of τ at the $(i + 1)$-th iteration is determined by

$$\tau^{i+1} = \tau^i - (T^i)^{-1} f^i$$

The estimation procedure begins with the initial value

$$\tau^0 = (\sqrt{\sigma'(0)}, 0, \ldots, 0)^T$$

and terminates at iteration i when either $\|f^i\|$ is less than `RELERR` or i equals `MAXIT`. The moving average parameters are determined from the final estimate of τ by setting $\theta_j = -\tau_j/\tau_0$ for $j = 1, \ldots, q$. The random shock variance is determined according to

$$\sigma_A^2 = \begin{cases} \sigma(0) - \sum_{i=1}^{p} \phi_i \sigma(i) & q = 0 \\ \tau_0^2 & q \geq 0 \end{cases}$$

In practice, both the autocovariances and the autoregressive parameters are estimated. The solution of the system of nonlinear equations using these sample moments yields the method of moments estimates of the moving average parameters and the random shock variance. Note that autocorrelations $\rho(k)$ may be used instead of autocovariances $\sigma(k)$ to compute $\sigma'(k)$ for $k = 1, \ldots, K$. See Box and Jenkins (1976, pages 203–204) for additional motivation concerning the initial estimation of moving average parameters using a Newton-Raphson algorithm.

NSPE

Computes preliminary estimates of the autoregressive and moving average parameters of an ARMA model.

Required Arguments

W — Vector of length NOBS containing the stationary time series. (Input)

CNST — Estimate of the overall constant. (Output)

PAR — Vector of length NPAR containing the autoregressive parameter estimates. (Output)

PMA — Vector of length NPMA containing the moving average parameter estimates. (Output)

AVAR — Estimate of the random shock variance. (Output)

Optional Arguments

NOBS — Number of observations in the stationary time series *W*. (Input)
NOBS must be greater than NPAR + NPMA + 1.
Default: NOBS = size (W,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the mean of the time series, the estimate of the overall constant, the estimates of the autoregressive parameters, the estimates of the moving average parameters, and the estimate of the random shock variance.

IMEAN — Option for centering the time series X. (Input)
Default: IMEAN = 1.

IMEAN	Action
0	WMEAN is user specified.
1	WMEAN is set to the arithmetic mean of X.

WMEAN — Constant used to center the time series X. (Input, if IMEAN = 0; output, if IMEAN = 1)
Default: WMEAN = 0.0.

NPAR — Number of autoregressive parameters. (Input)
NPAR must be greater than or equal to zero.
Default: NPAR = size (PAR,1).

NPMA — Number of moving average parameters. (Input)
NPMA must be greater than or equal to zero.
Default: NPMA = size (PMA,1).

RELERR — Stopping criterion for use in the nonlinear equation solver. (Input)
If RELERR = 0.0, then the default value RELERR = 100.0 * AMACH(4) is used. See the documentation for routine AMACH (in the Reference Material section of the manual).
Default: RELERR = 0.0.

MAXIT — The maximum number of iterations allowed in the nonlinear equation solver. (Input)
If MAXIT = 0, then the default value MAXIT = 200 is used.
Default: MAXIT = 0.

FORTRAN 90 Interface

Generic: `CALL NSPE (W, CNST, PAR, PMA, AVAR [,…])`

Specific: The specific interface names are S_NSPE and D_NSPE.

FORTRAN 77 Interface

Single:
```
CALL NSPE (NOBS, W, IPRINT, IMEAN, WMEAN, NPAR, NPMA,
RELERR, MAXIT, CNST, PAR, PMA, AVAR)
```

Double: The double precision name is DNSPE.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Routine NSPE (page 727) is used to compute preliminary estimates

$$\hat{\theta}_0 \text{ (output in CNST)}$$

$$\hat{\phi}_1, \phi_2, \text{(output in PAR)}$$

$$\hat{\theta}_1 \text{ (output in PMA)}$$

$$\hat{\sigma}_A^2 \text{ (output in AVAR)}$$

for the following ARMA (2, 1) model

$$w_t = \theta_0 + \phi_1 w_{t-1} + \phi_2 w_{t-2} - \theta_1 A_{t-1} + A_t$$

where the errors A_t are independently distributed each normal with mean zero and variance

$$\sigma_A^2$$

```
      USE GDATA_INT
      USE NSPE_INT
      INTEGER    IPRINT, LDX, NDX, NOBS, NOPRIN, NPAR, NPMA
      PARAMETER  (IPRINT=1, LDX=176, NDX=2, NOBS=100, NOPRIN=0, NPAR=2, &
                 NPMA=1)
!
      INTEGER    IMEAN, MAXIT, NCOL, NROW
      REAL       AVAR, CNST, PAR(NPAR), PMA(NPMA), RDATA(LDX,NDX), &
                 RELERR, W(NOBS), WMEAN
!
      EQUIVALENCE (W(1), RDATA(22,2))
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL )
!                               USE Default Convergence parameters
!                               Compute preliminary parameter
!                               estimates for ARMA(2,1) model
      CALL NSPE (W, CNST, PAR, PMA, AVAR, IPRINT=IPRINT)
!
      END
```

Output

```
Results from NSPE/N2PE

WMEAN =     46.9760
CONST =     15.5440
AVAR  =     287.242

      PAR
     1       2
 1.244  -0.575

  PMA
-0.1241
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `N2PE/DN2PE`. The reference is:

   The additional arguments are as follows:

   ACV — Work vector of length equal to `NPAR` + `NPMA` + 1.

   PARWK — Work vector of length equal to `NPAR` + 1.

   ACVMOD — Work vector of length equal to `NPMA` + 1.

   TAUINI — Work vector of length equal to `NPMA` + 1.

   TAU — Work vector of length equal to `NPMA` + 1.

FVEC — Work vector of length equal to NPMA + 1.

FJAC — Work vector of length equal to (NPMA + 1)^2.

R — Work vector of length equal to (NPMA + 1) * (NPMA + 2)/2.

QTF — Work vector of length equal to NPMA + 1.

WKNLN — Work vector of length equal to 5 * (NPMA + 1).

A — Work vector of length equal to NPAR^2.

FAC — Work vector of length equal to NPAR^2.

IPVT — Work vector of length equal to NPAR.

WKARMM — Work vector of length equal to NPAR.

2. Informational error

Type	Code	
4	1	The nonlinear equation solver did not converge to RELERR within MAXIT iterations.

3. The value of WMEAN is used in the computation of the sample autocovariances of W in the process of obtaining the preliminary autoregressive parameter estimates. Also, WMEAN is used to obtain the value of CNST.

Description

Routine NSPE computes preliminary estimates of the parameters of an ARMA process given a sample of n = NOBS observations $\{W_t\}$ for $t = 1, 2, \ldots, n$.

Suppose the time series $\{W_t\}$ is generated by an ARMA(p,q) model of the form

$$\phi(B)W_t = \theta_0 + \theta(B)A_t t \in \{0, \pm 1, \pm 2, \ldots\}$$

where B is the backward shift operator,

$$\phi(B) = 1 - \phi_1(B) - \phi_2(B)^2 - \ldots - \phi_p(B)^p$$

$$\theta(B) = 1 - \theta_1(B) - \theta_2(B)^2 - \ldots - \theta_q(B)^q$$

p = NPAR and q = NPMA. Let

$$\hat{\mu} = \text{WMEAN}$$

be the estimate of the mean of the time series $\{W_t\}$ where

$$\hat{\mu} = \begin{cases} \mu & \mu \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} W_t & \mu \text{ unknown} \end{cases}$$

The autocovariance function $\sigma(k)$ is estimated by

$$\hat{\sigma}(k) = \frac{1}{n}\sum_{t=1}^{n-k}(W_t - \hat{\mu})(W_{t+k} - \hat{\mu}) \quad k = 0,1,\ldots,K$$

where $K = p + q$. Note that

$$\hat{\sigma}(0)$$

is an estimate of the sample variance.

Given the sample autocovariances, the routine ARMME (page 720) is used to compute the method of moments estimates of the autoregressive parameters using the extended Yule-Walker equations

$$\hat{\Sigma}\hat{\phi} = \hat{\sigma}$$

where

$$\hat{\phi} = (\hat{\phi}_1, \ldots, \hat{\phi}_p)^T$$
$$\hat{\Sigma}_{ij} = \hat{\sigma}(|q + i - j|) \quad i, j = 1, \ldots, p$$
$$\hat{\sigma}_i = \hat{\sigma}(q + i) \quad i = 1, \ldots, p$$

The overall constant θ_0 is estimated by

$$\hat{\theta}_0 = \begin{cases} \hat{\mu} & p = 0 \\ \hat{\mu}(1 - \sum_{i=1}^{p}\hat{\phi}_i) & p > 0 \end{cases}$$

The moving average parameters are estimated using the routine MAMME (page 723). Let

$$W_t' = \phi(B)W_t$$

then the autocovariances of the *derived* moving average process

$$W_t' = \theta(B)A_t$$

are estimated by

$$\hat{\sigma}'(k) = \begin{cases} \hat{\sigma}(k) & p = 0 \\ \sum_{i=0}^{p}\sum_{j=0}^{p}\hat{\phi}_i\hat{\phi}_j\hat{\sigma}(|k + i - j|) & p \geq 1, \hat{\phi}_0 \equiv -1 \end{cases}$$

The iterative procedure for determining the moving average parameters is based on the relation

$$\sigma'(k) = \begin{cases} (1 + \theta_1^2 + \cdots + \theta_q^2)\sigma_A^2 & k = 0 \\ (-\theta_k + \theta_1\theta_{k+1} + \cdots + \theta_{q-k}\theta_q)\sigma_A^2 & k \geq 1 \end{cases}$$

where $\sigma(k)$ denotes the autocovariance function of the original W_t process.

Let $\tau = (\tau_0, \tau_1, \ldots, \tau_q)^T$ and $f = (f_0, f_1, \ldots, f_q)^T$ where

$$\tau_j = \begin{cases} \sigma_A & j = 0 \\ -\theta_j / \tau_0 & j = 1, \ldots, q \end{cases}$$

and

$$f_j = \sum_{i=0}^{q-j} \tau_i \tau_{i+j} - \hat{\sigma}'(j) \quad j = 0, 1, \ldots, q$$

Then, the value of τ at the $(i + 1)$-th iteration is determined by

$$\tau^{i+1} = \tau^i - (T^i)^{-1} f^i$$

The estimation procedure begins with the initial value

$$\tau^0 = (\sqrt{\hat{\sigma}'(0)}, 0, \ldots, 0)^T$$

and terminates at iteration i when either $\|f^i\|$ is less than RELERR or i equals MAXIT. The moving average parameter estimates are obtained from the final estimate of τ by setting

$$\hat{\theta}_j = -\tau_j / \tau_0 \quad \text{for } j = 1, \ldots, q$$

The random shock variance is estimated by

$$\hat{\sigma}_A^2 = \begin{cases} \hat{\sigma}(0) - \sum_{i=1}^{p} \hat{\phi}_i \hat{\sigma}(i) & q = 0 \\ \tau_0^2 & q \geq 0 \end{cases}$$

See Box and Jenkins (1976, pages 498–500) for a description of a similar routine.

NSLSE

Computes least-squares estimates of parameters for a nonseasonal ARMA model.

Required Arguments

W — Vector of length NOBS containing the stationary time series. (Input)

PAR — Vector of length NPAR containing the autoregressive parameters.(Input/ Output)

On input, PAR contains the preliminary estimate. On output, PAR contains the final estimate.

LAGAR — Vector of length NPAR containing the order of the autoregressive parameters. (Input)
The elements of LAGAR must be greater than or equal to one.

PMA — Vector of length NPMA containing the moving average parameters.(Input/Output)
On input, PMA contains the preliminary estimate. On output, PMA contains the final estimate.

LAGMA — Vector of length NPMA containing the order of the moving average parameters. (Input)
The elements of LAGMA must be greater than or equal to one.

MAXBC — Maximum length of backcasting. (Input)
MAXBC must be greater than or equal to zero.

CNST — Estimate of the overall constant. (Output)
For IMEAN = 0, CNST is set to zero. For IMEAN = 1, CNST = WMEAN * (1 − PAR(1) − PAR(2) − … − PAR(NPAR)).

COV — NP by NP variance-covariance matrix of the estimates of the parameters where NP = IMEAN + NPAR + NPMA. (Output)
The ordering of variables in COV is WMEAN (if defined), PAR, and PMA. NP must 1 or more.

AVAR — Estimate of the random shock variance. (Output)
AVAR = (A(1)^2 + … + A(NA)^2)/(NOBS − IMEAN − NPAR − NPMA).

Optional Arguments

NOBS — Number of observations in the stationary time series W. (Input)
NOBS must be greater than IARDEG + IMADEG where IARDEG = max(LAGAR(*i*)) and IMADEG = max(LAGMA(*j*)).
Default: NOBS = size (W,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the least-squares estimates of the parameters, their associated standard errors, and the residual sum of squares at the final iteration.
2	Prints the least-squares estimates of the parameters and the residual sum of squares at each iteration and at the final iteration. Print the standard errors of the parameters at the final iteration.

IMEAN — Option for centering the time series W. (Input)
Default: IMEAN = 0.

IMEAN	Action
0	W is not centered.
1	W is centered about WMEAN. Centering the time series W about WMEAN is equivalent to inclusion of the overall constant in the model.

WMEAN — Estimate of the mean of the time series W. (Input/Output, if IMEAN = 1; not used if IMEAN = 0)
For IMEAN = 1, on input, WMEAN contains the preliminary estimate, on output, WMEAN contains the final estimate.
Default: WMEAN = 0.0.

NPAR — Number of autoregressive parameters. (Input)
NPAR must be greater than or equal to zero.
Default: NPAR = size (PAR,1).

NPMA — Number of moving average parameters. (Input)
NPMA must be greater than or equal to zero.
Default: NPMA = size (PMA,1).

TOLBC — Tolerance level used to determine convergence of the backcast algorithm. (Input)
Backcasting terminates when the absolute value of a backcast is less than TOLBC. Typically, TOLBC is set to a fraction of WSTDEV where WSTDEV is an estimate of the standard deviation of the time series. If TOLBC = 0.0, then TOLBC = 0.01 * WSTDEV is used.
Default: TOLBC = 0.0.

TOLSS — Tolerance level used to determine convergence of the nonlinear least-squares algorithm. (Input)
Default: TOLSS = 0.0.
TOLSS represents the minimum relative decrease in sum of squares between two iterations required to determine convergence. Hence, TOLSS must be greater than or equal to zero and less than one where TOLSS = 0.0 specifies the default value is to be used. The default value is

$\max\{10^{-10}, \text{EPS}^{2/3}\}$ for single precision and
$\max\{10^{-20}, \text{EPS}^{2/3}\}$ for double precision

where EPS = AMACH(4). See the documentation for routine AMACH (see the Reference Material section of this manual).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

NA — Number of residuals computed (including backcasts). (Output)
If NB values of the time series are backcast, then NA = NOBS − IARDEG + NB.

A — Vector of length NOBS − IARDEG + MAXBC containing the residuals (including backcasts) at the final parameter estimate point in the first NA locations. (Output)

FORTRAN 90 Interface

Generic: `CALL NSLSE (W, PAR, LAGAR, PMA, LAGMA, MAXBC, CNST, COV, AVAR [,…])`

Specific: The specific interface names are S_NSLSE and D_NSLSE.

FORTRAN 77 Interface

Single:
```
CALL NSLSE (NOBS, W, IPRINT, IMEAN, WMEAN, NPAR, PAR,
LAGAR, NPMA, PMA, LAGMA, MAXBC, TOLBC, TOLSS, CNST, COV,
LDCOV, NA, A, AVAR)
```

Double: The double precision name is DNSLSE.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Routine NSPE (page 727) is first invoked to compute preliminary estimates for an ARMA(2, 1) model. Then, NSLSE is invoked with the preliminary estimates as input in order to compute the least-squares estimates

$$\hat{\theta}_0 \text{ (output in CNST)}$$

$$\hat{\phi}_1, \hat{\phi}_2, \text{(output in PAR)}$$

$$\hat{\theta}_1 \text{ (output in PMA)}$$

$$\hat{\sigma}_A^2 \text{ (output in AVAR)}$$

for the ARMA(2, 1) model

$$w_t = \theta_0 + \phi_1 w_{t-1} + \phi_2 w_{t-2} - \theta_1 A_{t-1} + A_t$$

where the errors A_t are independently distributed each normal with mean zero and variance

$$\sigma_A^2$$

Note at the end of the output a warning error appears. Most of the time this error message can be ignored. There are three general reasons this error can occur.

1. Convergence was declared using the criterion based on TOLSS, but the gradient of the residual sum of squares function was nonzero. This occurred in this example. Either the message can be ignored or TOLSS can be reduced to allow more iterations and a slightly more accurate solution.

2. Convergence is declared based on the fact that a very small step was taken, but the gradient of the residual sum of squares function was nonzero. The message can usually be ignored. However, sometimes the algorithm is making very slow progress and is not near a minimum.

3. Convergence is not declared after 100 iterations.

Examination of the history of iterations using IPRINT = 2 and trying a smaller value for TOLSS can help you determine what caused the error message.

```
      USE GDATA_INT
      USE NSPE_INT
      USE NSLSE_INT
      INTEGER    IARDEG, IMEAN, LDCOV, LDX, MAXBC, MDX, NOBS, NP, &
                 NPAR, NPMA
      PARAMETER  (IARDEG=2, IMEAN=1, LDX=176, MAXBC=10, MDX=2, &
                 NOBS=100, NPAR=2, NPMA=1, NP=NPAR+NPMA+IMEAN, &
                 LDCOV=NP)
!
      INTEGER    IPRINT, LAGAR(NPAR), LAGMA(NPMA), MAXIT, NA, NCOL, &
                 NROW
      REAL       A(NOBS-IARDEG+MAXBC), AVAR, CNST, COV(LDCOV,NP), &
                 PAR(NPAR), PMA(NPMA), RELERR, TOLBC, TOLSS, W(NOBS), &
                 WMEAN, X(LDX,MDX)
!
      EQUIVALENCE (W(1), X(22,2))
!
      DATA LAGAR/1, 2/, LAGMA/1/
!                                  Wolfer Sunspot Data for
!                                  years 1770 through 1869
      CALL GDATA (2, X, NROW, NCOL)

!                                 USE Default Convergence parameters
!                                 Compute preliminary parameter
!                                 estimates for ARMA(2,1) model
      IPRINT = 1
      CALL NSPE (W, CNST, PAR, PMA, AVAR, IPRINT=IPRINT, WMEAN=WMEAN)
!
      TOLBC  = 0.0
      TOLSS  = 0.125
      IPRINT = 2
!
      CALL NSLSE (W, PAR, LAGAR, PMA, LAGMA, MAXBC, CNST, COV, &
                 AVAR, IMEAN=IMEAN,  WMEAN=WMEAN, TOLSS=TOLSS, &
                 IPRINT=IPRINT)
!
      END
```

Output

```
Results from NSPE/N2PE

WMEAN =     46.9760
CONST =     15.5440
AVAR  =     287.242
```

```
     PAR
    1       2
1.244  -0.575

PMA
-0.1241
------------------------------------------------------------------------
Iteration       1

WMEAN =   52.638233185

     PAR
    1       2
1.264  -0.606

  PMA
-0.1731

Residual SS (including backcasts) =  23908.66210937500
Number of residuals               =     108
Number of backcasts               =      10
------------------------------------------------------------------------
Iteration       2

WMEAN =   54.756504059

     PAR
    1       2
1.360  -0.688

  PMA
-0.1411

Residual SS (including backcasts) =  23520.71484375000
Number of residuals               =     108
Number of backcasts               =      10
------------------------------------------------------------------------
Final Results, Iteration       3

Parameter         Estimate            Std. Error            t-ratio

WMEAN             53.9187279            5.5178852            9.7716293

                                PAR
1                  1.3925704            0.0960639           14.4962845
2                 -0.7329484            0.0866115           -8.4624796

                                PMA
1                 -0.1375125            0.1223797           -1.1236545

CNST =          18.3527489
AVAR  =         243.4830170

Residual SS (including backcasts) =      23374.3691406
```

```
Number of residuals               =        108

Residual SS (excluding backcasts) =      20931.7519531
Number of residuals               =         98

*** WARNING  ERROR 1 from NSLSE.  Least squares estimation of the parameters
***          has failed to converge.  Increase MAXBC and/or TOLBC and/or
***          TOLSS.  The estimates of the parameters at the last iteration
***          may be used as new starting values.
```

Comments

1. Workspace may be explicitly provided, if desired, by use of N2LSE/DN2LSE. The reference is:

   ```
   CALL N2LSE (NOBS, W, IPRINT, IMEAN, WMEAN, NPAR, PAR, LAGAR,
   NPMA, PMA, LAGMA, MAXBC, TOLBC, TOLSS, CNST, COV, LDCOV, NA, A,
   AVAR, XGUESS, XSCALE, FSCALE, X, FVEC, FJAC, LDFJAC, RWKUNL,
   IWKUNL, WKNSRE, AI, FCST)
   ```

   The additional arguments are as follows:

   XGUESS — Work vector of length NP.

   XSCALE — Work vector of length NP.

   FSCALE — Work vector of length M.

   X — Work vector of length NP.

   FVEC — Work vector of length M.

   FJAC — Work vector of length M * NP.

   LDFJAC — Integer scalar equal to M.

   RWKUNL — Work vector of length 10 * NP + 2 * M – 1.

   IWKUNL — Work vector of length NP.

   WKNSRE — Work vector of length NOBS + MAXBC.

   AI — Work vector of length IMADEG.

   FCST — Work vector of length MAXBC.

2 Informational error
Type Code

3	1	Least-squares estimation of the parameters has failed to converge. Increase MAXBC and/or TOLBC and/or TOLSS. The estimates of the parameters at the last iteration may be used as new starting values.

Description

Routine NSLSE computes least-squares estimates of parameters for a nonseasonal ARMA model given a sample of n = NOBS observations $\{W_t\}$ for $t = 1, 2, \ldots, n$.

Suppose the time series $\{W_t\}$ is generated by a nonseasonal ARMA model of the form

$$\phi(B)(W_t - \mu) = \theta(B)A_t \qquad t \in \{0, \pm 1, \pm 2, \ldots\}$$

where B is the backward shift operator, μ is the mean of W_t,

$$\phi(B) = 1 - \phi_1 B^{l_\phi(1)} - \phi_2 B^{l_\phi(2)} - \cdots - \phi_p B^{l_\phi(p)} \qquad p \geq 0$$
$$\theta(B) = 1 - \theta_1 B^{l_\theta(1)} - \theta_2 B^{l_\theta(2)} - \cdots - \theta_q B^{l_\theta(q)} \qquad q \geq 0$$

with p = NPAR and q = NPMA. Without loss of generality, we assume

$$1 \leq l_\phi(1) \leq l_\phi(2) \leq \cdots \leq l_\phi(p)$$
$$1 \leq l_\theta(1) \leq l_\theta(2) \leq \cdots \leq l_\theta(q)$$

so that the nonseasonal ARMA model is of order (p', q') where $p' = l_\phi(p)$ and $q' = l_\theta(q)$. Note that the usual hierarchal model assumes

$$l_\phi(i) = i \qquad 1 \leq i \leq p$$
$$l_\theta(j) = j \qquad 1 \leq j \leq q$$

Consider the sum of squares function

$$S_T(\mu, \phi, \theta) = \sum_{-T+1}^{n}$$

where

$$[A_t] = E[A_t \mid \mu, \phi, \theta, W]$$

and T is the *backward origin*. The random shocks $\{A_t\}$ are assumed to be independent and identically distributed

$$N(0, \sigma_A^2)$$

random variables. Hence, the *log-likelihood function* is given by

$$l(\mu, \phi, \theta, \sigma_A) = f(\mu, \phi, \theta) - n \ln \sigma_A - \frac{S_T(\mu, \phi, \theta)}{2\sigma_A^2}$$

where $f(,\mu, \phi, \theta)$ is a function of μ, ϕ, and θ.

For $T = 0$, the log-likelihood function is *conditional* on the past values of both W_t and A_t required to initialize the model. The method of selecting these initial values usually introduces

transient bias into the model (Box and Jenkins 1976, pages 210–211). For T = ∞, this dependency vanishes, and the estimation problem concerns maximization of the *unconditional* log-likelihood function. Box and Jenkins (1976, page 213) argue that

$$S_{\infty}(\mu,\phi,\theta)/2\sigma_A^2$$

dominates

$$l(\mu,\phi,\theta,\sigma_A^2)$$

The parameter estimates that minimize the sum of squares function are called *least-squares estimates*. For large *n*, the unconditional least-squares estimates are approximately equal to the maximum likelihood estimates.

In practice, a finite value of *T* will enable sufficient approximation of the unconditional sum of squares function. The values of $[A_t]$ needed to compute the unconditional sum of squares are computed iteratively with initial values of W_t obtained by back-forecasting. The residuals (including backcasts), estimate of random shock variance, and covariance matrix of the final parameter estimates are also computed. Note that application of an appropriate transformation using routine BCTR (page 689) followed by differencing using routine DIFF (page 692) allows for fitting of nonseasonal ARIMA models. The algorithm for nonseasonal ARIMA models is developed in Chapter 7 of Box and Jenkins (1976). The extension to multiplicative seasonal ARIMA models is given in Box and Jenkins (1976, pages 500–504).

MAX_ARMA

Exact maximum likelihood estimation of the parameters in a univariate ARMA (auto-regressive, moving average) time series model.

Required Arguments

W — Vector of length NOBS containing the stationary time series. (Input)

PAR — Vector of length NPAR. On input PAR contains initial estimates for the autoregressive parameters. On output these are replaced by the exact maximum likelihood estimates for the autoregressive parameters. (Input/Output)

PMA — Vector of length NPMA. On input PMA contains initial estimates for the moving average parameters. On output these are replaced by the exact maximum likelihood estimates for the moving average parameters. (Input/Output)

Optional Arguments

NOBS — Number of values in the time series. (Input)
Default: NOBS = size(W,1).

NPAR — Number of autoregressive parameters. (Input)
Default: NPAR = size(PAR,1).

NPMA — Number of moving average parameters. (Input)
Default: NPMA = size(PMA,1).

IPRINT — Printing option. (Input)

0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: IPRINT = 0.

MAXIT — Maximum number of estimation iterations. (Input)
Default: MAXIT = 500.

AVAR — Estimate of the noise variance. (Output)

F — Value of –2*(ln(likelihood)) for fitted model. (Output)

FORTRAN 90 Interface

Generic: `CALL MAX_ARMA (W, PAR, PMA, [,…])`

Specific: The specific interface names are `S_MAX_ARMA` and `D_MAX_ARMA`.

Example

Consider the Wolfer Sunspot Data (Box and Jenkins, 1976, page 530) consisting of the number of sunspots observed each year from 1770 through 1869. In this example, `MAX_ARMA` is used to fit the following ARMA model:

$$w_t - \mu = \phi_1 w_{t-1} + \phi_2 w_{t-2} - \theta_1 a_{t-1} + a_t .$$

For these data, `MAX_ARMA` calculated the following estimates:

$$w_t - \mu = 1.22 w_{t-1} - 0.56 w_{t-2} + 0.38 a_{t-1} + a_t$$

Letting $\phi_0 = \mu\left(1 - \sum_{i=1}^{NPAR} \phi_i\right)$, we can obtain the following equivalent representations:

$$w_t = \phi_0 + \phi_1 w_{t-1} + \phi_2 w_{t-2} - \theta_1 a_{t-1} + a_t \text{ and,}$$
$$w_t = 0.33\mu + 1.22 w_{t-1} - 0.56 w_{t-2} + 0.38 a_{t-1} + a_t$$

```
      USE MAX_ARMA_INT
      USE GDATA_INT
      USE NSPE_INT
      IMPLICIT NONE
!                                     SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER    I, NOUT
      REAL(KIND(1E0)) PAR(2), PMA(1), AVAR, F
      REAL(KIND(1E0)) X(176,2)
      REAL(KIND(1E0)) CONST
      INTEGER         NCOL, NROW
!                                     Get Wolfer Sunspot Data
      CALL GDATA(2,X,NROW,NCOL)
!                                     Get preliminary PAR and PMA estimates
      CALL NSPE(X(22:,2),CONST, PAR, PMA, AVAR, NOBS=100)
!                                     TEST #1: DOCUMENT EXAMPLE
      CALL MAX_ARMA(x(22:,2), PAR, PMA, nobs=100, MAXIT=12000, &
                    AVAR=AVAR, F=F)
      WRITE(*,99994) SIZE(PAR)
      WRITE (NOUT,99996) (PAR(I),I=1,SIZE(PAR))
      WRITE(NOUT,99995) SIZE(PMA)
      WRITE(NOUT,99996) (PMA(I),I=1,SIZE(PMA))

      WRITE(NOUT,*) "-2*LN(MAXIMUM LOG LIKELIHOOD) = ", F
      WRITE(NOUT,*) "WHITE NOISE VARIANCE = ", AVAR
99994 FORMAT(//1H ,5('-'),2X,'FINAL PAR(I)',2X,'NPAR=',I3,2X,5('-'))
99995 FORMAT(//1H ,5('-'),2X,'FINAL PMA(I)',2X,'NPMA =',I3,2X,5('-'))
99996 FORMAT(1H ,5E20.10,/(1H ,5E20.10))

      END
```

Output

```
 -----  FINAL PAR(I)  NPAR=  2  -----
     0.1224243164E+01   -0.5600821972E+00


 -----  FINAL PMA(I)  NPMA =  1  -----
    -0.3847315013E+00
 -2*LN(MAXIMUM LOG LIKELIHOOD) =  539.5841
 WHITE NOISE VARIANCE =  214.50406
```

Comments

Informational error

Type	Code	
3	1	Input values for autoregressive coefficients are invalid. They do not represent a stationary time series. New values have been generated for PAR.
4	1	Maximum number of iterations exceeded. Try increasing MAXIT or using double precision routine.

4 2 Estimation process converged to a non-stationary solution.

Description

Routine MAX_ARMA is derived from the maximum likelihood estimation algorithm described by Akaike, Kitagawa, Arahata and Tada (1979), and the XSARMA routine published in the TIMSAC-78 Library.

Using the notation developed in the introduction to this chapter, the stationary time series W_t with mean μ can be represented by the nonseasonal autoregressive moving average (ARMA) model by the following relationship:

$$\phi(B)(W_t - \mu) = \theta(B)A_t$$

where

$$t \in ZZ = \{\cdots, -2, -1, 0, 1, 2, \cdots\},$$

B is the backward shift operator defined by

$$B^k W_t = W_{t-k},$$

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_{NPAR} B^{NPAR} \quad NPAR \geq 0,$$

and

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_{NPMA} B^{NPMA} \quad NPMA \geq 0.$$

MAX_ARMA estimates the coefficients

$$\phi_1, \phi_2, \cdots, \phi_{NPAR}$$

and

$$\theta_1, \theta_2, \cdots, \theta_{NPMA}$$

using maximum likelihood estimation.

MAX_ARMA checks the initial estimates for the autoregressive coefficients to ensure that they represent a stationary series. If

$$\phi_1, \phi_2, \cdots, \phi_{NPAR}$$

are the initial estimates for a stationary series then all (complex) roots of the following polynomial will fall outside the unit circle:

$$1 - \phi_1 z - \phi_2 z^2 - \cdots - \phi_{NPAR} z^{NPAR}$$

MAX_ARMA computes the roots of this polynomial for the initial estimates supplied in the vector PAR. If these estimates represent a non-stationary series, MAX_ARMA issues a warning message and replaces PAR with initial values that are stationary.

Initial estimates can be obtained using NSPE, (see page 727), and NSLSE, (see page 733), procedures for calculating the autoregressive and moving average parameters of a series.

MAX_ARMA also validates its final estimates to ensure that they too represent a stationary series. This is done to guard against the possibility that MAX_ARMA converged to a non-stationary solution. If non-stationary estimates are encountered, MAX_ARMA quits and issues a fatal message. Routines N1RTY and IERCD (see the Reference Material section of this manual) can be used to verify that the stationary condition was met.

For model selection, the ARMA model with the minimum value for AIC might be preferred

$$\text{AIC} = F+2p$$

where p = NPAR + MPMA.

GARCH

Computes estimates of the parameters of a GARCH(p,q) model.

Required Arguments

W — Vector of length NOBS containing the observed time series data. (Input)

NP — Number of GARCH parameters, p. (Input)

NQ — Number of ARCH parameters, q. (Input)

XGUESS — Vector of length NP+NQ +1 containing the initial values for the parameter vector X. (Input)

X — Vector of length NP+NQ+1 containing the estimates for σ^2, the GARCH parameters and the ARCH parameters. X(1) contains the estimate for σ^2, X(2)… X(NP+1) contain the GARCH estimates, X(NP+2)…. X(NP+NQ+1) contain the ARCH estimates. (Output)

Optional Arguments

SIG2MAX— Upperbound for σ^2 , the first element of X. (Input)
Default: SIG2MAX = 10.

NOBS — Length of the observed time series. (Input)
Default: NOBS = size(W).

A — Value of Log-likelihood function evaluated at X. (Output)

AIC — Akaike's Information Criterion evaluated at X. (Output)

VAR — (NP+NQ+1) by (NP+NQ+1) matrix containing the variance-covariance matrix. (Output)

NDIM — Column dimension (NP+NQ+1) of VAR. (Input) Default: NDIM = NP+NQ+1.

FORTRAN 90 Interface

Generic: CALL GARCH (W, NP, NQ, XGUESS, X[,…])

Specific: The specific interface names are S_GARCH and D_GARCH.

Example

The data for this example are generated to follow a GARCH(2,1) process by using a standard normal random number generation routine WG2RCH . The data set is analyzed and estimates of sigma, the GARCH parameters, and the ARCH parameters are returned. The values of the Log-likelihood function and Akaike's Information Criterion are returned from the optional arguments A and AIC.

```
USE GARCH_INT
USE RNSET_INT
IMPLICIT NONE

INTERFACE
SUBROUTINE WG2RCH (W, NP, NQ, NOBS, X, Z, Y0, SIGMA)
INTEGER    NP, NQ, NOBS
REAL(KIND(1D0))       W(:), X(:), Z(:), Y0(:), SIGMA(:)
END SUBROUTINE
END INTERFACE

INTEGER :: NP, NQ, NOBS, N
PARAMETER  (NP=2, NQ=1, NOBS=1000)
PARAMETER  (N=NP+NQ+1)
REAL(KIND(1D0)) :: A, AIC, Z(NOBS + 1000), Y0(NOBS + 1000), &
           SIGMA(NOBS + 1000), X0(N), X(N), XGUESS(N),  W(NOBS)
X0=(/1.3,0.2,0.3,0.4/)
XGUESS = (/1.0,0.1,0.2,0.3/)
CALL RNSET (182198625)
CALL WG2RCH (W, NP, NQ, NOBS, X0, Z, Y0, SIGMA)
CALL GARCH(W, NP, NQ, XGUESS, X, NOBS=NOBS, A=A, AIC=AIC)
WRITE(*,*)"Variance estimate is ", x(1)
WRITE(*,*)"GARCH(1) estimate is ", x(2)
WRITE(*,*)"GARCH(2) estimate is ", x(3)
WRITE(*,*)"ARCH(1) estimate is ", x(4)
WRITE(*,*)"Log-likelihood function is ", A
WRITE(*,*)"Akaike's Information Criterion is ", AIC
END

SUBROUTINE WG2RCH (W, NP, NQ, NOBS, X, Z, Y0, SIGMA)
USE RNNOR_INT
INTEGER    NP, NQ, NOBS
REAL(KIND(1D0))       W(:), X(:), Z(:), Y0(:), SIGMA(:)
INTEGER    I, J, L
REAL(KIND(1D0))       S1, S2, S3
```

```
!     RNNOR GENERATES STANDARD NORMAL OBSERVATIONS
      CALL RNNOR (NOBS+1000, Z)
!     INITIAL VALUES
      L = MAX(NP,NQ)
      L = MAX(L,1)
      DO I=1, L
         Y0(I) = Z(I)*X(1)
      END DO
!     COMPUTE THE INITIAL VALUE OF SIGMA
      S3 = 0.0;
      IF (MAX(NP,NQ) .GE. 1) THEN
         DO I=1, NP + NQ
            S3 = S3 + X(I+1)
         END DO
      END IF
      DO I=1, L
         SIGMA(I) = X(1)/(1.0-S3)
      END DO
      DO  I=L + 1, NOBS + 1000
         S1 = 0.0
         S2 = 0.0
         IF (NQ .GE. 1) THEN
            DO J=1, NQ
               S1 = S1 + X(J+1)*Y0(I-J)*Y0(I-J)
            END DO
         END IF
         IF (NP .GE. 1) THEN
            DO J=1, NP
               S2 = S2 + X(NQ+1+J)*SIGMA(I-J)
            END DO
         END IF
         SIGMA(I) = X(1) + S1 + S2
         Y0(I) = Z(I)*SQRT(SIGMA(I))
      END DO
! DISCARD THE FIRST 1000 SIMULATED OBSERVATIONS
      DO I=1, NOBS
         W(I) = Y0(1000+I)
      END DO
      RETURN
      END
```

Output

```
Variance estimate is  1.6915576416511892
GARCH(1) estimate is  0.24499571998823416
GARCH(2) estimate is  0.3372325349834042
ARCH(1) estimate is  0.3095905689822821
Log-likelihood function is  -2707.072433499691
Akaike's Information Criterion is  5422.144866999382
```

Description

The Generalized Autoregressive Conditional Heteroskedastic (GARCH) model for a time series $\{w_t\}$ is defined as

$$w_t = z_t \sigma_t$$
$$\sigma_t^2 = \sigma^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 + \sum_{i=1}^{q} \alpha_i w_{t-i}^2,$$

where z_t's are independent and identically distributed standard normal random variables,

$$0 < \sigma^2 < SIG2MAX, \ \beta_i \geq 0, \ \alpha_i \geq 0 \ \text{ and}$$
$$\sum_{i=2}^{p+q+1} x(i) = \sum_{i=1}^{p} \beta_i + \sum_{i=1}^{q} \alpha_i < 1.$$

The above model is denoted as GARCH(p,q). The β_i and α_i coeffecients will be referred to as GARCH and ARCH coefficents, respectively. When $\beta_i = 0$, $i = 1,2,\ldots,p$, the above model reduces to ARCH(q) which was proposed by Engle (1982). The nonnegativity conditions on the parameters imply a nonnegative variance and the condition on the sum of the β_i's and α_i's is required for wide sense stationarity.

In the empirical analysis of observed data, GARCH(1,1) or GARCH(1,2) models have often found to appropriately account for conditional heteroskedasticity (Palm 1996). This finding is similar to linear time series analysis based on ARMA models.

It is important to notice that for the above models positive and negative past values have a symmetric impact on the conditional variance. In practice, many series may have strong asymmetric influence on the conditional variance. To take into account this phenomena, Nelson (1991) put forward Exponential GARCH (EGARCH). Lai (1998) proposed and studied some properties of a general class of models that extended linear relationship of the conditional variance in ARCH and GARCH into nonlinear fashion.

The maximum likelihood method is used in estimating the parameters in GARCH(p,q). The log-likelihood of the model for the observed series $\{w_t\}$ with length m = `nobs` is

$$\log(L) = -\frac{m}{2}\log(2\pi) - \frac{1}{2}\sum_{t=1}^{m} y_t^2 / \sigma_t^2 - \frac{1}{2}\sum_{t=1}^{m} \log \sigma_t^2,$$
$$where \ \sigma_t^2 = \sigma^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 + \sum_{i=1}^{q} \alpha_i w_{t-i}^2.$$

Thus $\log(L)$ is maximized subject to the constraints on the α_i, β_i, and σ.

In this model, if $q = 0$, the GARCH model is singular since the estimated Hessian matrix is singular.

The initial values of the parameter vector `x` entered in vector `xguess` must satisfy certain constraints. The first element of `xguess` refers to σ^2 and must be greater than zero and less than `sig2max`. The remaining `p+q` initial values must each be greater than or equal to zero and sum to a value less than one.

To guarantee stationarity in model fitting,

$$\sum_{i=2}^{p+q+1} x(i) = \sum_{i=1}^{p} \beta_i + \sum_{i=1}^{q} \alpha_i < 1$$

is checked internally. The initial values should selected from values between zero and one.

AIC is computed by

$$-2 \log(L) + 2(p+q+1),$$

where $\log(L)$ is the value of the log-likelihood function.

In fitting the optimal model, the routine NNLPF as well as its associated subroutines are modified to find the maximum likelihood estimates of the parameters in the model. Statistical inferences can be performed outside the routine GARCH based on the output of the log-likelihood function (A), the Akaike Information Criterion (AIC), and the variance-covariance matrix (VAR).

SPWF

Computes the Wiener forecast operator for a stationary stochastic process.

Required Arguments

W — Vector of length NOBS containing the stationary time series. (Input)

WNADJ — White noise adjustment factor. (Input)
WNADJ must be greater than or equal to zero.

EPS — Bound on the normalized mean square error. (Input)
EPS must be in the range (0, 1) inclusive.

MLFOP — Maximum length of the forecast operator. (Input)
MLFOP must be greater than or equal to one and less than NOBS.

LFOP — Length of the estimated forecast operator. (Output)

FOP — Vector of length LFOP containing the estimated forecast operator coefficients. (Output)

Optional Arguments

NOBS — Number of observations in the stationary time series W. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (W,1).

IWMEAN — Option for estimation of the mean of W. (Input)
Default: IWMEAN = 1.

IWMEAN	Action

0 WMEAN is user specified.

1 WMEAN is set equal to the arithmetic mean of W.

WMEAN — Estimate of the mean of the time series W. (Input, if IWMEAN = 0; output, if IWMEAN = 1)
WMEAN is used to center the time series W prior to estimation of the forecast operator. Default: WMEAN = 0.0.

FORTRAN 90 Interface

Generic: `CALL SPWF (W, WNADJ, EPS, MLFOP, LFOP, FOP [,…])`

Specific: The specific interface names are S_SPWF and D_SPWF.

FORTRAN 77 Interface

Single: `CALL SPWF (NOBS, W, IWMEAN, WMEAN, WNADJ, EPS, MLFOP, LFOP, FOP)`

Double: The double precision name is DSPWF.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Application of routine SPWF to these data produces the following results:

```
      USE GDATA_INT
      USE SPWF_INT
      USE UMACH_INT
      INTEGER    MLFOP, NOBS
      PARAMETER  (MLFOP=1, NOBS=100)
!     INTEGER    I, IMEAN, LFOP, NCOL, NOUT, NROW

      REAL       EPS, FOP(MLFOP), RDATA(176,2), W(NOBS), WMEAN, WNADJ
!
      EQUIVALENCE (W(1), RDATA(22,2))
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Center on arithmetic mean
      IMEAN = 0
      WMEAN = 46.976
!                                White noise adjustment
      WNADJ = 0.0
!                                Bound on normalized MSE
      EPS = 0.1
!                                Determine autoregressive model
```

```
      CALL SPWF (W, WNADJ, EPS, MLFOP, LFOP, FOP, IWMEAN=IMEAN, &
                 WMEAN=WMEAN)
!                                  Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99997) LFOP
99997 FORMAT (/, 1X, 'Forecast operator length, LFOP = ', I2)
      WRITE (NOUT,99998)
99998 FORMAT (/, 1X, ' I           FOP(I)')
      DO 10  I=1, LFOP
         WRITE (NOUT,99999) I, FOP(I)
99999    FORMAT (1X, I2, 2X, F12.4)
   10 CONTINUE
!
      END
```

Output

```
*** WARNING  ERROR 5 from SPWF.  No operator could be found of length less
***          than or equal to 1 which produced a normalized mean square
***          error less than 1.000000E-01.

Forecast operator length, LFOP =  1

I           FOP(I)
1         0.8063
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `S2WF/DS2WF`. The reference is:

   ```
   CALL S2WF (NOBS, W, IWMEAN, WMEAN, WNADJ, EPS, MLFOP, LFOP, FOP,
   CW, WK)
   ```

   The additional arguments are as follows:

   CW — Vector of length `NOBS` containing the centered time series `W`. (Output)

   WK — Vector of length 2 * `MLFOP` + 1. (Output)

2. Informational error

Type	Code	
3	5	No operator could be found of length less than or equal to `MLFOP` that produced a normalized mean square error less than `EPS`.

3. The length of the forecast operator is determined by the arguments `EPS` and `MLFOP`. Iteration to a longer forecast operator stops when either the normalized mean square error is less than `EPS`, or the operator reaches the maximum allowable length, `MLFOP`.

4. The white noise adjustment factor, `WNADJ`, is used to modify the the estimate of the variance of the time series `W` used in the computation of the autocorrelation function of `W`. In the absence of white noise, `WNADJ` should be set to zero.

Description

Routine SPWF performs least-squares estimation of parameters for successive autoregressive models of a stationary stochastic process given a sample of n = NOBS observations $\{W_t\}$ for $t = 1, \ldots, n$.

Let

$$\hat{\mu} = \text{WMEAN}$$

be the estimate of the mean μ of the stochastic process $\{W_t\}$ where

$$\hat{\mu} = \begin{cases} \mu & \mu \text{ known} \\ \dfrac{1}{n}\sum_{t=1}^{n} W_t & \mu \text{ unknown} \end{cases}$$

Consider the autoregressive model of order k defined by

$$\phi_k(B)\tilde{W}_t = A_t \qquad k \geq 0$$

where

$$\tilde{W}_t = W_t - \hat{\mu}$$

and

$$\phi_k(B) = 1 - \phi_{1k}B - \phi_{2k}B^2 - \cdots - \phi_{kk}B^k \qquad k \geq 1$$

Successive AR(k) models are fit to the centered data using Durbin's algorithm (1960) based on the sample autocovariances

$$\hat{\sigma}(k) = \frac{1}{n}\sum_{t=1}^{n-k}(W_t - \hat{\mu})(W_{t+k} - \hat{\mu}) \qquad k \geq 0$$

Note that the variance

$$\hat{\sigma}^*(0)$$

used in the fitting algorithm is adjusted by the amount δ = WNADJ according to

$$\hat{\sigma}^*(0) = (1+\delta)\hat{\sigma}(0)$$

See Robinson (1967, page 96).

Iteration to the next higher order model terminates when either the expected mean square error of the model is less than EPS or when k = MLFOP. The forecast operator $\phi = (\phi_1, \phi_2, \ldots, \phi_{k^*})^T$ for k^* = LFOP is contained in FOP. See also Craddock (1969).

NSBJF

Computes Box-Jenkins forecasts and their associated probability limits for a nonseasonal ARMA model.

Required Arguments

W — Vector of length NOBS containing the time series. (Input)

PAR — Vector of length NPAR containing the autoregressive parameters. (Input)

LAGAR — Vector of length NPAR containing the order of the autoregressive parameters. (Input)
The elements of LAGAR must be greater than zero.

PMA — Vector of length NPMA containing the moving average parameters. (Input)

LAGMA — Vector of length NPMA containing the order of the moving average parameters. (Input)
The elements of LAGMA must be greater than zero.

ICNST — Option for including the overall constant in the model. (Input)

ICNST	Action
0	No overall constant is included.
1	The overall constant is included.

CNST — Estimate of the overall constant. (Input)

AVAR — Estimate of the random shock variance. (Input)
AVAR must be greater than 0.

ALPHA — Value in the exclusive interval (0, 1) used to specify the 100(1 − ALPHA)% probability limits of the forecasts. (Input)
Typical choices for ALPHA are 0.10, 0.05, and 0.01.

MXBKOR — Maximum backward origin. (Input)
MXBKOR must be greater than or equal to zero and less than or equal to NOBS − max(MAXAR, MAXMA) where MAXAR = max(LAGAR(*i*)) and MAXMA = max(LAGMA(*j*)). Forecasts at origins NOBS − MXBKOR through NOBS are generated.

MXLEAD — Maximum lead time for forecasts. (Input)
MXLEAD must be greater than zero.

FCST — MXLEAD by (MXBKOR + 3) matrix defined below. (Output)

Column	Content
j	Forecasts for lead times $l = 1, \ldots,$ MXLEAD at origins NOBS − MXBKOR − 1 + *j*, *j* = 1, …, MXBKOR + 1.
MXBKOR + 2	Deviations from each forecast that give the 100(1 − ALPHA)% probability limits.
MXBKOR + 3	Psi weights of the infinite order moving average form of the model.

Optional Arguments

NOBS — Number of observations in the time series W. (Input)
NOBS must be greater than ICONST + max(LAGAR(*i*)) + max(LAGMA(*j*)).
Default: NOBS = size (W,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the forecasts for lead times $l = 1, \ldots,$ MXLEAD at each origin $t =$ (NOBS − MXBKOR), …, NOBS, the 100(1 − ALPHA)% probability limit deviations, and the psi weights.

NPAR — Number of autoregressive parameters. (Input)
NPAR must be greater than or equal to zero.
Default: NPAR = size (PAR,1).

NPMA — Number of moving average parameters. (Input)
NPMA must be greater than or equal to zero.
Default: NPMA = size (PMA,1).

LDFCST — Leading dimension of FCST exactly as specified in the dimension statement in the calling program. (Input)
LDFCST must be greater than or equal to MXLEAD.
Default: LDFCST = size (FCST,1).

FORTRAN 90 Interface

Generic: `CALL NSBJF (W, PAR, LAGAR, PMA, LAGMA, ICNST, CNST, AVAR, ALPHA, MXBKOR, MXLEAD, FCST [,…])`

Specific: The specific interface names are S_NSBJF and D_NSBJF.

FORTRAN 77 Interface

Single: `CALL NSBJF (NOBS, W, IPRINT, NPAR, PAR, LAGAR, NPMA, PMA, LAGMA, ICNST, CNST, AVAR, ALPHA, MXBKOR, MXLEAD, FCST, LDFCST)`

Double: The double precision name is `DNSBJF`.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Routine `NSBJF` is used to computed forecasts and 95% probability limits for the forecasts for an `ARMA`(2, 1) model fit using routine `NSPE` (page 727). With `MXBKOR` = 3, columns one through four of `FCST` give forecasts given the data through 1866, 1867, 1868, and 1869, respectively. Column 5 gives the deviations from the forecast for computing probability limits, and column six gives the psi weights, which can be used to update forecasts once more data is available. For example, the forecast for the 102-nd observation (year 1871) given the data through the 100-th observation (year 1869) is 77.21, and 95% probability limits are given by 77.21 $\mp$ 56.30. After observation 101 (W_{101} for year 1870) is available, the forecast can be updated by using equation 7 with the psi weight (ψ_1 = 1.37) and the one-step-ahead forecast error for observation 101 ($W_{101} - 83.72$) to give

$$77.21 + 1.37\,(W_{101} - 83.72)$$

Since this updated forecast is one step ahead, the 95% probability limits are now given by the forecast $\mp$ 33.22.

```
      USE GDATA_INT
      USE NSPE_INT
      USE NSBJF_INT
      USE WRRRL_INT
      INTEGER    LDFCST, MXBKOR, MXLEAD, NOBS, NPAR, NPMA
      PARAMETER  (MXBKOR=3, MXLEAD=12, NOBS=100, NPAR=2, NPMA=1, &
                 LDFCST=MXLEAD)
!
      INTEGER    ICNST, LAGAR(NPAR), LAGMA(NPMA), NCOL, NROW
      REAL       ALPHA, AVAR, CNST, FCST(LDFCST,MXBKOR+3), PAR(NPAR), &
                 PMA(NPMA), RDATA(176,2), W(NOBS), WMEAN
      CHARACTER  CLABEL(MXBKOR+4)*40, RLABEL(1)*6
!
      EQUIVALENCE (W(1), RDATA(22,2))
!
      DATA LAGAR(1), LAGAR(2)/1, 2/
      DATA LAGMA(1)/1/
      DATA RLABEL/'NUMBER'/, CLABEL/'%/Lead%/Time', &
          '%/Forecast%/From 1866', '%/Forecast%/From 1867', &
          '%/Forecast%/From 1868', '%/Forecast%/From 1869', &
          ' Deviation %/  for 95%  %/Prob. Limits', '%/%/Psi'/
!                                 Wolfer Sunspot Data for
!                                 years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
```

```
!                                  Compute preliminary parameter
!                                  estimates for ARMA(2,1) model
      CALL NSPE (W, CNST, PAR, PMA, AVAR)
!
!                                  Include constant in forecast model
      ICONST = 1
!                                  Specify 95 percent probability
!                                  limits for forecasts
      ALPHA = 0.05
!                                  Compute forecasts
      CALL NSBJF (W, PAR, LAGAR, PMA, LAGMA, &
                  ICNST, CNST, AVAR, ALPHA, MXBKOR, MXLEAD, FCST)
!                                  Print results
      CALL WRRRL ('FCST', FCST, RLABEL, CLABEL, FMT='(5F9.2, F6.3)')
!
      END
```

Output

```
                                FCST
                                                        Deviation
Lead    Forecast    Forecast    Forecast    Forecast      for 95%
Time   From 1866   From 1867   From 1868   From 1869   Prob. Limits     Psi
 1       18.28       16.62       55.19       83.72          33.22    1.368
 2       28.92       32.02       62.76       77.21          56.30    1.127
 3       41.01       45.83       61.89       63.46          67.62    0.616
 4       49.94       54.15       56.46       50.10          70.64    0.118
 5       54.09       56.56       50.19       41.38          70.75   -0.208
 6       54.13       54.78       45.53       38.22          71.09   -0.326
 7       51.78       51.17       43.32       39.30          71.91   -0.286
 8       48.84       47.71       43.26       42.46          72.53   -0.169
 9       46.53       45.47       44.46       45.77          72.75   -0.045
10       45.35       44.69       45.98       48.08          72.77    0.041
11       45.21       44.99       47.18       49.04          72.78    0.077
12       45.71       45.82       47.81       48.91          72.82    0.072
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `N2BJF/DN2BJF`. The reference is:

   ```
   CALL N2BJF (NOBS, W, IPRINT, NPAR, PAR, LAGAR, NPMA, PMA, LAGMA,
   ICNST, CNST, AVAR, ALPHA, MXBKOR, MXLEAD, FCST, LDFCST, PARH,
   PMAH, PSIH, PSI, LAGPSI)
   ```

   The additional arguments are as follows:

   PARH — Work vector of length equal to `IARDEG` + 1.

   PMAH — Work vector of length equal to `IMADEG` + 1.

   PSIH — Work vector of length equal to `MXLEAD` + 1.

   PSI — Work vector of length equal to `MXLEAD` + 1.

LAGPSI — Work vector of length equal to MXLEAD + 1.

2. If the W series has been transformed using a Box-Cox transformation with parameters POWER and SHIFT, the forecasts and probability limits for the original series may be obtained by application of routine BCTR (page 689) with the same parameters and argument IDIR set equal to one.

Description

Routine NSBJF computes Box-Jenkins forecasts and their associated probability limits for a nonseasonal ARMA model given a sample of n = NOBS observations $\{W_t\}$ for $t = 1, 2, \ldots, n$.

Suppose the time series $\{W_t\}$ is generated by a nonseasonal ARMA model of the form

$$\phi(B)W_t = \theta_0 + \theta(B)A_t \qquad t \in \{0, \pm 1, \pm 2, \ldots\}$$

where B is the backward shift operator, θ_0 = CONST,

$$\phi(B) = 1 - \phi_1 B^{l_\phi(1)} - \phi_2 B^{l_\phi(2)} - \cdots - \phi_p B^{l_\phi(p)}$$
$$\theta(B) = 1 - \theta_1 B^{l_\theta(1)} - \phi_2 B^{l_\theta(2)} - \cdots - \theta_q B^{l_\theta(q)}$$

p = NPAR and q = NPMA. Without loss of generality, we assume

$$1 \le l_\phi(1) \le l_\phi(2) \le \cdots \le l_\phi(p)$$
$$1 \le l_\theta(1) \le l_\theta(2) \le \cdots \le l_\theta(q)$$

so that the nonseasonal ARMA model is of order (p', q') where $p' = l_\phi(p)$ and $q' = l_\theta(q)$. Note that the usual hierarchal model assumes

$$l_\phi(i) = i \qquad 1 \le i \le p$$
$$l_\theta(j) = j \qquad 1 \le j \le q$$

The Box-Jenkins forecast at origin t for lead time l of W_{t+l} is defined in terms of the difference equation

$$\hat{W}_t(l) = \theta_0 + \phi_1[W_{t+l-l_\phi(1)}] + \cdots + \phi_p[W_{t+l-l_\phi(p)}]$$
$$+ [A_{t+l}] - \theta_1[A_{t+l-l_\theta(1)}] - \cdots - \theta_q[A_{t+l-l_\theta(q)}]$$

where

$$[W_{t+k}] = \begin{cases} W_{t+k} & k = 0, -1, -2, \ldots \\ \hat{W}_t(k) & k = 1, 2, \ldots \end{cases}$$

$$[A_{t+k}] = \begin{cases} W_{t+k} - \hat{W}_{t+k-1}(1) & k = 0, -1, -2, \ldots \\ 0 & k = 1, 2, \ldots \end{cases}$$

The $100(1 - \alpha)\%$ probability limits for W_{t+l} are given by

$$\hat{W}_t(l) \pm z_{\alpha/2}\left\{1+\sum_{j=1}^{l-1}\psi_j^2\right\}^{1/2}\sigma_A$$

where $z_{(1-\alpha/2)}$ is the $100(1-\alpha/2)$ percentile of the standard normal distribution,

$$\sigma_A^2 = \text{AVAR}$$

and $\{\psi_j\}$ are the parameters of the random shock form of the difference equation. Note that the forecasts are computed for lead times $l = 1, 2, \ldots, L$ at origins $t = (n-b), (n-b+1), \ldots, n$ where L = MXLEAD and b = MXBKOR.

The Box-Jenkins forecasts minimize the mean square error

$$E[W_{t+l} - \hat{W}_t(l)]^2$$

Also, the forecasts may be easily updated according to the equation

$$\hat{W}_{t+1}(l) = \hat{W}_t(l+1) + \psi_l A_{t+1} \qquad (7)$$

This approach and others are given in Chapter 5 of Box and Jenkins (1976).

IRNSE

Computes estimates of the impulse response weights and noise series of a univariate transfer function model.

Required Arguments

X — Vector of length NOBS containing the input time series. (Input)

Y — Vector of length NOBS containing the output time series. (Input)

MWTIR — Maximum index of the impulse response weights. (Input)
MWTIR must be greater than or equal to zero and less than or equal to NOBS − 1.

MWTSN — Maximum index of the impulse response weights used to compute the noise series. (Input)
MWTSN must be greater than or equal to zero and less than or equal to MWTIR.

WTIR — Vector of length MWTIR + 1 containing the impulse response weight estimates. (Output)
The impulse response weight estimate of index k is given by WTIR(k) for $k = 0, 1, \ldots,$ MWTIR.

SNOISE — Vector of length NOBS − MWTSN containing the noise series based on the impulse response weight estimates. (Output)

XPW — Vector of length NOBS − NPAR containing the prewhitened input time series X. (Output)

YPW — Vector of length NOBS – NPAR containing the prewhitened output time series Y. (Output)

Optional Arguments

NOBS — Number of observations in each time series. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the estimates of the impulse response weights and the noise series.

NPAR — Number of prewhitening autoregressive parameters. (Input)
NPAR must be greater than or equal to zero.
Default: NPAR = size (PAR,1) if PAR is present otherwise NPAR = 0.

PAR — Vector of length NPAR containing the prewhitening autoregressive parameters. (Input)
Default: PAR = 0.0.

NPMA — Number of prewhitening moving average parameters. (Input)
NPMA must be greater than or equal to zero.
Default: NPMA = size (PMA,1) if PAR is present, otherwise NPMA = 0.

PMA — Vector of length NPMA containing the prewhitening moving average parameters. (Input)
Default: PMA = 0.0.

FORTRAN 90 Interface

Generic: `CALL IRNSE (X, Y, MWTIR, MWTSN, WTIR, SNOISE, XPW, YPW,[,…])`

Specific: The specific interface names are S_IRNSE and D_IRNSE.

FORTRAN 77 Interface

Single: `CALL IRNSE (NOBS, X, Y, IPRINT, NPAR, PAR, NPMA, PMA, MWTIR, MWTSN, WTIR, SNOISE, XPW, YPW)`

Double: The double precision name is DIRNSE.

Example

Consider the Gas Furnace Data (Box and Jenkins 1976, pages 532–533) where X is the input gas rate in cubic feet/minute and Y is the percent CO_2 in the outlet gas. Application of routine IRNSE to these data produces the following results:

```
      USE GDATA_INT
      USE IRNSE_INT
      USE WRRRN_INT
      INTEGER    LDX, MWTIR, MWTSN, NDX, NOBS, NOPRIN, NPAR, NPMA
      PARAMETER  (LDX=296, MWTIR=10, NDX=2, NOBS=296, &
                 NOPRIN=0, NPAR=3, NPMA=0, MWTSN=MWTIR)
!
      INTEGER    NCOL, NROW
      REAL       PAR(NPAR), PMA(1), RDATA(296,2), SNOISE(NOBS-MWTSN), &
                 WTIR(MWTIR+1), X(NOBS), XPW(NOBS-NPAR), Y(NOBS), &
                 YPW(NOBS-NPAR)
!
      EQUIVALENCE (X(1), RDATA(1,1)), (Y(1), RDATA(1,2))
!                                Gas Furnace Data
      CALL GDATA (7, RDATA, NROW, NCOL)
!                                Specify AR parameters for
!                                prewhitening transformation
      PAR(1) = 1.97
      PAR(2) = -1.37
      PAR(3) = 0.34
!                                Compute estimates of impulse
!                                response weights and noise series
      CALL IRNSE (X, Y, MWTIR, MWTSN, WTIR, SNOISE, XPW, YPW, PAR=PAR)
!                                Print results
      CALL WRRRN ('WTIR', WTIR, 1, 11, 1)
      CALL WRRRN ('SNOISE', SNOISE, 1, 20, 1)
!
      END
```

Output

```
                                  WTIR
      1        2        3        4        5        6        7        8
-0.0355   0.0716  -0.0764  -0.5655  -0.6549  -0.8936  -0.5358  -0.3482

      9       10       11
-0.0782   0.0277  -0.1364

                                 SNOISE
    1      2      3       4       5       6       7       8       9      10
53.21  53.49  53.72   54.05   53.98   53.95   53.69   53.02   52.56   52.33

   11     12     13      14      15      16      17      18      19      20
52.47  52.69  52.57   52.63   52.81   53.14   53.21   53.20   53.05   52.88
```

Comments

1. Workspace may be explicitly provided, if desired, by use of I2NSE/DI2NSE. The reference is:

```
CALL I2NSE (NOBS, X, Y, IPRINT, NPAR, PAR, NPMA, PMA, MWTIR,
MWTSN, WTIR, SNOISE, XPW, YPW, ACPWX, ACPWY, CCPW)
```

The additional arguments are as follows:

ACPWX — Vector of length MWTIR + 1 containing the estimated autocorrelation function of PWX. (Output)

ACPWY — Vector of length MWTIR + 1 containing the estimated autocorrelation function of PWY. (Output)

CCPW — Vector of length 2 * MWTIR + 1 containing the estimated cross-correlation function of PWX and PWY. (Output)

2. The input series X and output series Y are assumed to be the result of transforming and differencing the raw input and output series. The routines BCTR (page 689) and DIFF (page 692) may be used, respectively, to perform a Box-Cox transformation and difference the raw input and output series.

3. Note that the prewhitened input and output are computed at time t = NPAR + 1 through t = NOBS. Also, the noise series is computed at time t = MWTSN + 1 through t = NOBS.

Description

Routine IRNSE estimates the impulse response weights and noise series of a transfer function model given a sample of n = NOBS observations of the input $\{x_t\}$ and output $\{y_t\}$ for $t = 1, 2, \ldots, n$. Define $\{x_t\}$ and $\{y_t\}$, respectively, by

$$x_t = \begin{cases} X_t - \hat{\mu}_X & d = 0 \\ \nabla^d X_t & d > 0 \end{cases}$$

and

$$y_t = \begin{cases} Y_t - \hat{\mu}_Y & d = 0 \\ \nabla^d Y_t & d > 0 \end{cases}$$

where $\{X_t\}$ and $\{Y_t\}$ for $t = (-d + 1), \ldots, n$ represent the undifferenced input and output series with

$$\hat{\mu}_X \text{ and } \hat{\mu}_Y$$

estimates of their respective means. The differenced input and output series may be obtained using the routine DIFF (page 692) following any preliminary transformation of the data.

The transfer function model is defined by

$$Y_t = \nu(B)X_t + N_t$$

or, equivalently,

$$y_t = \nu(B)x_t + n_t$$

with transfer function

$$\nu(B) = \nu_0 + \nu_1 B + \nu_2 B^2 + \ldots$$

and differenced noise series $n_t = \nabla^d N_t$.

The prewhitened input and output series are computed for $t = (p + 1), \ldots, n$ according to

$$\alpha_t = \phi(B)x_t + \theta_1(B)\alpha_t$$
$$\beta_t = \phi(B)y_t + \theta_1(B)\beta_t$$

where

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p$$
$$\theta(B) = \theta_1 B + \theta_2 B^2 + \ldots + \theta_q B^q$$

The parameters of the prewhitening transformation may be estimated roughly using the routine NSPE (page 727) or more precisely using the routine NSLSE (page 733). The correlation relationship between $\{\alpha_t\}$, $\{\beta_t\}$, and $\{n_t\}$ may be further examined using the routines ACF (page 697), PACF (page 702), and CCF (page 706).

The *impulse response weights* $\{\nu_k\}$ are estimated by

$$\hat{\nu}_k = \frac{\hat{\sigma}_\beta}{\hat{\sigma}_\alpha}\hat{\rho}_{\alpha\beta}(k) \quad k = 0,1,\ldots,K$$

where K = MWTIR,

$$\hat{\sigma}_\alpha \text{ and } \hat{\sigma}_\beta$$

denote the standard deviation of $\{\alpha_t\}$ and $\{\beta_t\}$;

$$\hat{\rho}_{\alpha\beta}(k) \text{ for } k = 0,1,\ldots,K$$

represents the cross-correlation function between $\{\alpha_t\}$ and $\{\beta_t\}$. The differenced noise series $\{n_t\}$ for $t = (K' + 1), \ldots, n$ is reconstructed using the model

$$n_t = y_t - \hat{\nu}_0 x_t - \hat{\nu}_1 x_{t-1} - \cdots - \hat{\nu}_{K'} x_{t-K'}$$

where K' = MWTSN.

TFPE

Computes preliminary estimates of parameters for a univariate transfer function model.

Required Arguments

NDELAY — Time delay parameter. (Input)
NDELAY must be greater than or equal to zero.

WTIR — Vector of length MWTIR + 1 containing the impulse response weight estimates. (Input)
The impulse response weight estimate of index k is given by WTIR(k) for k = 0, 1, …, MWTIR.

SNOISE — Vector of length NSNOIS containing the noise series. (Input)

AVAR — Estimate of the random shock variance. (Output)

Optional Arguments

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints estimates of transfer function parameters, estimates of noise model parameters, and an of the random shock variance.

NPLHS — Number of left-hand side transfer function parameters. (Input)
NPLHS must be greater than or equal to zero.
Default: NPLHS = size (PLHS,1) if PLHS is present. Otherwise, NPLHS=0.

NPRHS — Number of right-hand side transfer function parameters (excluding the index 0 parameter). (Input)
NPRHS must be greater than or equal to zero.
Default: NPRHS = size (PRHS,1) – 1 if PRHS is present. Otherwise, NPRHS=0.

NPNAR — Number of noise autoregressive parameters. (Input)
NPNAR must be greater than or equal to zero.
Default: NPNAR = size (PNAR,1) if PNAR is present. Otherwise, NPNAR=0.

NPNMA — Number of noise moving average parameters. (Input)
NPNMA must be greater than or equal to zero.
Default: NPNMA = size (PNMA,1) if PNMA is present. Otherwise, NPNMA=0.

MWTIR — Maximum index of the impulse response weights. (Input)
MWTIR must be greater than or equal to NPLHS + NPRHS + NDELAY.
Default: MWTIR = size (WTIR,1) –1.

NSNOIS — Number of elements in the noise series. (Input)
NSNOIS must be greater than or equal to NPNAR + NPNMA + 1.
Default: NSNOIS = size (SNOISE,1).

RELERR — Stopping criterion for use in the nonlinear equation solver. (Input)
If RELERR = 0.0, then the default value RELERR = 100.0 * AMACH(4) is used. See the

documentation for routine AMACH (in the Reference Material section of this manual). Default: RELERR = 0.0.

MAXIT — The maximum number of iterations allowed in the nonlinear equation solver. (Input)
If MAXIT = 0, then the default value MAXIT = 200 is used.
Default: MAXIT = 0.

PLHS — Vector of length NPLHS containing the estimates of the left-hand side transfer function parameters. (Output)
The LHS weight estimates are PLHS(k), $k = 1, \ldots,$ NPLHS.

PRHS — Vector of length NPRHS + 1 containing the estimates of the right-hand side transfer function parameters. (Output)
The RHS weight estimates are PRHS(k), $k = 0, \ldots,$ NPRHS.

PNAR — Vector of length NPNAR containing the estimates of the noise autoregressive parameters. (Output)

PNMA — Vector of length NPNMA containing the estimates of the noise moving average parameters. (Output)

FORTRAN 90 Interface

Generic: `CALL TFPE (NDELAY, WTIR, SNOISE, AVAR [,…])`

Specific: The specific interface names are S_TFPE and D_TFPE.

FORTRAN 77 Interface

Single:
```
CALL TFPE (IPRINT, NPLHS, NPRHS, NPNAR, NPNMA, NDELAY,
MWTIR, WTIR, NSNOIS, SNOISE, RELERR, MAXIT, PLHS, PRHS,
PNAR, PNMA, AVAR)
```

Double: The double precision name is DTFPE.

Example

Consider the Gas Furnace Data (Box and Jenkins 1976, pages 532–533) where X is the input gas rate in cubic feet/minute and Y is the percent CO_2 in the outlet gas. The data is retrieved by routine GDATA (page 1302). Routine IRNSE (page 758) computes the impulse response weights. Application of routine TFPE to these weights produces the following results:

```
USE GDATA_INT
USE IRNSE_INT
USE WROPT_INT
USE TFPE_INT
INTEGER    MWTIR, MWTSN, NDELAY, NOBS, NPAR, NPLHS, NPMA, NPNAR, &
          NPNMA, NPRHS, NSNOIS
```

```
      PARAMETER  (MWTIR=10, NDELAY=3, NOBS=100, NPAR=3, NPLHS=2, &
                 NPMA=0, NPNAR=2, NPNMA=0, NPRHS=2, MWTSN=MWTIR, &
                 NSNOIS=NOBS-MWTSN)
!
      INTEGER    IPRINT, ISETNG, NCOL, NROW
      REAL       AVAR, PAR(NPAR), PLHS(NPLHS), PMA(1), PNAR(NPNAR), &
                 PNMA(1), PRHS(NPRHS+1), RDATA(296,2), &
                 SNOISE(NOBS-MWTSN), WTIR(MWTIR+1), X(NOBS), &
                 XPW(NOBS-NPAR), Y(NOBS), YPW(NOBS-NPAR)
!
      EQUIVALENCE (X(1), RDATA(1,1)), (Y(1), RDATA(1,2))
!                                Gas Furnace Data
      CALL GDATA (7, RDATA, NROW, NCOL)
!                                Specify AR parameters for
!                                prewhitening transformation
      PAR(1) = 1.97
      PAR(2) = -1.37
      PAR(3) = 0.34
!                                Compute estimates of impulse
!                                response weights and noise series
      CALL IRNSE (X, Y, MWTIR, MWTSN, WTIR, &
                  SNOISE, XPW, YPW, PAR=PAR)
!                                Convergence parameters
!                                Compute preliminary estimates of
!                                transfer function parameters
      ISETNG = 1
      CALL WROPT (-6, ISETNG, 1)
      IPRINT = 1
      CALL TFPE (NDELAY, WTIR, SNOISE, AVAR, IPRINT=IPRINT, NPLHS=NPLHS, &
                  NPRHS=NPRHS, NPNAR=NPNAR, PLHS=PLHS, PRHS=PRHS, PNAR=PNAR)
!
      END
```

Output

```
PLHS from TFPE/T2PE
       1            2
0.120342      0.326149

        PRHS from TFPE/T2PE
        1             2             3
-0.623240      0.318698      0.362488

PNAR from TFPE/T2PE
      1            2
1.64679      -0.70916

PNMA is not written since NPNMA = 0

AVAR from TFPE/T2PE =     2.85408E-02
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `T2PE/DT2PE`. The reference is:

```
CALL T2PE (IPRINT, NPLHS, NPRHS, NPNAR, NPNMA, NDELAY, MWTIR,
WTIR, NSNOIS, SNOISE, RELERR, MAXIT, PLHS, PRHS, PNAR, PNMA,
AVAR, A, FAC, IPVT, WK, ACV, PARWK, ACVMOD, TAUINI, TAU, FVEC,
FJAC, R, QTF, WKNLN, H)
```

The additional arguments are as follows:

A — Work vector of length (max(NPLHS, NPNAR))^2.

FAC — Work vector of length (max(NPLHS, NPNAR))^2.

IPVT — Work vector of length max(NPLHS, NPNAR).

WK — Work vector of length max(NPLHS, NPNAR).

ACV — Work vector of length NPNAR + NPNMA + 1.

PARWK — Work vector of length NPNAR + 1.

ACVMOD — Work vector of length NPNMA + 1.

TAUINI — Work vector of length NPNMA + 1.

TAU — Work vector of length NPNMA + 1.

FVEC — Work vector of length NPNMA + 1.

FJAC — Work vector of length (NPNMA + 1)^2.

R — Work vector of length (NPNMA + 1) * (NPNMA + 2)/2.

QTF — Work vector of length NPNMA + 1.

WKNLN — Work vector of length 5 * (NPNMA + 1).

H — Work vector of length NPLHS.

2. Informational error

Type	Code	
4	1	The nonlinear equation solver did not converge to RELERR within MAXIT iterations.

3. The impulse response weight estimates and the noise series may be computed using routine IRNSE (page 758).

Description

Routine TFPE computes preliminary estimates of the parameters of a transfer function model given a sample of n = NOBS observations of the differenced input $\{x_t\}$ and differenced output $\{y_t\}$ for $t = 1, 2, \ldots, n$.

Define $\{x_t\}$ and $\{y_t\}$, respectively, by

$$x_t = \begin{cases} X_t - \hat{\mu}_X & d = 0 \\ \nabla^d X_t & d > 0 \end{cases}$$

and

$$y_t = \begin{cases} Y_t - \hat{\mu}_Y & d = 0 \\ \nabla^d Y_t & d > 0 \end{cases}$$

where $\{X_t\}$ and $\{Y_t\}$ for $t = (-d + 1), \ldots, n$ represent the undifferenced input and output series with

$$\hat{\mu}_X \text{ and } \hat{\mu}_Y$$

estimates of their respective means. The differenced input and output series may be obtained using the routine DIFF (page 692) following any preliminary transformation of the data.

The transfer function model is defined by

$$Y_t = \delta^{-1}(B)\omega(B)X_{t-b} + N_t$$

or, equivalently,

$$y_t = \delta^{-1}(B)\omega(B)x_{t-b} + n_t$$

where $n_t = \nabla^d N_t$ and the left-hand side and right-hand side transfer function polynomial operators are

$$\delta(B) = 1 - \delta_1 B - \delta_2 B^2 - \ldots - \delta_r B^r$$
$$\omega(B) = \omega_0 - \omega_1 B - \omega_2 B^2 - \ldots - \omega_s B^s$$

with r = NPLHS, s = NPRHS, and b = NDELAY. The noise process $\{N_t\}$ and the input process $\{X_t\}$ are assumed to be independent with the noise process given by the ARIMA model

$$\phi(B)n_t = \theta(B)A_t$$

where

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p$$
$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q$$

with p = NPNAR and q = NPNMA.

The impulse response weights and the transfer function parameters are related by

$$\nu_k = \begin{cases} 0 & k = 0, 1, \ldots, b-1 \\ \sum_{j=1}^{r} \delta_j \nu_{k-j} + \omega_0 & k = b \\ \sum_{j=1}^{r} \delta_j \nu_{k-j} - \omega_{k-b} & k = b+1, b+2, \ldots, b+s \\ \sum_{j=1}^{r} \delta_j \nu_{k-j} & k = b+s+1, b+s+2, \ldots \end{cases}$$

See Abraham and Ledolter (1983, page 341). The r left-hand side transfer function parameters are estimated using the difference equation given as the last case above. The resulting estimates

$$\hat{\delta}_1, \ldots, \hat{\delta}_r$$

are then substituted into the middle two cases to determine the $s + 1$ estimates

$$\hat{\omega}_0, \hat{\omega}_1, \ldots, \hat{\omega}_s$$

The noise series parameters are estimated using the routine NSPE (page 727). The impulse response weights $\{\nu_k\}$ and differenced noise series $\{n_t\}$ may be computed using the routine IRNSE (page 758). See Box and Jenkins (1976, pages 511–513).

MLSE

Computes least-squares estimates of a linear regression model for a multichannel time series with a specified base channel.

Required Arguments

X — NOBSX by NCHANX matrix containing the time series. (Input)
Each row of X corresponds to an observation of a multivariate time series, and each column of X corresponds to a univariate time series. The base time series or output channel is contained in the first column.

NDIFF — Vector of length NCHANX containing the order of differencing for each channel of X. (Input)
The elements of NDIFF must be greater than or equal to zero.

NDPREG — Vector of length NCHANX containing the number of regression parameters in the differenced form of the model for each channel of X. (Input) The elements of NDPREG must be greater than or equal to zero.

LAG — Vector of length NCHANX containing the amount of time that each channel of X is to lag the base series. (Input)
The elements of LAG must be greater than or equal to zero.

CNST — Estimate of the overall constant. (Output)

NPREG — Number of regression parameters in the undifferenced model. (Output)

NPREG = IADD + (NDPREG(1) + NDIFF(1)) + … + (NDPREG(NCHANX) + DIFF(NCHANX)

where

IADD = NDIFF(1), if NDPREG(1) = 0
IADD = max (0, min(LAG(1) – 1, NDIFF(1))), if NDPREG(1) > 0.

PREG — Vector of length NPREG containing the regression parameters in the undifferenced model. (Output)
The parameter estimates are concatenated as follows.

Channel 1: REG(i), i = 1, 2, …, IADD + NDPREG(1) + NDIFF(1)

Channel j: PREG(i), i = I(j) + 1, I(j) + 2, …, I(j) + NDPREG(j) + NDIFF(j)

where

$I(j)$ = IADD + NDPREG(1) + NDIFF(1) + … + NDPREG(j – 1) + NDIFF(j – 1)
for j = 2, 3, …, NCHANX.

Optional Arguments

NOBSX — Number of observations in each channel of the time series X. (Input)
NOBSX must be less than or equal LDX and greater than max(NDPREG(i) + LAG(i)) for i = 1, 2, …, NCHANX.
Default: NOBSX = size (X,1).

NCHANX — Number of channels in the time series X. (Input)
NCHANX must be greater than or equal to one.
Default: NCHANX = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement of the calling program. (Input)
Default: LDX = size (X,1).

IMEAN — Option for computation of the means of the channels of X. (Input)
Default: IMEAN = 1.

IMEAN	Action
0	XMEAN is user specified.
1	XMEAN is set to the vector of arithmetic means of the channels of X.

XMEAN — Vector of length NCHANX containing the means of the channels of X. (Input, if IMEAN = 0; output, if IMEAN = 1)

FORTRAN 90 Interface

Generic: `CALL MLSE (X, NDIFF, NDPREG, LAG, CNST, NPREG, PREG [,…])`

Specific: The specific interface names are `S_MLSE` and `D_MLSE`.

FORTRAN 77 Interface

Single:
```
CALL MLSE (NOBSX, NCHANX, X, LDX, IMEAN, XMEAN, NDIFF,
NDPREG, LAG, CNST, NPREG, PREG)
```

Double: The double precision name is `DMLSE`.

Example 1

Consider the Wölfer Sunspot Data (Box and Jenkins 1976, page 530) along with data on northern light activity and earthquake activity (Robinson 1967, page 204) to be a three-channel time series. Routine `MLSE` is applied to these data to examine the regressive relationship between the channels.

```
      USE GDATA_INT
      USE MLSE_INT
      USE UMACH_INT
      INTEGER    LDX, NCHANX, NOBSX
      PARAMETER  (NCHANX=3, NOBSX=100, LDX=NOBSX)
!
      INTEGER    I, IMEAN, LAG(NCHANX), NCOL, NDIFF(NCHANX), &
                 NDPREG(NCHANX), NOUT, NPREG, NROW
      REAL       CNST, PREG(20), RDATA(100,4), X(LDX,NCHANX), &
                 XMEAN(NCHANX)
!
      EQUIVALENCE (X(1,1), RDATA(1,2)), (X(1,2), RDATA(1,3)), &
                 (X(1,3), RDATA(1,4))
!
      DATA NDIFF(1), NDIFF(2), NDIFF(3)/1, 1, 0/
      DATA LAG(1), LAG(2), LAG(3)/1, 2, 1/
      DATA NDPREG(1), NDPREG(2), NDPREG(3)/2, 1, 3/
!
      CALL GDATA (8, RDATA, NROW, NCOL)
!     USE Default Option to estimate channel means
!                                Compute regression parameters
      CALL MLSE (X, NDIFF, NDPREG, LAG, CNST, NPREG, PREG, XMEAN=XMEAN)
!
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99993)
99993 FORMAT (//, 1X, '     Results of MLSE/M2SE')
      WRITE (NOUT,99994)
99994 FORMAT (1X, '  I   NDIFF(I)  LAG(I)  NDPREG(I)     XMEAN(I)')
      DO 10  I=1, NCHANX
         WRITE (NOUT,99995) I, NDIFF(I), LAG(I), NDPREG(I), XMEAN(I)
99995    FORMAT (1X, 4(I3,6X), F12.4)
   10 CONTINUE
```

```
      WRITE (NOUT,99996) CNST
99996 FORMAT (1X, 'Overall constant, CNST = ', F12.4)
      WRITE (NOUT,99997) NPREG
99997 FORMAT (//, 1X, 'Total number of parameters, NPREG = ', I2)
      WRITE (NOUT,99998)
99998 FORMAT (//, 1X, ' I          PREG(I)')
      DO 20  I=1, NPREG
         WRITE (NOUT,99999) I, PREG(I)
99999    FORMAT (1X, I2, 5X, F12.4)
   20 CONTINUE
!
      END
```

Output

```
Results of MLSE/M2SE
I   NDIFF(I)  LAG(I)  NDPREG(I)     XMEAN(I)
1        1        1        2           46.9400
2        1        2        1           63.4300
3        0        1        3           97.9700
Overall constant, CNST =      -7.2698

Total number of parameters, NPREG =  8

I          PREG(I)
1          -0.1481
2          -1.3444
3           0.4925
4          -0.0302
5           0.0302
6          -0.0210
7           0.0187
8           0.0765
```

Comments

1. Workspace may be explicitly provided, if desired, by use of M2SE/DM2SE. The reference is:

   ```
   CALL M2SE (NOBSX, NCHANX, X, LDX, IMEAN, XMEAN, NDIFF, NDPREG,
   LAG, CNST, NPREG, PREG, XWK, IWK)
   ```

   The additional arguments are as follows:

   XWK — Work vector of length NOBSX * NCHANX+ 2 * $NSUM^2$+ max(IADD, NCHANX + NSUM), where NSUM = NDPREG(1) + … + NDPREG(NCHANX).

   IWK — Work vector of length NSUM.

2. Prior to parameter estimation, the channels of X are centered and/or differenced according to XMEAN and NDIFF, respectively.

3. The undifferenced predictive form of the model is

X(t, 1) = CNST + PREG(1) * X(t − 1, 1) + … + PREG(IADD) * X(t − IADD, 1) + PREG(IADD + 1) * X(t − LAG(1), 1) + … + PREG(IADD + NDPREG(1) + NDIFF(1)) * X(t − LAG(1) + 1 − NDPREG(1) − NDIFF(1), 1) + … + PREG($I(j)$ + 1) * X(t − LAG(j), j) + … + PREG($I(j)$+NDPREG(j)+NDIFF(j)) * X(t − LAG(j) + 1 − NDPREG(j) − NDIFF(j), j) + …

where

$I(j)$ = IADD + NDPREG(1) + NDIFF(1) + … + NDPREG(j − 1) + NDIFF(j − 1)

for j = 2, 3, …, NCHANX.

Description

Routine MLSE performs least-squares estimation of a linear regression model for a multichannel time series with a specified base channel.

Define the multichannel time series X by

$$X = (X_1, X_2, \ldots, X_m)$$

where

$$X_j = (X_{1j}, X_{2j}, \ldots, X_{nj})^T \qquad j = 1, 2, \ldots, m$$

with n = NOBSX and m = NCHANX. The columns of X correspond to individual channels of a multichannel time series and may be examined from a univariate perspective. The rows of X correspond to observations of an m-variate time series and may be examined from a multivariate perspective. Note that an alternative characterization of the multivariate time series X considers the columns of X to be observations of an m-variate time series with the rows of X containing univariate time series. For example, see Priestley (1981, page 692) and Fuller (1976, page 14).

The model is formed by regressing the base series X_1 on its previous values and on the remaining channels $X_2, \ldots, X_m$. The differenced form of the model is given by

$$X_{t1} = \theta_0 + \phi_1(B)\nabla^{d_1}B^{l_1}X_{t1} + \phi_2(B)\nabla^{d_2}B^{l_2}X_{t2} + \ldots + \phi_m(B)\nabla^{d_m}B^{l_m}X_{tm}$$

where θ_0 = CNST is the overall constant, d_k = NDIFF(k) is the order of differencing X_k, l_k = LAG(k) is the amount X_k lags X_1,

$$\phi_k(B) = \phi_{1k} + \phi_{2k}B + \ldots + \phi_{p_k,k}B^{p_k-1}$$

and p_k = NDPREG(k) for k = 1, 2, …, m.

The undifferenced form of the model is given by

$$X_{t1} = \theta_0 + \varphi_1(B)X_{t-l_1,1} + \varphi_2(B)X_{t-l_2,2} + \ldots + \varphi_m(B)X_{t-l_m,m}$$

where the undifferenced parameters φ_k = PREG(k) are defined by

$$\varphi_k(B) = \phi_k(B)\nabla^{d_k}$$
$$= \varphi_1 + \varphi_2 B + \cdots + \varphi_{p_k+d_k} B^{p_k+d_k-1}$$

for $k = 1, 2, \ldots, m$. Note that if $l_1 \geq d_1 \geq 0$, the base series terms $X_{t-j,1}$ at lags $j = 1, \ldots, (l_1 - 1)$ are omitted from the right-hand side of the above model when $d_1 \geq 1$. In the actual computations, these terms are included.

Example 2

Consider the Gas Furnace Data (Box and Jenkins 1976, pages 532–533) where X_1 is the percent CO_2 in the outlet gas and X_2 is the input gas rate in cubic feet/minute. Application of routine MLSE to these data produces the following results:

```
      USE GDATA_INT
      USE SCOPY_INT
      USE MLSE_INT
      USE UMACH_INT
      INTEGER    LDX, NCHANX, NOBSX
      PARAMETER  (NCHANX=2, NOBSX=296, LDX=NOBSX)
!
      INTEGER    I, IMEAN, LAG(NCHANX), NCOL, NDIFF(NCHANX), &
                 NDPREG(NCHANX), NOUT, NPREG, NROW
      REAL       CNST, PREG(20), RDATA(296,2), X(LDX,NCHANX), &
                 XMEAN(NCHANX)
!
      DATA NDIFF(1), NDIFF(2)/0, 0/
      DATA LAG(1), LAG(2)/1, 3/
      DATA NDPREG(1), NDPREG(2)/2, 3/
!                                Gas Furnace Data
      CALL GDATA (7, RDATA, NROW, NCOL)
!                                Multichannel X consists of
!                                Column 1: Output percent CO2
!                                Column 2: Input gas rate
      CALL SCOPY (NOBSX, RDATA(1:,2), 1, X(1:,1), 1)
      CALL SCOPY (NOBSX, RDATA(1:,1), 1, X(1:,2), 1)
!                                Option to estimate channel means
      IMEAN = 1
!                                Compute regression parameters
      CALL MLSE (X, NDIFF, NDPREG, LAG, CNST, NPREG, PREG, XMEAN=XMEAN)
!
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99993)
99993 FORMAT (1X, 'Results of MLSE/M2SE on Gas Furnace Data')
      WRITE (NOUT,99994)
99994 FORMAT (1X, '  I   NDIFF(I)  LAG(I)  NDPREG(I)       XMEAN(I)')
      DO 10  I=1, NCHANX
         WRITE (NOUT,99995) I, NDIFF(I), LAG(I), NDPREG(I), XMEAN(I)
99995    FORMAT (1X, 4(I3,6X), F12.4)
   10 CONTINUE
      WRITE (NOUT,99996) CNST
99996 FORMAT (1X, 'Overall constant, CNST = ', F12.4)
      WRITE (NOUT,99997) NPREG
```

```
99997 FORMAT (1X, 'Total number of parameters, NPREG = ', I2)
      WRITE (NOUT,99998)
99998 FORMAT (1X, ' I          PREG(I)')
      DO 20  I=1, NPREG
         WRITE (NOUT,99999) I, PREG(I)
99999    FORMAT (1X, I2, 5X, F12.4)
   20 CONTINUE
!
      END
```

Output

```
Results of MLSE/M2SE on Gas Furnace Data
I   NDIFF(I)  LAG(I)  NDPREG(I)       XMEAN(I)
1        0        1        2           53.5091
2        0        3        3           -0.0568
Overall constant, CNST =       2.6562
Total number of parameters, NPREG =  5
I          PREG(I)
1           1.6063
2          -0.6561
3          -0.4837
4          -0.1653
5           0.5052
```

MWFE

Computes least-squares estimates of the multichannel Wiener filter coefficients for two mutually stationary multichannel time series.

Required Arguments

CXX — Array of size NCHX by NCHX by MLFIL containing the autocovariances of the input time series X. (Input)

CZX — Array of size NCHZ by NCHX by MLFIL containing the cross-covariances between the desired output time series Z and the input time series X. (Input)

EPS — Lower bound for the normalized mean square error. (Input)

TRACE — Trace of the autocovariance matrix of the desired output time series Z at time lag zero. (Input)

LFIL — Length of the Wiener filter. (Output)

FIL — Array of size NCHZ by NCHX by MLFIL containing the multichannel Wiener filter coefficients. (Output)

ENMS — Vector of length MLFIL containing the normalized mean square error corresponding to each filter length. (Output)

Optional Arguments

NCHX — Number of input channels. (Input)
NCHX must be greater than or equal to one.
Default: NCHX = size (CXX,1).

MLFIL — Maximum length of the Wiener filter. (Input)
MLFIL must be greater than or equal to one.
Default: MLFIL = size (CXX,3).

LDCXX — Leading dimension of CXX exactly as specified in the dimension statement of the calling program. (Input)
LDCXX must be greater than or equal to NCHX.
Default: LDCXX = size (CXX,1).

MDCXX — Middle dimension of CXX exactly as specified in the dimension statement of the calling program. (Input)
MDCXX must be greater than or equal to NCHX.
Default: MDCXX = size (CXX,2).

NCHZ — Number of channels in desired output time series. (Input)
NCHZ must be greater than or equal to one.
Default: NCHZ = size (CZX,1).

LDCZX — Leading dimension of CZX exactly as specified in the dimension statement of the calling program. (Input)
LDCZX must be greater than or equal to NCHZ.
Default: LDCZX = size (CZX,1).

MDCZX — Middle dimension of CZX exactly as specified in the dimension statement of the calling program. (Input)
MDCZX must be greater than or equal to NCHX.
Default: MDCZX = size (CZX,2).

LDFIL — Leading dimension of FIL exactly as specified in the dimension statement of the calling program. (Input)
LDFIL must be greater than or equal to NCHZ.
Default: LDFIL = size (FIL,1).

MDFIL — Middle dimension of FIL exactly as specified in the dimension statement of the calling program. (Input)
MDFIL must be greater than or equal to NCHX.
Default: MDFIL = size (FIL,2).

FORTRAN 90 Interface

Generic: `CALL MWFE (CXX, CZX, EPS, TRACE, LFIL, FIL, ENMS [,…])`

Specific: The specific interface names are `S_MWFE` and `D_MWFE`.

FORTRAN 77 Interface

Single:
```
CALL MWFE (NCHX, MLFIL, CXX, LDCXX, MDCXX, NCHZ, CZX,
LDCZX, MDCZX, EPS, TRACE, LFIL, FIL, LDFIL, MDFIL ENMS)
```

Double: The double precision name is `DMWFE`.

Example

Consider the Wölfer Sunspot Data (Box and Jenkins 1976, page 530) along with data on northern light activity and earthquake activity (Robinson 1967, page 204) to be a three-channel time series. Routine `MWFE` applied to these data determines the following Wiener filter:

```
!                                  SPECIFICATIONS FOR PARAMETERS
      USE GDATA_INT
      USE SCOPY_INT
      USE MCCF_INT
      USE MWFE_INT
      USE UMACH_INT

      INTEGER    LDCXX, LDCZX, LDFIL, LDX, LDZ, MAXLAG, MDCXX, &
                 MDCZX, MDFIL, MLFIL, NCHANX, NCHANZ, NOBSX, NOBSZ
      PARAMETER  (MLFIL=3, NCHANX=3, NCHANZ=3, NOBSX=99, NOBSZ=99, &
                  LDCXX=NCHANX, LDCZX=NCHANZ, LDFIL=NCHANX, LDX=NOBSX, &
                  LDZ=NOBSZ, MAXLAG=MLFIL-1, MDCXX=NCHANX, MDCZX=NCHANX, &
                  MDFIL=NCHANZ)
!
      INTEGER    I, J, K, LFIL, NCOL, NOUT, NROW
      REAL       CVXX(LDCXX,MDCXX,-MAXLAG:MAXLAG), CVXX1(3,3,3), &
                 CVZX(LDCZX,MDCZX,-MAXLAG:MAXLAG), CVZX1(3,3,3), &
                 CXX(LDCXX,MDCXX,-MAXLAG:MAXLAG), &
                 CZX(LDCZX,MDCZX,-MAXLAG:MAXLAG), ENMS(MLFIL), EPS, &
                 FIL(LDFIL,MDFIL,MLFIL), R(0:2), RDATA(100,4), &
                 TRACE, X(LDX,NCHANX), XMEAN(NCHANX), XVAR(NCHANX), &
                 YMEAN, YVAR, Z(LDZ,NCHANZ), ZMEAN(NCHANZ), &
                 ZVAR(NCHANZ)

!
      EQUIVALENCE (CVXX(1,1,0), CVXX1(1,1,1)), (CVZX(1,1,0), CVZX1(1,1, &
                1))
!                                  Wolfer sunspot numbers
!                                  Northern lights activity
!                                  Earthquake activity
      CALL GDATA (8, RDATA, NROW, NCOL)
!
      CALL SCOPY (NOBSX, RDATA(1:,2), 1, X(1:,1), 1)
      CALL SCOPY (NOBSX, RDATA(1:,3), 1, X(1:,2), 1)
      CALL SCOPY (NOBSX, RDATA(1:,4), 1, X(1:,3), 1)
!
      CALL SCOPY (NOBSZ, RDATA(2:,2), 1, Z(1:,1), 1)
      CALL SCOPY (NOBSZ, RDATA(2:,3), 1, Z(1:,2), 1)
```

```
      CALL SCOPY (NOBSZ, RDATA(2:,4), 1, Z(1:,3), 1)
!                                  Compute multichannel ACF of Z
      CALL MCCF (Z, Z, MAXLAG, CXX, XVAR=XVAR, CCV=CVXX)
!                                  Compute TRACE
      TRACE = SSUM(NCHANZ,XVAR,1)
!                                  Compute multichannel ACF of X
      CALL MCCF (X, X, MAXLAG, CXX, CCV=CVXX)
!                                  Compute multichannel CCF of Z and X
      CALL MCCF (Z, X, MAXLAG, CZX, CCV=CVZX)
!                                  Bound normalized MSE to be positive
      EPS = 0.0
!                                  Reverse the LAG direction and scale
!                                  to agree with Robinson (1967)
      R(0)  = 99.D0
      R(1)  = 98.D0
      R(2)  = 97.D0
      TRACE = TRACE*R(0)
      DO 10  K=0, MAXLAG
         DO 10  J=1, NCHANX
            DO 10  I=1, NCHANX
               CVXX(I,J,K) = CVXX(I,J,-K)*R(K)
               CVZX(I,J,K) = CVZX(I,J,-K)*R(K)
   10 CONTINUE
!                                  Compute multichannel Wiener filter
      CALL MWFE (CVXX1, CVZX1, EPS, TRACE, LFIL, FIL, ENMS)
!                                  Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99994) LFIL
99994 FORMAT (1X, 'Number of filter coefficients, LFIL = ', I3)
      DO 30  K=1, LFIL
         WRITE (NOUT,99995) K
99995    FORMAT (//, 1X, 'Wiener filter coefficient of index K = ', I3)
         DO 20  I=1, NCHANX
            WRITE (NOUT,99996) (FIL(I,J,K),J=1,NCHANZ)
99996       FORMAT (1X, 3F12.4)
   20    CONTINUE
   30 CONTINUE
      WRITE (NOUT,99997)
99997 FORMAT (//, 1X, 'Normalized mean square error')
      WRITE (NOUT,99998)
99998 FORMAT (1X, ' K          ENMS(K)')
      DO 40  K=1, LFIL
         WRITE (NOUT,99999) K, ENMS(K)
99999    FORMAT (1X, I2, 5X, F12.4)
   40 CONTINUE
!
      END
```

Output

```
Number of filter coefficients, LFIL =   3

Wiener filter coefficient of index K =   1
 1.3834      0.0348      0.0158
 0.0599      0.8266      0.0629
-0.1710     -0.0332     -0.1205
```

```
Wiener filter coefficient of index K =   2
 -0.7719     -0.0183     -0.0318
-0.0040     -0.2328      0.0484
-0.2170      0.1912     -0.0667

Wiener filter coefficient of index K =   3
 0.0516      0.0563     -0.0138
-0.0568      0.1084     -0.1731
 0.0007      0.2177     -0.0152

Normalized mean square error
 K          ENMS(K)
 1           0.6042
 2           0.5389
 3           0.5174
```

Comments

1. Workspace may be explicitly provided, if desired, by use of M2FE/DM2FE. The reference is:

   ```
   CALL M2FE (NCHX, MLFIL, CXX, LDCXX, MDCXX, NCHZ, CZX, LDCZX,
   MDCZX, EPS, TRACE, LFIL, FIL, LDFIL, MDFIL, ENMS, IWK, WK)
   ```

   The additional arguments are as follows:

   IWK — Work vector of length NCHX.

   WK — Work vector of length NCHX * NCHX * (2 * MLFIL + 12) + NCHZ.

2. The length of the filter is determined by the arguments EPS and MLFIL. Iteration to a longer filter stops when either the normalized mean square error ENMS is less than EPS, or the filter reaches the maximum allowable length, MLFIL.

3. The routine MCCF (page 711) may be used to obtain the input arguments CXX, CZX, and TRACE. For TRACE, routine MCCF may be used to obtain the autocovariances of the desired output series Z. In particular, TRACE = ZVAR(1) + … + ZVAR(NCHZ).

4. For a given lag k, the multichannel cross-covariance coefficient between Z and X is defined as the array of size NCHZ by NCHX whose elements are the single-channel crosscovariance coefficients CZX(i, j, k).

Description

Routine MFFE computes least-squares estimates of the multichannel Wiener filter coefficients for two mutually stationary multichannel time series.

Define the multichannel time series X by

$$X = (X_1, X_2, \ldots, X_p)$$

where

$$X_j = (X_{1j}, X_{2j}, \ldots, X_{nj})^T \qquad j = 1, 2, \ldots, p$$

with p = NCHX. Similarly, define the multichannel time series Z by

$$Z = (Z_1, Z_2, \ldots, Z_q)$$

where

$$Z_j = (Z_{1j}, Z_{2j}, \ldots, Z_{mj})^T \qquad j = 1, 2, \ldots, q$$

with q = NCHZ. The columns of X and Z correspond to individual channels of multichannel time series and may be examined from a univariate perspective. The rows of X and Z correspond to observations of p-variate and q-variate time series and may be examined from a multivariate perspective. Note that an alternative characterization of a multivariate time series X considers the columns of X to be observations of a p-variate time series with the rows of X containing univariate time series. For example, see Priestley (1981, page 692) and Fuller (1976, page 14).

Let

$$\hat{\mu}_X$$

be the row vector containing the means of the channels of X. In particular,

$$\hat{\mu}_X = (\hat{\mu}_{X_1}, \hat{\mu}_{X_2}, \ldots, \hat{\mu}_{X_p})$$

where for $j = 1, 2, \ldots, p$

$$\hat{\mu}_{X_j} = \begin{cases} \mu_{X_j} & \mu_{X_j} \text{ known} \\ \dfrac{1}{n}\displaystyle\sum_{t=1}^{n} X_{tj} & \mu_{X_j} \text{ unknown} \end{cases}$$

Let

$$\hat{\mu}_Z$$

be similarly defined. In what follows, assume the channels of both X and Z have been centered about their respective means

$$\hat{\mu}_X \text{ and } \hat{\mu}_Z$$

Suppose the desired output is the multichannel time series Z defined by the model

$$Z_{t\bullet} = X_{t\bullet}\phi_0 + X_{(t-1),\bullet}\phi_1 + \ldots + X_{(t-K),\bullet}\phi_K$$

where

$$X_{t\bullet} = (X_{t1}, X_{t2}, \ldots, X_{tp})$$
$$Z_{t\bullet} = (Z_{t1}, Z_{t2}, \ldots, Z_{tp})$$

and ϕ_k is the array of dimension $p \times q$ containing the Wiener filter coefficients

$$\phi_k = \begin{bmatrix} \phi_{11k} & \phi_{12k} & \cdots & \phi_{1qk} \\ \phi_{21k} & \phi_{22k} & \cdots & \phi_{2qk} \\ \vdots & \vdots & & \vdots \\ \phi_{p1k} & \phi_{p2k} & \cdots & \phi_{pqk} \end{bmatrix}$$

for $k = 1, \ldots, K$. The array ϕ_k is the $(k + 1)$-st level of the 3-dimensional array FIL.

The filter coefficients are computed by solving a set of normal equations. The algorithm utilizes the block Toeplitz (or Töplitz) matrix structure of these equations and is given by Robinson (1967, pages 238–246). In particular, the required input consists of the multichannel autocovariance matrices Σ_X, Σ_Z, and the multichannel cross-covariance matrix Σ_{ZX}. The routine MCCF (page 711) may be used to estimate these covariance matrices.

Note that successively longer filters are estimated until either the normalized mean square error is less than EPS or the filter length K = LFIL equals MLFIL. The normalized mean square error is defined by

$$Q_k = 1 - \frac{\text{tr} \sum_{k=0}^{K} \Sigma_{ZX}(k) \phi_k}{\text{tr} \Sigma_Z(0)}$$

where $\text{tr}\Sigma_Z(0)$ = TRACE is the trace of the multichannel autocorrelation coefficient of the desired output at lag zero. The values of Q_k for the successive filters of length $k = 1, 2, \ldots, K$ are contained in ENMS.

KALMN

Performs Kalman filtering and evaluates the likelihood function for the state-space model.

Required Arguments

Y — Vector of length NY containing the observations. (Input)

Z — NY by NB matrix relating the observations to the state vector in the observation equation. (Input)

R — NY by NY matrix such that R * σ^2 is the variance-covariance matrix of errors in the observation equation. (Input)
σ^2 is a positive unknown scalar. Only elements in the upper triangle of R are referenced.

B — Estimated state vector of length NB. (Input/Output)
The input is the estimated state vector at time k given the observations thru time $k - 1$. The output is the estimated state vector at time $k + 1$ given the observations thru time k. On the first call to KALMN, the input B must be the prior mean of the state vector at time 1.

COVB — NB by NB matrix such that COVB * σ^2 is the mean squared error matrix for B. (Input/Output)
Before the first call to KALMN, COVB * σ^2 must equal the variance-covariance matrix of the state vector.

N — Rank of the variance-covariance matrix for all the observations. (Input/Output)
N must be initialized to zero before the first call to KALMN. In the usual case when the variance-covariance matrix is nonsingular, N equals the sum of the NY's from the invocations to KALMN.

SS — Generalized sum of squares. (Input/Output)
SS must be initialized to zero before the first call to KALMN. The estimate of σ^2 is given by SS/N.

ALNDET — Natural log of the product of the nonzero eigenvalues of *P* where *P* * σ^2 is the variance-covariance matrix of the observations. (Input/Output)
Although ALNDET is computed, KALMN avoids the explicit computation of *P*. ALNDET must be initialized to zero before the first call to KALMN. In the usual case when *P* is nonsingular, ALNDET is the natural log of the determinant of *P*.

Optional Arguments

NY — Number of observations for current update. (Input)
If NY = 0, no update is performed.
Default: NY = size (Y,1).

NB — Number of elements in the state vector. (Input)
Default: NB = size (Z,2).

LDZ — Leading dimension of Z exactly as specified in the dimension statement in the calling program. (Input)
Default: LDZ = size (Z,1).

LDR — Leading dimension of R exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

IT — Transition matrix option. (Input)
Default: IT = 1.

IT	Action
0	T is the transition matrix in the state equation.
1	The identity is the transition matrix in the state equation.

T — NB by NB transition matrix in the state equation. (Input, if IT = 0)
If IT = 1, then T is not referenced and can be a vector of length one.

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

IQ — State equation error option. (Input)
Default: IQ = 1.

IQ	Action
0	There is an error term in the state equation.
1	There is no error term in the state equation.

Q — NB by NB matrix such that Q * σ^2 is the variance-covariance matrix of the error vector in the state equation. (Input, if IQ = 0)
σ^2 is a positive unknown scalar. If IQ = 1, then Q is not referenced and can be a 1x1 array. If IQ = 0, only the elements in the upper triangle of Q are referenced.

LDQ — Leading dimension of Q exactly as specified in the dimension statement in the calling program. (Input)
Default: LDQ = size (Q,1).

TOL — Tolerance used in determining linear dependence. (Input)
TOL = 100.0 * AMACH(4) is a common choice. See the documentation for routine AMACH(Reference Material).
Default: TOL = 1.e-5 for single precision and 2.d –14 for double precision.

LDCOVB — Leading dimension of COVB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOVB = size (COVB,1),

V — Vector of length NY containing the one-step-ahead prediction error. (Output)
If Y is not needed, then V and Y can occupy the same storage locations.

COVV — NY by NY matrix such that COVV * σ^2 is the variance-covariance matrix of V. (Output)
If R is not needed, then COVV and R can occupy the same storage locations.

LDCOVV — Leading dimension of COVV exactly as specified in the dimension statement in the calling program. (Input)

FORTRAN 90 Interface

Generic: `CALL KALMN (Y, Z, R, B, COVB, N, SS, ALNDET [,…])`

Specific: The specific interface names are S_KALMN and D_KALMN.

FORTRAN 77 Interface

Single:
```
CALL KALMN (NY, Y, NB, Z, LDZ, R, LDR, IT, T, LDT, IQ, Q,
LDQ, TOL, B, COVB, LDCOVB, N, SS, ALNDET, V, COVV, LDCOVV)
```

Double: The double precision name is DKALMN.

Example 1

Routine KALMN is used to compute the filtered estimates and one-step-ahead estimates for a scalar problem discussed by Harvey (1981, pages 116–117). The observation equation and state equation are given by

$$y_k = b_k + e_k$$
$$b_{k+1} = b_k + w_{k+1} \qquad k = 1, 2, 3, 4$$

where the e_k's are identically and independently distributed normal with mean 0 and variance σ^2, the w_k's are identically and independently distributed normal with mean 0 and variance $4\sigma^2$, and b_1 is distributed normal with mean 4 and variance $16\sigma^2$. Two invocations of KALMN are needed for each time point in order to compute the filtered estimate and the one-step-ahead estimate. The first invocation uses Default: IQ = 1 and IT = 1 so that the prediction equations are skipped in the computations. The second invocation uses NY = 0 so that the update equations are skipped in the computations.

This example also computes the one-step-ahead prediction errors. Harvey (1981, page 117) contains a misprint for the value v_4 that he gives as 1.197. The correct value of v_4 = 1:003 is computed by KALMN.

```
      USE UMACH_INT
      USE KALMN_INT
      INTEGER    LDCOVB, LDCOVV, LDQ, LDR, LDT, LDZ, NB, NOBS, NY
      PARAMETER  (NB=1, NOBS=4, NY=1, LDCOVB=NB, LDCOVV=NY, LDQ=NB, &
                 LDR=NY, LDT=NB, LDZ=NY)
!
      INTEGER    I, IQ, IT, N, NOUT
      REAL       ALNDET, B(NB), COVB(LDCOVB,NB), &
                 COVV(LDCOVV,NY), Q(LDQ,NB), R(LDR,NY), SS, T(LDT,NB), &
                 V(NY), Y(NY), YDATA(NOBS), Z(LDZ,NB)
!
      DATA YDATA/4.4, 4.0, 3.5, 4.6/, Z/1.0/, R/1.0/, Q/4.0/, T/1.0/
!
```

```
      CALL UMACH (2, NOUT)
!                                Initial estimates for state vector
!                                and variance-covariance matrix.
!                                Initialize SS and ALNDET.
      B(1)      = 4.0
      COVB(1,1) = 16.0
      N         = 0
      SS        = 0.0
      ALNDET    = 0.0
      WRITE (NOUT,99998)
!
      DO 10  I=1, NOBS
!                                Update
         Y(1) = YDATA(I)
         CALL KALMN (Y, Z, R, B, COVB, N, SS, ALNDET, T=T, Q=Q, V=V, &
                     COVV=COVV)
         WRITE (NOUT,99999) I, I, B(1), COVB(1,1), N, SS, ALNDET, &
                            V(1), COVV(1,1)
!                                Prediction
         IQ = 0
         IT = 0
         CALL KALMN (Y, Z, R, B, COVB, N, SS, ALNDET, NY=0, IT=IT, T=T, &
                     IQ=IQ, Q=Q, V=V, COVV=COVV)
         WRITE (NOUT,99999) I + 1, I, B(1), COVB(1,1), N, SS, ALNDET, &
                            V(1), COVV(1,1)
   10 CONTINUE
99998 FORMAT (' k/j', '   B    ', '  COVB  ', ' N', '   SS   ', &
             ' ALNDET ', '   V    ', '  COVV  ')
99999 FORMAT (I2, '/', I1, 2F8.3, I2, 4F8.3)
      END
```

Output

```
k/j   B      COVB   N   SS    ALNDET    V      COVV
1/1   4.376   0.941 1   0.009   2.833   0.400  17.000
2/1   4.376   4.941 1   0.009   2.833   0.400  17.000
2/2   4.063   0.832 2   0.033   4.615  -0.376   5.941
3/2   4.063   4.832 2   0.033   4.615  -0.376   5.941
3/3   3.597   0.829 3   0.088   6.378  -0.563   5.832
4/3   3.597   4.829 3   0.088   6.378  -0.563   5.832
4/4   4.428   0.828 4   0.260   8.141   1.003   5.829
5/4   4.428   4.828 4   0.260   8.141   1.003   5.829
```

Comments

1. Workspace may be explicitly provided, if desired, by use of K2LMN/DK2LMN. The reference is:

   ```
   CALL K2LMN (NY, Y, NB, Z, LDZ, R, LDR, IT, T, LDT, IQ, Q, LDQ,
   TOL, B, COVB, LDCOVB, N, SS, ALNDET, V, COVV, LDCOVV, COVVCH,
   WK1, WK2)
   ```

   The additional arguments are as follows.

COVVCH — Work vector of length NY * NY containing the Cholesky factor of the COVV matrix. If R and COVV are not needed, COVVCH, R, and COVV can occupy the same storage locations and LDR must equal LDCOVV.

WK1 — Work vector of length NB * NB.

WK2 — Work vector of length NB * NY + max(NB, NY).

2. Informational errors

Type	Code	
4	1	R + Z * COVB * Z^T is not nonnegative definite within the tolerance defined by TOL. Either TOL is too small, or R or COVB is not nonnegative definite.
4	2	The system of equations $COVVCH^T * x = V$ is inconsistent. The variance-covariance matrix of the observations is inconsistent with the observations input in Y.
4	3	The system of equations $COVVCH^T * x = Z * COVB$ is inconsistent. The Cholesky factorization to compute COVVCH may be based on too large a value for TOL. The input of a smaller value for TOL may be appropriate.

3. If R, Q, and T are known functions of unknown parameters, KALMN can be used in conjunction with routine UMINF (IMSL MATH/LIBRARY) to perform maximum likelihood estimation of these unknown parameters. UMINF should be used to minimize the function

N * ALOG(SS/N) + ALNDET:

4. In order to maintain acceptable numerical accuracy, the double precision version of KALMN is usually required.

Description

Routine KALMN is based on a recursive algorithm given by Kalman (1960), which has come to be known as the Kalman filter. The underlying model is known as the state-space model. The model is specified stage by stage where the stages generally correspond to time points at which the observations become available. The routine KALMN avoids many of the computations and storage requirements that would be necessary if one were to process all the data at the end of each stage in order to estimate the state vector. This is accomplished by using previous computations and retaining in storage only those items essential for processing of future observations.

The notation used here follows that of Sallas and Harville (1981). Let y_k (input in Y) be the $n_k \times 1$ vector of observations that become available at time k. The subscript k is used here rather than t, which is more customary in time series, to emphasize that the model is expressed in stages $k = 1, 2, \ldots$ and that these stages need not correspond to equally spaced time points. In fact, they need not correspond to time points of any kind. The *observation equation* for the state-space model is

$$y_k = Z_k b_k + e_k \qquad k = 1, 2, \ldots$$

Here, Z_k (input in Z) is an $n_k \times q$ known matrix and b_k is the $q \times 1$ state vector. The state vector b_k is allowed to change with time in accordance with the *state equation*

$$b_{k+1} = T_{k+1} b_k + w_{k+1} \quad k = 1, 2, \ldots$$

starting with $b_1 = \mu_1 + w_1$.

The change in the state vector from time k to $k + 1$ is explained in part by the *transition matrix* T_{k+1}(input in T), which is assumed known. It is assumed that the q-dimensional w_k's ($k = 1, 2, \ldots$).are independently distributed multivariate normal with mean vector 0 and variance-covariance matrix $\sigma^2 Q_k$, that the n_k-dimensional e_k's ($k = 1, 2, \ldots$).are independently distributed multivariate normal with mean vector 0 and variance-covariance matrix $\sigma^2 R_k$, and that the w_k's and e_k's are independent of each other. Here, μ_1 is the mean of b_1 and is assumed known, σ^2 is an unknown positive scalar. Q_{k+1}(input in Q) and R_k (input in R) are assumed known.

Denote the estimator of the realization of the state vector b_k given the observations $y_1, y_2, \ldots, y_j$ by

$$\hat{\beta}_{k|j}$$

By definition, the mean squared error matrix for

$$\hat{\beta}_{k|j}$$

is

$$\sigma^2 C_{k|j} = E(\hat{\beta}_{k|j} - b_k)(\hat{\beta}_{k|j} - b_k)^T$$

At the time of the k-th invocation, we have

$$\hat{\beta}_{k|k-1}$$

and $C_{k|k-1}$, which were computed from the $(k-1)$-st invocation, input in B and COVB, respectively. During the k-th invocation, routine KALMN computes the filtered estimate

$$\hat{\beta}_{k|k}$$

along with $C_{k|k}$. These quantities are given by the *update equations*:

$$\hat{\beta}_{k|k} = \hat{\beta}_{k|k-1} + C_{k|k-1} Z_k^T H_k^{-1} v_k$$
$$C_{k|k} = C_{k|k-1} - C_{k|k-1} Z_k^T H_k^{-1} Z_k C_{k|k-1}$$

where

$$v_k = y_k - Z_k \hat{\beta}_{k|k-1}$$

and where

$$H_k = R_k + Z_k C_{k|k-1} Z_k^T$$

Here, v_k (stored in V) is the one-step-ahead prediction error, and $\sigma^2 H_k$ is the variance-covariance matrix for v_k. H_k is stored in COVV. The "start-up values" needed on the first invocation of KALMN are

$$\hat{\beta}_{1|0} = \mu_1$$

and $C_{1|0} = Q_1$ input via B and COVB, respectively. Computations for the k-th invocation are completed by KALMN computing the one-step-ahead estimate

$$\hat{\beta}_{k+1|k}$$

along with $C_{k+1|k}$ given by the *prediction equations*:

$$\hat{\beta}_{k+1|k} = T_{k+1} \hat{\beta}_{k|k}$$
$$C_{k+1|k} = T_{k+1} C_{k|k} T_{k+1}^T + Q_{k+1}$$

If both the filtered estimates and one-step-ahead estimates are needed by the user at each time point, KALMN can be invoked twice for each time point—first with IT = 1 and IQ = 1 to produce

$$\hat{\beta}_{k|k}$$

and $C_{k|k}$, and second with NY = 0 to produce

$$\hat{\beta}_{k+1|k}$$

and $C_{k+1|k}$ (With IT = 1 and IQ = 1, the prediction equations are skipped. With NY = 0, the update equations are skipped.)

Often, one desires the estimate of the state vector more than one-step-ahead, i.e., an estimate of

$$\hat{\beta}_{k|j}$$

is needed where $k > j + 1$. At time j, KALMN is invoked to compute

$$\hat{\beta}_{j+1|j}$$

Subsequent invocations of KALMN with NY = 0 can compute

$$\hat{\beta}_{j+2|j}, \hat{\beta}_{j+3|j}, \ldots, \hat{\beta}_{k|j}$$

Computations for

$$\hat{\beta}_{k|j}$$

and $C_{k|j}$ assume the variance-covariance matrices of the errors in the observation equation and state equation are known up to an unknown positive scalar multiplier, σ^2. The maximum likelihood estimate of σ^2 based on the observations $y_1, y_2, \ldots, y_m$, is given by

$$\hat{\sigma}^2 = SS / N$$

where

$$N = \sum_{k=1}^{m} n_k \text{ and } SS = \sum_{k=1}^{m} v_k^T H_k^{-1} v_k$$

If σ^2 is known, the R_k's and Q_k's can be input as the variance-covariance matrices exactly. The earlier discussion is then simplified by letting $\sigma^2 = 1$.

In practice, the matrices T_k, Q_k, and R_k are generally not completely known. They may be known functions of an unknown parameter vector θ. In this case, KALMN can be used in conjunction with an optimization program (see routine UMINF, (IMSL MATH/LIBRARY) to obtain a maximum likelihood estimate of θ. The natural logarithm of the likelihood function for $y_1, y_2, \ldots, y_m$ differs by no more than an additive constant from

$$L(\theta, \sigma^2; y_1, y_2, \ldots, y_m) = -\frac{1}{2} N \ln \sigma^2 - \frac{1}{2} \sum_{k=1}^{m} \ln[\det(H_k)] - \frac{1}{2} \sigma^{-2} \sum_{k=1}^{m} v_k^T H_k^{-1} v_k$$

(Harvey 1981, page 14, equation 2.21). Here,

$$\sum_{k=1}^{m} \ln[\det(H_k)]$$

(stored in ALNDET) is the natural logarithm of the determinant of V where $\sigma^2 V$ is the variance-covariance matrix of the observations.

Minimization of $-2L(\theta, \sigma^2; y_1, y_2, \ldots, y_m)$ over all θ and σ^2 produces maximum likelihood estimates. Equivalently, minimization of $-2L_c(\theta; y_1, y_2, \ldots, y_m)$ where

$$L_c(\theta; y_1, y_2, \ldots, y_m) = -\frac{1}{2} N \ln\left(\frac{SS}{N}\right) - \frac{1}{2} \sum_{k=1}^{m} \ln[\det(H_k)]$$

produces maximum likelihood estimates

$$\hat{\theta} \text{ and } \hat{\sigma}^2 = SS / N$$

The minimization of $-2L_c(\theta; y_1, y_2, \ldots, y_m)$ instead of $-2L(\theta, \sigma^2; y_1, y_2, \ldots, y_m)$, reduces the dimension of the minimization problem by one. The two optimization problems are equivalent since

$$\hat{\sigma}^2(\theta) = SS(\theta) / N$$

minimizes $-2L(\theta, \sigma^2; y_1, y_2, \ldots, y_m)$ for all θ, consequently,

$$\hat{\sigma}^2(\theta)$$

can be substituted for σ^2 in $L(\theta, \sigma^2; y_1, y_2, \ldots, y_m)$ to give a function that differs by no more than an additive constant from $L_c(\theta; y_1, y_2, \ldots, y_m)$.

The earlier discussion assumed H_k to be nonsingular. If H_k is singular, a modification for singular distributions described by Rao (1973, pages 527–528) is used. The necessary changes in the preceding discussion are as follows:

1. Replace

$$H_k^{-1}$$

by a generalized inverse.

2. Replace $\det(H_k)$ by the product of the nonzero eigenvalues of H_k.

3. Replace N by

$$\sum_{k=1}^{m} \text{rank}\left(H_k\right)$$

Maximum likelihood estimation of parameters in the Kalman filter is discussed by Sallas and Harville (1988) and Harvey (1981, pages 111–113).

Example 2

Routine KALMN is used with routine UMINF (IMSL MATH/LIBRARY) to find a maximum likelihood estimate of the parameter θ in a MA(1) time series represented by $y_k = \varepsilon_k - \theta\varepsilon_{k-1}$. Routine RNARM (see the Reference Material section of this manual) is used to generate 200 random observations from an MA(1) time series with $\theta = 0.5$ and $\sigma^2 = 1$.

The MA(1) time series is cast as a state-space model of the following form (see Harvey 1981, pages 103–104, 112):

$$y_k = \begin{pmatrix} 1 & 0 \end{pmatrix} b_k$$

$$b_k = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} b_{k-1} + w_k$$

where the two-dimensional w_k's are independently distributed bivariate normal with mean 0 and variance $\sigma^2 Q_k$ and

$$Q_1 = \begin{pmatrix} 1+\theta^2 & -\theta \\ -\theta & \theta^2 \end{pmatrix}$$

$$Q_k = \begin{pmatrix} 1 & -\theta \\ -\theta & \theta^2 \end{pmatrix} \qquad k = 2, 3, \ldots, 200$$

The warning error that is printed as part of the output is not serious and indicates that UMINF is generally used for multi-parameter minimization.

```
      USE RNSET_INT
      USE RNARM_INT
      USE UMINF_INT
      USE UMACH_INT
      INTEGER    NOBS, NTHETA
      PARAMETER  (NOBS=200, NTHETA=1)
!
      INTEGER    IADIST, IPARAM(7), ISEED, LAGAR(1), LAGMA(1), NOUT, &
```

```
                NPAR, NPMA
      REAL       A(NOBS+1), AVAR, CNST, FSCALE, FVALUE, PAR(1), &
                PMA(1), RPARAM(7), THETA(NTHETA), WI(1), XGUESS(1), &
                XSCALE(1), YDATA(NOBS)
      COMMON     /MA1/ YDATA
      EXTERNAL   FCN
!
      ISEED = 123457
      CALL RNSET (ISEED)
      PMA(1)   = 0.5
      LAGMA(1) = 1
      CNST    = 0.0
      NPAR     = 0
      NPMA     = 1
      IADIST   = 0
      AVAR     = 1.0
      CALL RNARM (CNST, PAR, LAGAR, PMA, LAGMA, &
                  IADIST, AVAR, A, WI, YDATA, NPAR=NPAR)
!                                 Use UMINF to find maximum likelihood
!                                 estimate of the MA parameter THETA.
      CALL UMINF (FCN, THETA)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' '
      WRITE (NOUT,*) '* * * Final Estimate for THETA * * *'
      WRITE (NOUT,*) 'Maximum likelihood estimate, THETA = ', THETA(1)
      END
!                                Use KALMN to evaluate the likelihood.
      SUBROUTINE FCN (NTHETA, THETA, FUNC)
      USE KALMN_INT
      INTEGER    NTHETA
      REAL       THETA(NTHETA), FUNC
!
      INTEGER    LDCOVB, LDCOVV, LDQ, LDR, LDT, LDZ, NB, NOBS, NY
      PARAMETER  (NB=2, NOBS=200, NY=1, LDCOVB=NB, LDCOVV=NY, LDQ=NB, &
                LDR=NY, LDT=NB, LDZ=NY)
!
      INTEGER    I, IQ, IT, N
      REAL       ABS, ALNDET, ALOG, B(NB), COVB(LDCOVB,NB), &
                COVV(LDCOVV,NY), Q(LDQ,NB), R(LDR,NY), SS, T(LDT,NB), &
                TOL, V(NY), Y(NY), YDATA(NOBS), Z(LDZ,NB)
      COMMON     /MA1/ YDATA
      INTRINSIC  ABS, ALOG
!
      DATA T/0.0, 0.0, 1.0, 0.0/, Z/1.0, 0.0/
!
      IF (ABS(THETA(1)) .GT. 1.0) THEN
!                                 Estimate out of parameter space.
!                                 Set function to a large number.
         FUNC = 1.E10
         RETURN
      END IF
      IQ     = 0
      Q(1,1) = 1.0
      Q(1,2) = -THETA(1)
      Q(2,1) = -THETA(1)
```

```
      Q(2,2) = THETA(1)**2
      IT     = 0
!                                  No error in the
!                                  observation equation.
      R(1,1) = 0.0
!                                  Initial estimates for state vector
!                                  and variance-covariance matrix.
!                                  Initialize SS and ALNDET.
      B(1)      = 0.0
      B(2)      = 0.0
      COVB(1,1) = 1.0 + THETA(1)**2
      COVB(1,2) = -THETA(1)
      COVB(2,1) = -THETA(1)
      COVB(2,2) = THETA(1)**2
      N         = 0
      SS        = 0.0
      ALNDET    = 0.0
!
      DO 10  I=1, NOBS
         Y(1) = YDATA(I)
         CALL KALMN (Y, Z, R, B, COVB, N, SS, ALNDET, IT=IT, T=T, &
                     IQ=IQ, Q=Q)
   10 CONTINUE
      FUNC = N*ALOG(SS/N) + ALNDET
      RETURN
      END
```

Output

```
*** WARNING  ERROR 1 from U5INF.  This routine may be inefficient for a
***          problem of size N = 1.
  Here is a traceback of subprogram calls in reverse order:
  Routine name                    Error type  Error code
  ------------                    ----------  ----------
   U5INF                               6           1    (Called internally)
   U3INF                               0           0    (Called internally)
   U2INF                               0           0    (Called internally)
   UMINF                               0           0
   USER                                0           0
* * * Final Estimate for THETA * * *
Maximum likelihood estimate, THETA =    0.452944
```

AUTO_UNI_AR

Automatic selection and fitting of a univariate autoregressive time series model. The lag for the model is automatically selected using Akaike's Information Criterion (AIC). Estimates of the autoregressive parameters for the model with minimum AIC are calculated using method of moments or maximum likelihood.

Required Arguments

W — Vector containing the stationary time series. (Input)

MAXLAG — Maximum number of autoregressive parameters requested. (Input)

NPAR — Number of autoregressive parameters. (Output)

PAR — Vector of length `MAXLAG`, which contains the estimates for the autoregressive parameters in the model with the minimum `AIC`. The estimates are in the first `NPAR` values of this vector. (Output)

Optional Arguments

IPRINT — Printing option. (Input)

0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: `IPRINT` = 0.

IMETH — Estimation method option. (Input)

0	Method of moments
1	Maximum likelihood

Default: `IMETH` = 0.

MAXIT — Maximum number of estimation iterations. (Input)
Default: `MAXIT` = 500.

AVAR — Innovation variance. (Output)

AIC — Akaike's Information Criterion. (Output)

FORTRAN 90 Interface

Generic: `CALL AUTO_UNI_AR (W, MAXLAG, NPAR, PAR [,…])`

Specific: The specific interface names are `S_AUTO_UNI_AR` and `D_AUTO_UNI_AR`.

Example

Consider the Wolfer Sunspot Data (Box and Jenkins 1976, page 530) consisting of the number of sunspots observed for each year from 1770 through 1869. In this example, `AUTO_UNI_AR` found the minimum AIC fit is an autoregressive model with 10 lags:

$$w_t = \phi_0 + \phi_1 w_{t-1} + \ldots + \phi_{10} w_{t-10} + a_t$$

using the formula

$$\phi_0 = \mu\left(1 - \sum_{i=1}^{NPAR} \phi_i\right),$$

the lag 10 AR model for this series can be represented as:

$$w_t = 0.85\ M + 1.24\ w_{t-1} - 0.50\, w_{t-2} - 0.16\ w_{t-3}$$
$$+0.23\, w_{t-4} \quad - 0.20 w_{t-5} + 0.11 w_{t-6}$$
$$-0.08 w_{t-7} + 0.09 w_{t-8} + 0.01 w_{t-9} + 0.10 w_{t-10} + a_t$$

```
      use auto_uni_ar_int
      use wrrrn_int
      use gdata_int
      implicit none
!                                     SPECIFICATIONS FOR PARAMETERS
      integer, parameter   :: maxlag=20
      integer              :: npar
      real(kind(1e0))      :: aic, avar
      real(kind(1e0))      :: par(maxlag)
      real(kind(1e0))      :: x(176,2)
      integer              :: ncol, nrow
!                                     SPECIFICATIONS FOR LOCAL VARIABLES
      integer              :: nout
!
      call umach (2, nout)
      write(nout,*) 'AIC Automatic Order selection '
      write(nout,*) 'AR coefficients estimated using Maximum Likelihood'
      write(nout,*)
!                                     Get Wolfer Sunspot Data
      call gdata(2,x,nrow,ncol)
!                                     Example #1
      call auto_uni_ar(x(22:,2), maxlag, npar, par, aic=aic, imeth=1,&
                       avar=avar)
      write(nout,*) 'Order Selected: ', npar
      write(nout,*) 'AIC = ', aic,'      Variance = ', avar
      call wrrrn('Final AR Coefficients estimated by Method of Moments', &
                 par, nra=npar, nca=1, lda=npar)
      end
```

Output

```
AIC Automatic Order selection
AR coefficients estimated using Maximum Likelihood

 Order Selected:  10
 AIC =  958.0346       Variance =  211.70743

 Final AR Coefficients estimated by Method of Moments
                       1   1.243
                       2  -0.503
                       3  -0.158
                       4   0.230
                       5  -0.200
                       6   0.114
                       7  -0.079
                       8   0.094
                       9   0.010
                      10   0.096
```

Description

The routine AUTO_UNI_AR automatically selects the order of the AR model that best fits the data and then computes the AR coefficients. The algorithm used in AUTO_UNI_AR is derived from the work of Akaike, H., et. al (1979) and Kitagawa & Akaike (1978). This code was adapted from the UNIMAR procedure published as part of the TIMSAC-78 Library.

The best-fit AR model is determined by successively fitting AR models with 1, 2, ..., MAXLAG autoregressive coefficients. For each model, Akaike's Information Criterion (AIC) is calculated based on the formula:

$$AIC = -2\ln(likelihood) + 2(NPAR) .$$

AUTO_UNI_AR uses the approximation to this formula developed by Ozaki and Oda (1979).

$$AIC = (NOBS - MAXLAG)\ln(\hat{\sigma}^2) + 2(NPAR) + (NOBS - MAXLAG)(\ln(2\pi) + 1)$$

The best fit model is the model with minimum AIC. If the number of parameters in this model is equal to the highest order autoregressive model fitted, i.e., NPAR=MAXLAG, then a model with smaller AIC might exist for larger values of MAXLAG. In this case, increasing MAXLAG to explore AR models with additional autoregressive parameters might be warranted.

If IMETH = 0 estimates of the autoregressive coefficients for the model with the minimum AIC are calculated using method of moments. Otherwise, if IMETH =1, the model with the minimum AIC is identified and coefficients are then estimated using maximum likelihood.

AUTO_FPE_UNI_AR

Automatic selection and fitting of a univariate autoregressive time series model using Akaike's Final Prediction Error (FPE) criteria. Estimates of the autoregressive parameters for the model with minimum FPE are calculated using the methodology described in Akaike, H., et. al (1979).

Required Arguments

MAXLAG — Maximum lag of the sample autocovariances for the stationary time series, *W*. (Input)

ACV — Vector of length MAXLAG + 1 containing the sample autocovariances of *W*. The first element, ACV(0) must be the sample variance of the series and the remaining elements, ACV(1), …, ACV(MAXLAG), contain the autocovariances of the series for lags 1 through MAXLAG. (Input)

NOBS — Number of observations in the time series. (Input)

NPAR — Number of autoregressive parameters in the the selected model. (Output)

PAR — Vector of length MAXLAG containing estimates for the autoregressive parameters in the model with the minimum Final Prediction Error. The estimates are in the first NPAR values of this vector. The remaining values are set to 0. (Output)

Optional Arguments

IPRINT — Printing option. (Input)

0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: `IPRINT` = 0.

FPE — Final Prediction Error for fitted model. (Output)

CHISQ — Chi-square statistic, with 1 degree of freedom, for the selected model. `CHISQ` is used to examine the significance of the fitted model. (Output)

AVAR — Estimate of noise varaince. (Output)

FORTRAN 90 Interface

Generic: `CALL AUTO_FPE_UNI_AR (MAXLAG, ACV, NOBS, NPAR, PAR [,…])`

Specific: The specific interface names are `S_AUTO_FPE_UNI_AR` and `D_AUTO_FPE_UNI_AR`.

Example

Consider the Wolfer Sunspot Data (Box and Jenkins 1976, page 530) consisting of the number of sunspots observed each year from 1770 through 1869. In this example, `AUTO_FPE_UNI_AR`, found the minimum FPE solution is an autoregressive model with 10 lags. This is slightly different from the optimum solution found by `AUTO_UNI_AR`, using minimum AIC instead of FPE.

The solution reported by `AUTO_UNI_AR` is an AR model with 2 lags.

$$w_t = \phi_0 + \phi_1 w_{t-1} + \phi_2 w_{t-2} + a_t$$

Using the formula

$$\phi_0 = \mu\left(1 - \sum_{i=1}^{NPAR} \phi_i\right),$$

we obtain the following reprensentation for this series.

$$w_t = 0.32\mu + 1.318 w_{t-1} - 0.635 w_{t-2} + a_t \,.$$

```
      use auto_fpe_uni_ar_int
      use wrrrn_int
      use gdata_int
      use acf_int
      implicit none
!                                  SPECIFICATIONS FOR PARAMETERS
```

```
      integer, parameter   :: maxlag=20, nobs=100
      integer              :: npar
      real(kind(1e0))      :: fpe, chisq
      real(kind(1e0))      :: ac(maxlag+1), avar
      real(kind(1e0))      :: acv(maxlag+1), par(maxlag)
      real(kind(1e0))      :: x(176,2)
      integer              :: ncol, nrow
!
!                                  Get Wolfer Sunspot Data
      call gdata(2,x,nrow,ncol)
!
      npar = 20
      write(*,*) 'Univariate FPE Automatic Order selection '
      write(*,*) ' '
!                                  Compute the autocovariances
      call acf (x(22:,2), maxlag, ac, acv=acv, nobs=nobs)
!                                  Example #1
      call auto_fpe_uni_ar(maxlag, acv, nobs, npar, par,  &
                           fpe=fpe, chisq=chisq, avar=avar)
!
      write(*,*) 'Minimum FPE = ', fpe
      write(*,*) 'Chi-squared = ', chisq
      write(*,*) 'Avar = ', avar
      call wrrrn('AR Coefficients', par, nra=npar, nca=1, lda=npar)
!
      end
```

Output

```
 Univariate FPE Automatic Order selection

 Minimum FPE =  306.91787
 Chi-squared =  39.056877
 Avar = 289.03915

 AR Coefficients
    1   1.318
    2  -0.635
```

Description

This routine is based upon the FPEAUT program published in the TIMSAC –71 Library described by Akaike, H. and Nakagawa, T (1972).

The Final Prediction Error for an autoregressive model with lag *k* is defined as:

$$FPE_{\kappa} = \left[1 + \frac{k+1}{N}\right] \cdot r_k ,$$

where N = NOBS and r_k is the minimum of

$$r_k = E\left[W_t - \sum_{m=1}^{k} \phi_m^{(k)} W_{t-m} - \phi_0^{(k)} \right]^2$$

with respect to the autoregressive coefficients

$$\left\{ \phi_m^{(k)}; m = 0, 1, \cdots, k \right\}.$$

$FPE_\kappa = \left[1 + \frac{k+1}{N} \right] \cdot r_k$ is estimated using the formula:

$$FPE_k = \frac{(N + (k+1))}{(N - (k+1))} \cdot \sigma_k^2,$$

where σ_k^2 is calculated from the recursive relationship:

$$\sigma_k^2 = \sigma_{k-1}^2 (1 - (\hat{\phi}_k \cdot k)^2), \text{ where } \hat{\phi}_0^2 = 0 \text{ and } \sigma_0^2 = \tfrac{1}{N} \sum_{t=1}^{N} (W_t - \bar{W})^2.$$

The model selected and parameter estimates vary depending upon the value of MAXLAG. Akaike and Nakagawa (1972) recommend that MAXLAG start with values between $2\sqrt{N}$ and $3\sqrt{N}$.

In every case, however, MAXLAG must be strictly less than $N/2$.

As MAXLAG is increased numerical accuracy decreases. It is even possible numerically for the estimated FPE to become negative. If this happens, use double precision.

AUTO_MUL_AR

Automatic selection and fitting of a multivariate autoregressive time series model. This is the multivariate version of AUTO_UNI_AR. The lag for the model is automatically selected using Akaike's Information Criterion (AIC).

Required Arguments

X — NOBS by NCHANX matrix containing the stationary multivariate time series. (Input)
Each row of X corresponds to an observation of a multivariate time series, and each column of X corresponds to a univariate time series.

MAXLAG — Maximum number of autoregressive parameters requested. (Input)

NMATRIX — Number of autoregressive parameter matrices in the minimum AIC model. (Output)

PAR — Three dimensional NMATRIX by NCHANX by NCHANX array containing estimates for the autoregressive parameters in the model. (Output)

Optional Arguments

NOBS — Number of observations in each time series.
Default: NOBS = size(X,1) (Input)

NCHANX — Number of variables, channels, in the multivariate time series. (Input)
NCHANX is the number of columns in X that should be processed.
Default: NCHANX = size(X,2)

IPRINT — Printing option. (Input)

0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: IPRINT = 0

AVAR — NCHANX by NCHANX array of estimates of the noise variance. (Output)

NPAR — Vector of length NCHANX containing the number of autoregressive parameters fitted for each time series. (Output)

AIC — Akaike's Information Criterion . (Output)

FORTRAN 90 Interface

Generic: CALL AUTO_MUL_AR (X, MAXLAG, NMATRIX, PAR [,…])

Specific: The specific interface names are S_AUTO_MUL_AR and D_AUTO_MUL_AR.

Example

Consider the Wolfer Sunspot Data (Box and Jenkins 1976, page 530) consisting of the number of sunspots observed each year, along with data in northern light activity and earthquakes in Robinson 1967, page 204, for the years 1770 through 1869.

```
USE GDATA_INT
USE AUTO_MUL_AR_INT
USE WRRRN_INT

IMPLICIT NONE
INTEGER, PARAMETER :: NOBS=100, MAXLAG=10
INTEGER NCOL, NROW, NMATRIX, I
REAL(KIND(1E0)) RDATA(NOBS,4), X(NOBS,3), PAR(MAXLAG,3,3)

CALL GDATA (8, RDATA, NROW, NCOL)
X(:,1) = Rdata(:,3)
X(:,2) = Rdata(:,4)
X(:,3) = Rdata(:,2)
```

```
    CALL AUTO_MUL_AR(X,MAXLAG,NMATRIX,PAR)
      WRITE(*,*) "PAR = "
      DO I=1,NMATRIX
      CALL WRRRN("",PAR(I,:,:))
         ENDDO
    END
```

Output

```
PAR =

        1       2       3
1   0.822   0.000   0.000
2   0.000   0.000  -0.290
3   0.098   0.000   1.214

         1       2       3
1  -0.1136  0.0000  0.0000
2   0.2489  0.0000  0.0000
3   0.0040  0.0000 -0.8682

         1       2       3
1   0.0000  0.0000  0.0000
2   0.0000  0.0000  0.0000
3   0.0725  0.0000  0.3374

         1       2       3
1   0.0000  0.0000  0.0000
2   0.0000  0.0000  0.0000
3  -0.0693  0.0400 -0.2217

         1       2       3
1   0.0000  0.0000  0.0000
2   0.0000  0.0000  0.0000
3   0.0745 -0.0025  0.1253

         1       2       3
1   0.0000  0.0000  0.0000
2   0.0000  0.0000  0.0000
3  -0.0299  0.0510 -0.1570

          1        2        3
1   0.00000  0.00000  0.00000
2   0.00000  0.00000  0.00000
3   0.00806  0.05025  0.04558

          1        2        3
1   0.00000  0.00000  0.00000
2   0.00000  0.00000  0.00000
3   0.03858  0.01929 -0.09825

         1       2       3
1   0.0000  0.0000  0.0000
2   0.0000  0.0000  0.0000
3   0.0904 -0.0154  0.1376
```

```
          1          2          3
1   0.00000    0.00000    0.00000
2   0.00000    0.00000    0.00000
3   0.00000   -0.06879    0.00000
```

Description

The routine AUTO_MUL_AR automatically selects the order of the multivariate AR model that best fits the data and then displays the AR coefficients. This procedure is an adaptation of the MULMAR procedure published in the TIMSAC-78 Library by Akaike, H., et. al (1979) and Kitagawa & Akaike (1978).

A multivariate AR model can be expressed as:

$$X_t = \sum_{i=1}^{NMATRIX} A_i \cdot X_{t-i} + U_t,$$

where

$$X_t = (x_{1,t}, x_{2,t}, \ldots, x_{NCHANX,t})'$$

is a column vector containing the values for the NCHANX univariate time series at time = *t*.

$$A_1, A_2, \ldots, A_{NPAR}$$

are the NCHANX by NCHANX matrices containing the autoregressive parameter estimates for lags 1, 2, …, NMATRIX. And

$$U_t = (\varepsilon_{1,t}, \varepsilon_{2,t}, \ldots, \varepsilon_{NCHANX,t})'$$

is a column vector containing the values for the NCHANX white noise values for each time series at time = *t*.

The best fit AR model is determined by successfully fitting multivariate AR models with 1 to NMATRIX autoregressive coefficients. For each model, Akaike's Information Criterion (AIC) is calculated using the formula:

$$AIC = (NOBS - MAXLAG) \cdot \ln(\det(AVAR)) + 2 \cdot K$$
$$+ (NOBS - MAXLAG)(\ln(2\pi) + 1)$$

where

K = number of non-zero autoregressive coefficients in the parameter matrices.

The best fit model is the model with the minimum AIC. If the number of parameters in this model is equal to the highest order autoregressive model fitted, i.e., NMATRIX=MAXLAG, then a model with smaller AIC might exist for larger values of MAXLAG. In this case, increasing MAXLAG to explore AR models with additional autoregressive parameters might be warranted.

AUTO_FPE_MUL_AR

Automatic selection and fitting of a multivariate autoregressive time series model using Akaike's Multivariate Final Prediction Error (MFPE) criteria.

Required Arguments

X — NOBS by NCHANX matrix containing the stationary time series. Each row corresponds to an observation in the time series, and each column corresponds to a univariate time series for one of the channels. (Input)

MAXLAG — Maximum number of autoregressive parameters requested. (Input)

NMATRIX — Number of autoregressive parameter matrices in minimum FPE model. (Output)

PAR — MAXLAG by NCHANX by NCHANX array containing estimates for the autoregressive parameters in the selected model. (Output)

Optional Arguments

IPRINT — Printing option. (Input)

0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: IPRINT = 0

NOBS — Number of observations in each time series. (Input)
Default = size(X,1).

NCHANX — Number of variables, channels, in the multivariate time series. (Input)
Default = size(X,2).

CCV — NCHANX by NCHANX by MAXLAG+1 matrix containing the sample crosscovariances of the NCHANX time series variables. For the *i*-th time series variable, the first element, CCV(i,i, 1) must be the sample variance for the *i*-th series and the remaining elements, CCV(,i,i, 2) … CCV(i,i, MAXLAG+1), contain the autocovariances of the *i*-th series for lags 1 thru MAXLAG. Elements CCV(,i,j, 2), …, CCV(i,j, MAXLAG+1,), contain the autocovariances between the *i*-th and *j*-th series for lags 1 thru MAXLAG (Input)

AVAR — NCHANX by NCHANX matrix containing estimates of the noise variance for each of the NCHANX time series. (Output)

FPE — Multivariate Final Prediction Error for fitted model. (Output)

FORTRAN 90 Interface

Generic: `CALL AUTO_FPE_MUL_AR (X, MAXLAG, NMATRIX, PAR [,…])`

Specific: The specific interface names are `S_AUTO_FPE_MUL_AR` and `D_AUTO_FPE_MUL_AR`.

Example

Consider the Wolfer Sunspot Data (Box and Jenkins 1976, page 530) along with data on northern light and earthquake activity (Robinson 1967, page 204) for each year from 1770 through 1869 to be a 3-channel time series. The following program automatically fits this data to an AR model with a `MAXLAG` = 10. In this example, `AUTO_FPE_MUL_AR` selects a multivariate autoregressive model with 2 lags.

```
      use auto_fpe_mul_ar_int
      use wrrrl_int
      use gdata_int
      implicit none
!                                  SPECIFICATIONS FOR PARAMETERS
      integer,parameter     :: nobs=100, nvar=3, maxlag=10
      integer               :: nrow, ncol, nmatrix
      real(kind(1e0))       :: ccv(maxlag+1,nvar,nvar)
      real(kind(1e0))       :: x(nobs,nvar+1)
      real(kind(1e0))       :: par(maxlag, nvar, nvar)
      real(kind(1e0))       :: avar(nvar, nvar), aic, fpe

!                                  SPECIFICATIONS FOR LOCAL VARIABLES
      character             :: label(1)*4
      integer               :: nout, i
!                                  EXAMPLE #1
      label(1) = 'NONE'
!                                  GET ROBINSON MULTICHANNEL DATA
      call gdata( 8, x, nrow, ncol)
!
      call auto_fpe_mul_ar(x(1:,2:), maxlag, nmatrix, par,  aic=aic, &
           fpe=fpe, avar=avar)
!
      write(nout,*) 'Order Selected: ', nmatrix
      write(nout,*) 'FPE = ', fpe
      do i = 1, npar
         call wrrrl('PAR ', par(i, 1:nvar, 1:nvar), label, &
                 label, fmt='(F15.10)')
      enddo
      write(*,*) ' '
      call wrrrl('AVAR (White Noise)', avar, label, label,  &
                  fmt='(F15.10)')
      end
```

Output

```
Order Selected:  2
FPE =  825727500.0
                      PAR
   1.2989480495     0.0272710621     0.0238259397
   0.0364407972     0.7911528349     0.0851913393
  -0.2183818072    -0.0601412356    -0.0717520714
                      PAR
  -0.6405253410     0.0186619535    -0.0319643840
   0.0082284175    -0.1621686369     0.0587148108
  -0.1242173612     0.3418768048    -0.0425752103

               AVAR (White Noise)
 281.7733154297   231.8975372314    57.6690979004
 231.8975982666  1296.4528808594    70.7170333862
  57.6690635681    70.7170333862  1752.6368408203
```

Description

The AUTO_FPE_MUL_AR routine is based upon the FPEC program published in the TIMSAC –71 Library described by Akaike, H. and Nakagawa, T (1972). Estimates of the autoregressive parameters for the model with minimum FPE are calculated using the methodology described in Akaike, H., et. al (1979).

The multivariate Final Prediction Error for a multivariate autoregressive model with lag p is defined as:

$$MFPE_p = \frac{\left(1+\frac{pm+1}{N}\right)^m}{\left(1-\frac{pm+1}{N}\right)^m}\left\|\hat{\Sigma}_p\right\|$$

where $\left\|\hat{\Sigma}_p\right\|$ is the determinant of the estimated NCHANX by NCHANX matrix of covariances of the white noise in the NCHANX series, m = NCHANX and N = NOBS.

The model selected and parameter estimates vary depending upon the value of MAXLAG. Akaike and Nakagawa (1972) recommend that MAXLAG start with values between $2\sqrt{N}/m$ and $3\sqrt{N}/m$, and that MAXLAG does not exceed $N/(5\cdot m)$.

In every case, however, MAXLAG must never exceed $N/(2\cdot m)$.

The numerical accuracy decreases as MAXLAG increases. In this case, it is possible for the estimated FPE to become negative. If this happens , using double percision may help.

BAY_SEA

Bayesian seasonal adjustment modeling. The model allows for a decomposition of a time series into trend, seasonal, and an error component.

Required Arguments

W — Vector containing the stationary time series. (Input)

Optional Arguments

IORDER — Order of trend differencing. (Input)
Default: IORDER = 2.

ISORDER — Order of seasonal differencing. (Input)
Default: ISORDER = 1.

NFOCAST — Number of forecasted values. (Input)
Default: NFOCAST = 0.

NPERIOD — Number of seasonals within a period. (Input)
Default: NPERIOD = 12.

RIGID — Controls rigidity of the seasonal pattern. (Input)
Default: RIGID = 1.0.

LOGT — Model option. (Input)

0	Non-additive model
1	Log additive model

Default: LOGT = 0.

IPRINT — Printing option. (Input)

0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: IPRINT = 0.

ABIC — The Akaike Bayesian Information Criterion for the estimated model. (Output)

TREND — Vector of length size (W) + NFOCAST containing the estimated trend component for each data value followed by the trend estimates for the NFOCAST forecasted values. (Output)

SEASONAL — Vector of length size (W) + NFOCAST containing the estimated seasonal components for each data value followed by the estimates for the NFOCAST forecasted seasonal values. (Output)

COMP — Vector of length size (W) containing the estimated irregular components. (Output)

SMOOTHED — Vector of length size (W) + NFOCAST containing the estimated smoothed estimates for each of the time series values followed by the NFOCAST forecast values. (Output)

FORTRAN 90 Interface

Generic: `CALL BAY_SEA (W [,…])`

Specific: The specific interface names are S_BAY_SEA and D_BAY_SEA.

Example

This example uses unadjusted unemployment for women over 20 years of age in the U.S. for 1991-2001, as reported by the U.S. Bureau of Labor Statistics (www.bls.gov). Figure 1 displays this data together with the smoothed preditions from BAY_SEA. Figure 2 displays the same data with trend predictions from BAY_SEA.

```
      use bay_sea_int
      use wrrrl_int
!                                SPECIFICATIONS FOR PARAMETERS
      integer, parameter :: focast=12, nobs=132, nyears=11
      real(kind(1e0))           :: b(nobs)
!                                U.S. Labor Statistics
!                                unemployment for women
!                                over 20 years of age
      data b/ 2968D0,3009D0,2962D0,2774D0,3040D0,3165D0,& !1991
              3104D0,3313D0,3178D0,3142D0,3129D0,3107D0,&
              3397D0,3447D0,3328D0,3229D0,3286D0,3577D0,& !1992
              3799D0,3867D0,3655D0,3360D0,3310D0,3369D0,&
              3643D0,3419D0,3108D0,3118D0,3146D0,3385D0,& !1993
              3458D0,3468D0,3330D0,3244D0,3135D0,3005D0,&
              3462D0,3272D0,3275D0,2938D0,2894D0,3106D0,& !1994
              3150D0,3289D0,3136D0,2829D0,2776D0,2467D0,&
              2944D0,2787D0,2749D0,2762D0,2578D0,2900D0,& !1995
              3100D0,3102D0,2934D0,2864D0,2652D0,2456D0,&
              3088D0,2774D0,2701D0,2555D0,2677D0,2741D0,& !1996
              3052D0,2966D0,2772D0,2723D0,2705D0,2640D0,&
              2898D0,2788D0,2718D0,2406D0,2520D0,2645D0,& !1997
              2708D0,2811D0,2666D0,2380D0,2292D0,2187D0,&
              2750D0,2595D0,2554D0,2213D0,2218D0,2449D0,& !1998
              2532D0,2639D0,2449D0,2326D0,2302D0,2065D0,&
              2447D0,2398D0,2381D0,2250D0,2086D0,2397D0,& !1999
              2573D0,2475D0,2299D0,2054D0,2127D0,1935D0,&
              2425D0,2245D0,2298D0,2005D0,2208D0,2379D0,& !2000
              2459D0,2539D0,2182D0,1959D0,2012D0,1834D0,&
              2404D0,2329D0,2285D0,2175D0,2245D0,2492D0,& !2001
              2636D0,2892D0,2784D0,2771D0,2878D0,2856D0/

      integer  :: i
      real(kind(1e0)) :: abicm
      real(kind(1e0)) :: trend(nobs+focast), season(nobs+focast)
      real(kind(1e0)) :: irreg(nobs), smooth(nobs+focast)
```

```
      character       :: months(focast)*3, years(nyears+2)*4
      data months/'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', &
                  'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'/
      data years/'    ', '1991', '1992', '1993', '1994', '1995', &
                 '1996', '1997','1998','1999','2000','2001','2002'/

      call bay_sea(b, trend=trend, seasonal=season, &
                   comp=irreg, abic=abicm, iorder=2, isorder=1,&
                   nfocast=focast, smoothed=smooth)

      write(nout,*) 'ABIC = ', abicm
      call wrrrl('TREND with last 12 values forecasted',trend,months, &
                  years, nra=12,nca=nyears+1,lda=12)

        call wrrrl('SEASONAL with last 12 values forecasted',season,&
                    months,years, nra=12,nca=nyears+1, lda=12)
        call wrrrl('IRREGULAR=Original data-TREND-SEASONAL',irreg, &
                   months,years, nra=12,nca=nyears,lda=12)

      end
```

Output

```
ABIC =  1297.6403

                  TREND with last 12 values forecasted
        1991     1992     1993     1994     1995     1996     1997     1998
Jan   2879.8   3318.9   3422.6   3228.7   2827.3   2795.7   2743.1   2481.3
Feb   2918.2   3359.9   3387.8   3206.0   2815.8   2785.6   2720.6   2469.7
Mar   2955.1   3399.1   3355.5   3177.0   2812.4   2777.9   2694.2   2455.4
Apr   2990.0   3436.1   3329.5   3142.1   2814.7   2773.2   2665.2   2439.0
May   3022.7   3469.0   3309.6   3103.9   2819.3   2771.3   2636.0   2422.8
Jun   3052.8   3496.1   3294.8   3064.8   2825.5   2771.0   2607.4   2408.5
Jul   3082.8   3514.5   3283.8   3025.7   2830.9   2772.4   2580.8   2396.8
Aug   3116.2   3521.9   3276.0   2987.3   2833.2   2773.6   2557.2   2387.7
Sep   3153.4   3517.8   3270.3   2949.0   2831.7   2774.7   2536.3   2380.3
Oct   3193.8   3503.7   3264.8   2911.3   2826.1   2774.5   2518.0   2373.6
Nov   3235.6   3482.4   3256.8   2876.6   2816.6   2770.4   2503.0   2366.6
Dec   3277.6   3455.2   3245.3   2847.6   2805.9   2760.2   2491.5   2359.1

        1999     2000     2001     2002
Jan   2352.1   2235.9   2206.0   3166.1
Feb   2345.6   2237.6   2239.0   3275.1
Mar   2338.1   2239.3   2281.4   3384.1
Apr   2328.3   2238.9   2333.1   3493.0
May   2315.5   2235.2   2393.9   3602.0
Jun   2301.4   2226.3   2464.7   3710.9
Jul   2286.0   2212.8   2545.7   3819.9
Aug   2269.8   2196.6   2636.7   3928.9
Sep   2255.5   2181.3   2735.7   4037.8
Oct   2244.6   2171.9   2840.4   4146.8
Nov   2238.3   2172.1   2948.2   4255.8
Dec   2235.6   2183.3   3057.2   4364.7
```

SEASONAL with last 12 values forecasted

	1991	1992	1993	1994	1995	1996	1997	1998	1999
Jan	162.9	165.6	169.3	172.0	173.8	176.3	176.4	177.3	176.4
Feb	51.4	51.5	50.5	49.5	48.6	48.8	50.0	51.1	51.0
Mar	-24.0	-23.9	-23.4	-18.8	-16.3	-13.0	-8.7	-4.9	-3.0
Apr	-191.0	-190.1	-189.1	-188.0	-186.6	-187.8	-188.5	-187.9	-186.6
May	-140.6	-143.3	-145.4	-147.4	-148.1	-147.1	-147.1	-147.9	-147.7
Jun	67.2	66.6	65.8	64.4	63.3	62.2	62.7	63.6	64.9
Jul	176.9	180.1	181.6	183.0	185.5	186.6	185.8	186.1	187.2
Aug	251.9	253.0	252.6	253.3	253.1	252.8	253.7	254.4	255.2
Sep	76.4	77.2	77.1	77.3	75.5	73.3	72.6	70.7	68.9
Oct	-79.5	-80.5	-80.1	-80.8	-81.5	-84.3	-87.6	-90.1	-93.4
Nov	-119.8	-120.7	-120.5	-120.2	-120.4	-119.7	-119.7	-118.1	-117.6
Dec	-235.7	-237.9	-243.0	-247.9	-250.5	-251.2	-254.2	-256.2	-257.5

	2000	2001	2002
Jan	177.2	177.6	177.5
Feb	50.8	51.6	51.5
Mar	-1.9	-1.7	-1.8
Apr	-187.2	-186.5	-186.6
May	-145.7	-145.6	-145.7
Jun	65.6	65.1	65.0
Jul	186.4	184.7	184.7
Aug	256.8	256.9	256.9
Sep	67.3	67.0	67.0
Oct	-95.1	-94.6	-94.6
Nov	-117.3	-116.6	-116.6
Dec	-258.1	-257.1	-257.1

IRREGULAR=Original data-TREND-SEASONAL

	1991	1992	1993	1994	1995	1996	1997	1998	1999
Jan	-74.7	-87.5	51.1	61.2	-57.1	116.0	-21.5	91.4	-81.5
Feb	39.4	35.6	-19.3	16.5	-77.4	-60.4	17.3	74.2	1.5
Mar	30.9	-47.2	-224.1	116.8	-47.1	-63.9	32.5	103.5	45.9
Apr	-25.0	-17.0	-22.5	-16.1	133.9	-30.5	-70.6	-38.0	108.3
May	157.8	-39.7	-18.2	-62.6	-93.1	52.8	31.2	-56.9	-81.9
Jun	45.0	14.3	24.4	-23.1	11.2	-92.1	-25.1	-23.2	30.7
Jul	-155.6	104.4	-7.4	-58.6	83.6	93.0	-58.6	-50.9	99.9
Aug	-55.0	92.1	-60.6	48.4	15.7	-60.4	0.1	-3.1	-49.9
Sep	-51.8	60.0	-17.4	109.6	26.8	-76.0	57.1	-2.1	-25.3
Oct	27.7	-63.3	59.3	-1.5	119.4	32.8	-50.3	42.5	-97.1
Nov	13.2	-51.7	-1.3	19.5	-44.3	54.3	-91.3	53.5	6.3
Dec	65.1	151.7	2.7	-132.8	-99.4	131.0	-50.3	-37.9	-43.1

	2000	2001
Jan	11.9	20.4
Feb	-43.5	38.4
Mar	60.5	5.3
Apr	-46.7	28.5
May	118.5	-3.3
Jun	87.1	-37.7
Jul	59.9	-94.4
Aug	85.6	-1.6
Sep	-66.6	-18.7

```
Oct  -117.8     25.3
Nov   -42.7     46.4
Dec   -91.2     56.0
```

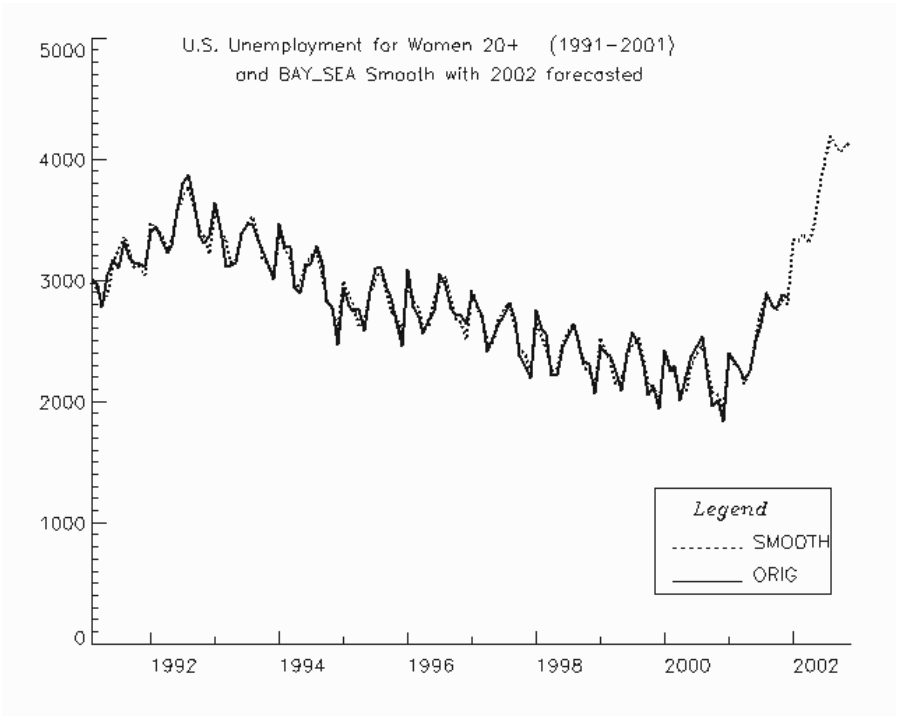


Figure 8-3 Sample Smoothed Predictions from `BAY_SEA`

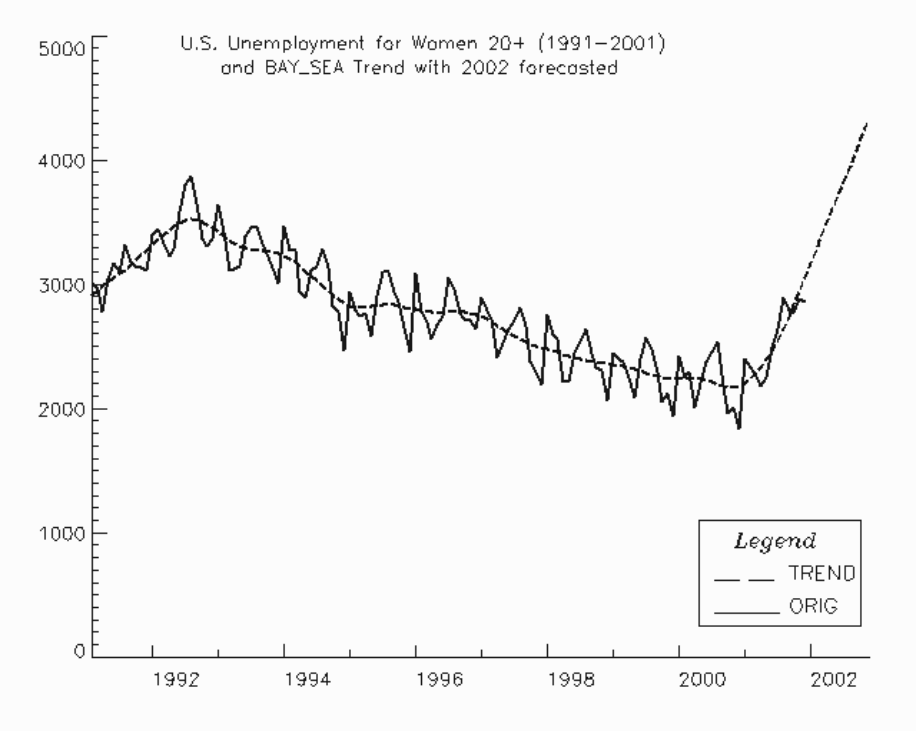


Figure 8-4 Sample Trend Predictions from BAY_SEA

Description

Routine BAY_SEA is based upon the algorithm published by Akaike (1980). This algorithm uses a Bayesian approach to the problem of fitting the following autoregressive model for a time series W_t decomposed into a trend and a seasonal component.

Adopting the notation described earlier in the "Usage Notes" section of this chapter (see page 673), if

$$t \in ZZ = \{\cdots, -2, -1, 0, 1, 2, \cdots\}$$

then a seasonal autoregressive model can be represented by the following relationship:

$$W_t = T_t + S_t + A_t$$

where W_t is the stationary time series with mean μ, T_t denotes an underlying trend, S_t seasonal component and A_t a noise or irregular component.

A non-Bayesian approach to this problem would be to estimate the trend and seasonal components by minimizing

$$\sum_{t=1}^{N}[\{W_t - T_t - S_t\}^2 + d^2\{(T_t - 2T_{t-2} + T_{t-2})^2 + r^2(S_t - S_{t-p})^2 + z^2(S_t + S_{t-1} + \cdots + S_{t-p})^2\}]$$

where p is the period of the seasonal component, and d, r, and z are properly chosen constants.

In BAY_SEA, the approach is to select the parameter *d*, which controls the smoothness of the trend and seasonality estimates, using Bayesian methods. The prior distribution controls the smoothness of the trend and seasonal components by assuming low order Gaussian autoregressive models for some differences of these components. The choice of the variance of the Gaussian distribution is realized by maximizing the log likelihood of the Bayesian model.

The other smoothing parameters, *r* and *z*, are determined by the value of RIGID. The default value for RIGID is 1. Larger values of RIGID produce a more rigid seasonal pattern. Normally, a series is first fit using the default value for RIGID. The smoothness of the trend and seasonality estimates are examined and then RIGID is either increased or decreased depending upon whether more or less seasonal smoothing is needed.

Additionally, BAY_SEA selects the optimum autoregressive model as the model that minimizes ABIC.

$$ABIC = -2 \bullet ln(likelihood),$$

where the *likelihood* in this case is the mixed Bayesian maximum likelihood. Smaller values of $ABIC$ represent a better fit.

OPT_DES

Optimal controller design based upon the methodology of Akaike and Nakagaw (1972). The routine allows for multiple channels for both the controlled and manipulated variables. The gain matrix is computed after automatically selecting the best autoregressive model using minimum multivariate final prediction error (MFPE).

Required Arguments

MAXLAG — Maximum number of autoregressive parameters requested. (Input)

X — NOBS by NCHANX matrix containing the multi-channel time series for the manipulated variables, also called control system input variables. Each row corresponds to a different observation in the series, and each column of X corresponds to a univariate time series for one of the controlled channels. (Input)

Y — NOBS by NCHANY matrix containing the multi-channel time series for the controlled variables, also called control system output variables. Each row corresponds to a different observations in the series, and each column of Y corresponds to a univariate time series for one of the manipulated channels. (Input)

NPAR — Number of autoregressive parameter matrices in minimum FPE model. (Output)

GAIN — NCHANX by NPAR gain matrix. (Output)

Optional Arguments

Q — NCHANY by NCHANY non-negative definite symmetric weighting matrix for the quadratic optimization criterion. (Input)

Default: If AVAR is non-singular, Q = Inverse of AVAR. If AVAR is singular, then Q = diag($1/AVAR_{1,1}$, $1/AVAR_{2,2}$, …, $1/AVAR_{NCHANY,NCHANY}$).

R — A NCHANX by NCHANX positive definite symmetric weighting matrix for the quadratic optimization criterion. (Input)
Default: R = diag($1/\sigma^2_{X1}$, $1/\sigma^2_{X2}$, …, $1/\sigma^2_{NCHANX}$)

IPRINT — Printing option. (Input)

0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: IPRINT = 0.

NCHANX — Number of time series for manipulated variables. (Input)
Default: NCHANX = size(X,2)

NCHANY — Number of time series for controlled variables. (Input)
Default: NCHANY = size(Y,2)

NOBS — Number of observations in each time series. (Input)
Default: NOBS = size(X,1)

NSTAGE — Number of stages used to compute the gain matrix, GAIN. (Input)
Default: NSTAGE = 20

TRANSY —NPAR by NCHANY by NCHANY transition matrix. (Output)

GAMMAX — NPAR by NCHANY by NCHANX gamma matrix. (Output)

CCV — NCHAN by NCHAN by (MAXLAG+1) matrix containing the sample autocovariances of the NCHAN time series variables, where NCHAN = NCHANX+NCHANY. For the ith time series variable, the first element, ACV(i ,i,1) must be the sample variance for the ith series and the remaining elements, ACV(i,i,2) … ACV(i,i,MAXLAG+1), contain the autocovariances of the ith series for lags 1 thru MAXLAG. Elements ACV(i,j,2) … ACV(i, j, MAXLAG+1), contain the autocovariances between the ith and jth series for lags 1 through MAXLAG (Input)

AVAR — NCHANY by NCHANY matrix containing estimates of the noise variances, autocovariances and cross-covariances for the NCHANY time series. (Output)

FPE — Final Prediction Error for fitted model. (Output)

AIC — Akaike's Information Criterion for fitted model. (Output)

FORTRAN 90 Interface

Generic: `CALL OPT_DES (MAXLAG, X, Y, NPAR, GAIN [,…])`

Specific: The specific interface names are `S_OPT_DES` and `D_OPT_DES`.

Example

The following example uses the Gas Furnace Data (Box and Jenkins 1976, pages 532-533. In this example, the controller has one variable, percent CO_2 , that is controlled by a single manipulated variable, input gas rate in cubic feet/minute. These multi-channel series consists of `NOBS`=100 observatons.

```
USE GDATA_INT
USE AUTO_MUL_AR_INT
USE WRRRN_INT

IMPLICIT NONE

INTEGER, PARAMETER :: NOBS=100, MAXLAG=10
 INTEGER NCOL, NROW, NMATRIX, I
 REAL RDATA(NOBS,4), X(NOBS,3), PAR(MAXLAG,3,3)

 CALL GDATA (8, RDATA, NROW, NCOL)
 X(:,1) = Rdata(:,3)
 X(:,2) = Rdata(:,4)
 X(:,3) = Rdata(:,2)

CALL AUTO_MUL_AR(X,MAXLAG,NMATRIX,PAR)
WRITE(*,*) "PAR = "
DO I=1,NMATRIX
CALL WRRRN("",PAR(I,:,:))
   ENDDO
 END
```

Output

```
 PAR =

         1       2       3
 1   0.822   0.000   0.000
 2   0.000   0.000  -0.290
 3   0.098   0.000   1.214

          1       2       3
 1  -0.1136  0.0000  0.0000
 2   0.2489  0.0000  0.0000
 3   0.0040  0.0000 -0.8682

          1       2       3
 1   0.0000  0.0000  0.0000
```

```
2   0.0000   0.0000   0.0000
3   0.0725   0.0000   0.3374

          1        2        3
1   0.0000   0.0000   0.0000
2   0.0000   0.0000   0.0000
3  -0.0693   0.0400  -0.2217

          1        2        3
1   0.0000   0.0000   0.0000
2   0.0000   0.0000   0.0000
3   0.0745  -0.0025   0.1253

          1        2        3
1   0.0000   0.0000   0.0000
2   0.0000   0.0000   0.0000
3  -0.0299   0.0510  -0.1570

           1         2         3
1   0.00000   0.00000   0.00000
2   0.00000   0.00000   0.00000
3   0.00806   0.05025   0.04558

           1         2         3
1   0.00000   0.00000   0.00000
2   0.00000   0.00000   0.00000
3   0.03858   0.01929  -0.09825

          1        2        3
1   0.0000   0.0000   0.0000
2   0.0000   0.0000   0.0000
3   0.0904  -0.0154   0.1376

           1         2         3
1   0.00000   0.00000   0.00000
2   0.00000   0.00000   0.00000
3   0.00000  -0.06879   0.00000
```

Description

The routine OPT_DES is based upon the FPEC and OPTDES program published in the TIMSAC –71 Library described by Akaike, H. and Nakagawa, T (1972). Estimates of the autoregressive parameters for the model with minimum multivariate final prediction error (MFPE) are calculated using the methodology described in Akaike, H., et. al (1979). Their methodology produces an estimate of the gain matrix, *G*, in a feedback control system with the following relationship:

$$X_t = GZ_t,$$

where

X_t is a vector containing the value of the NCHANX control variables at time = *t*,

Z_t = the state vector at time = *t*, and

G is the NCHANX by NPAR gain matrix (GAIN).

The gain matrix is estimated by minimizing the quadratic criterion:

$$J_I = E\left\{\sum_{t=1}^{I}\left[Z_t' Q Z_t + X_{t-1}' R X_{t-1}\right]\right\},$$

where E is the expectation operator, and I = NSTAGE.

Akaike and Nikagawa (1972) describe a process for obtaining the optimum gain matrix by adjusting the quadratic criterion matrices, Q and R. Initially they recommend setting Q and R to the default values described above for Q and R. They recommend that the behavior of the controller using the gain matrix obtained from these defaults be examined by simulating the controller.

From these simulations, the variances of the control input variables (X) should be examined to identify channels whose variances are too large or too small. If they are too large or small, their corresponding diagonal elements in R and Q should be decreased or increased.

The multivariate final prediction error for a multivariate autoregressive model with lag p is defined as:

$$MFPE_p = \frac{\left(1+\frac{k\cdot m+1}{N}\right)^m}{\left(1-\frac{k\cdot m+1}{N}\right)^m}\cdot\left\|\hat{\Sigma}_p\right\|,$$

where $\left\|\hat{\Sigma}_p\right\|$ is the determinant of the estimated NCHANY × NCHANY matrix of covariances of the white noise in the multivariate series, N =NOBS and m =NCHANY.

The model selected and parameter estimates vary depending upon the value of MAXLAG. Akaike and Nakagawa (1972) provide a rule of thumb for MAXLAG: start with values between $2\sqrt{N}/m$ and $3\sqrt{N}/m$, and keep MAXLAG below $N/(5\cdot m)$.

Similar to the univariate case, MAXLAG must never exceed $N/(2\cdot m)$.

The numerical accuracy decreases as MAXLAG increases. In this case, it is possible for the estimated MFPE to become negative. If this happens try using double precision.

NSTAGE, the number of stages used to compute the gain matrix, should be selected to provide an accurate estimate of the gain matrix. Essentially, the gain matrix is the matrix for control of the system at time = NSTAGE. As NSTAGE is increased, the gain matrix converges to a constant solution. That is, as NSTAGE is increased,

$$G_{NSTAGE} = G_{NSTAGE+1}$$

It is tempting to set NSTAGE to a large value. However, this will increase execution time and subject the final estimates to rounding errors.

LOFCF

Performs lack-of-fit test for a univariate time series or transfer function given the appropriate correlation function.

Required Arguments

NOBS — Number of observations in the stationary time series. (Input)
NOBS must be greater than or equal to two.

LAGMIN — Minimum lag of the correlation function. (Input)
LAGMIN corresponds to the lower bound of summation in the lack of fit test statistic. Generally, LAGMIN is set to one if CF is an autocorrelation function and is set to zero if CF is a cross correlation function.

LAGMAX — Maximum lag of the correlation function. (Input)
LAGMAX corresponds to the upper bound of summation in the lack of fit test statistic. LAGMAX must be greater than or equal to LAGMIN and less than NOBS.

CF — Vector of length LAGMAX + 1 containing the correlation function. (Input)
The correlation coefficient for lag k is given by CF(k + 1), k = LAGMIN, LAGMIN + 1, …, LAGMAX.

NPFREE — Number of free parameters in the formulation of the time series model. (Input)
NPFREE must be greater than or equal to zero and less than LAGMAX.

Q — Lack of fit test statistic. (Output)

PVALUE — p-value of the test statistic Q. (Output)
Under the null hypothesis, Q has an approximate chi-squared distribution with LAGMAX − LAGMIN + 1 − NPFREE degrees of freedom.

FORTRAN 90 Interface

Generic: `CALL LOFCF (NOBS, LAGMIN, LAGMAX, CF, NPFREE, Q, PVALUE)`

Specific: The specific interface names are `S_LOFCF` and `D_LOFCF`.

FORTRAN 77 Interface

Single: `CALL LOFCF (NOBS, LAGMIN, LAGMAX, CF, NPFREE, Q, PVALUE)`

Double: The double precision name is `DLOFCF`.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. An ARMA(2,1) with nonzero mean is fitted using routine NSLSE (page 733). The autocorrelations of the residuals are estimated using routine ACF (page 697). A portmanteau lack of fit test is computed using 10 lags with LOFCF.

The warning message from NSLSE in the output can be ignored. (See the example for routine NSLSE for a full explanation of the warning message.)

```
      USE IMSL_LIBRARIES
      INTEGER    IARDEG, IMEAN, IPRINT, ISEOPT, LAGMAX, LAGMIN, LDCOV,&
                 LDX, MAXBC, MDX, NOBS, NP, NPAR, NPFREE, NPMA
      PARAMETER  (IARDEG=2, IMEAN=1, IPRINT=0, ISEOPT=0, LAGMAX=10, &
                 LAGMIN=1, LDX=176, MAXBC=10, MDX=2, NOBS=100, NPAR=2, &
                 NPFREE=4, NPMA=1, NP=NPAR+NPMA+IMEAN, LDCOV=NP)
!
      INTEGER    LAGAR(NPAR), LAGMA(NPMA), MAXIT, NA, NCOL, NOUT, NROW
      REAL       A(NOBS-IARDEG+MAXBC), ACV(LAGMAX+1), AVAR, &
                 CF(LAGMAX+1), CNST, COV(LDCOV,NP), PAR(NPAR), &
                 PMA(NPMA), PVALUE, Q, RELERR, SEAC(LAGMAX), TOLBC, &
                 TOLSS, W(NOBS), WMEAN, X(LDX,MDX)
!
      EQUIVALENCE (W(1), X(22,2))
!
      DATA LAGAR/1, 2/, LAGMA/1/
!
      CALL UMACH (2, NOUT)
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, X, NROW, NCOL)
!                                USE Default Convergence parameters
!                                Compute preliminary parameter
!                                estimates for ARMA(2,1) model
      CALL NSPE (W, CNST, PAR, PMA, AVAR, WMEAN=WMEAN)
!                                Compute least squares estimates
!                                for model
      TOLSS = 0.125
!
      CALL NSLSE (W, PAR, LAGAR, PMA, LAGMA, MAXBC, CNST, COV, &
                 AVAR, IMEAN=IMEAN, WMEAN=WMEAN, TOLSS=TOLSS, A=A)
!                                Compute autocorrelations of the
!                                residuals
      CALL ACF (A, LAGMAX, CF)
!
      CALL LOFCF (NOBS, LAGMIN, LAGMAX, CF, NPFREE, Q, PVALUE)
!
      WRITE (NOUT,99998) Q
      WRITE (NOUT,99999) LAGMAX - LAGMIN + 1 - NPFREE, PVALUE
!
99998 FORMAT (/4X, 'Lack of Fit statistic (Q) = ', F12.3)
```

```
99999 FORMAT (/4X, 'Degrees of freedom (LAGMAX-LAGMIN+1-NPFREE) = ', &
            I8/4X, 'P-value (PVALUE) = ', F12.4)
      END
```

Output

```
***WARNING  ERROR 1 from NSLSE.  Least squares estimation of the parameters
***         has failed to converge.  Increase MAXBC and/or TOLBC and/or
***         TOLSS.  The estimates of the parameters at the last iteration
***         may be used as new starting values.

Lack of Fit statistic (Q) =       14.572

Degrees of freedom (LAGMAX-LAGMIN+1-NPFREE) =        6
P-value (PVALUE) =       0.9761
```

Comments

Routine LOFCF may be used to diagnose lack of fit in both ARMA and transfer function models. Typical arguments for these situations are

Model	LAGMIN	LAGMAX	NPFREE
ARMA(p, q)	1	$\sqrt{\text{NOBS}}$	$p+q$
Transfer function	0	$\sqrt{\text{NOBS}}$	$r+s$

See the "Description" section for further information.

Description

Routine LOFCF performs a portmanteau lack of fit test for a time series or transfer function containing n observations given the appropriate sample correlation function

$$\hat{\rho}(k)$$

for $k = L, L+1, \ldots, K$ where L = LAGMIN and K = LAGMAX.

The basic form of the test statistic Q is

$$Q = n(n+2)\sum_{k=L}^{K}(n-k)^{-1}\hat{\rho}(k)$$

with $L = 1$ if

$$\hat{\rho}(k)$$

is an autocorrelation function and $L = 0$ if

$$\hat{\rho}(k)$$

is a cross-correlation function. Given that the model is adequate, Q has a chi-squared distribution with $K - L + 1 - m$ degrees of freedom where m = NPFREE is the number of parameters estimated in the model. If the mean of the time series is estimated, Woodfield (1990) recommends not including this in the count of the parameters estimated in the model. Thus, for an ARMA(p, q) model set NPFREE = $p + q$ regardless of whether the mean is estimated or not.

The original derivation for time series models is due to Box and Pierce (1970) with the above modified version discussed by Ljung and Box (1978). The extension of the test to transfer function models is discussed by Box and Jenkins (1976, pages 394–395).

DIRIC

This function computes the Dirichlet kernel.

Function Return Value

DIRIC — Function value. (Output)

Required Arguments

M — Spectral window parameter. (Input)

RANGLE — Angle in radians. (Input)

EPS — Positive bound on |RANGLE| that determines when an approximation to the Dirichlet kernel is appropriate. (Input)
EPS must be between 0 and π inclusive. The approximation is used when |RANGLE| is less than EPS.

FORTRAN 90 Interface

Generic: `DIRIC (M, RANGLE, EPS)`

Specific: The specific interface names are `S_DIRIC` and `D_DIRIC`.

FORTRAN 77 Interface

Single: `DIRIC (M, RANGLE, EPS)`

Double: The double precision name is `DDIRIC`.

Example

In this example, DIRIC is used to compute the Dirichlet kernel at $\theta = \pm k\pi/(2M + 1)$ for $k = 0, 1, \ldots, (2M + 1)$ where $M = 5$ and $\varepsilon = 0.01$.

```
!                                  SPECIFICATIONS FOR LOCAL VARIABLES
      USE UMACH_INT
      USE DIRIC_INT

      INTEGER    K, M, NOUT
      REAL       EPS, PI, REAL, THETA, WT
      INTRINSIC  REAL
!
      M   = 5
```

```
      EPS = .01
      PI  = 3.14159
!
      CALL UMACH (2, NOUT)
!
      WRITE (NOUT,99998)
99998 FORMAT (’  K     THETA     WEIGHT ’)
      DO 10  K=0, 2*M + 1
         THETA = (PI*REAL(K))/REAL(2*M+1)
         WT    = DIRIC(M,THETA,EPS)
         WRITE (NOUT,99999) K, THETA, WT
99999    FORMAT (1X, I2, 2(3X,F8.5))
   10 CONTINUE
!
      END
```

Output

```
 K     THETA     WEIGHT
 0    0.00000    1.75070
 1    0.28560    1.11833
 2    0.57120    0.00000
 3    0.85680   -0.38312
 4    1.14240    0.00000
 5    1.42800    0.24304
 6    1.71359    0.00000
 7    1.99919   -0.18919
 8    2.28479    0.00000
 9    2.57039    0.16587
10    2.85599    0.00000
11    3.14159   -0.15915
```

Comments

1. The Dirichlet kernel is equivalent to the truncated periodogram spectral window. The spectral window parameter denotes the truncation point in the weighted sum of sample autocovariances used to estimate the spectral density.

2. The Dirichlet kernel produces negative values for certain values of RANGLE. Thus, spectral windows that use the Dirichlet kernel may also take on negative values.

3. The Dirichlet kernel is defined between $-\pi$ and π, inclusive, and is zero otherwise.

Description

Routine DIRIC evaluates the Dirichlet kernel, $D_M(\theta)$, for a given parameter M, angle θ = RANGLE, and tolerance ε = EPS. The computational form of the function is given by

$$D_M(\theta) = \begin{cases} \dfrac{(2M+1)}{2\pi}\left(\dfrac{\sin\left[\left(M+\frac{1}{2}\right)\theta\right]}{\left(M+\frac{1}{2}\right)\theta}\right) & |\theta| < \varepsilon \\ \dfrac{1}{2\pi}\left(\dfrac{\sin\left[\left(M+\frac{1}{2}\right)\theta\right]}{\sin(\theta/2)}\right) & \varepsilon \le |\theta| \le \pi \\ 0 & |\theta| > \pi \end{cases}$$

The first case is an approximation to $D_M(\theta)$ for small θ, and the second case is the usual theoretical definition.

In spectral analysis, the Dirichlet kernel corresponds to the truncated periodogram spectral window, and *M* is called the spectral window parameter. Since the Dirichlet kernel may be negative for certain values of θ, the truncated periodogram estimate of the spectral density may also be negative. This is an undesirable property since the true spectral density is a nonnegative function. See Priestley (1981, pages 437–438) and Anderson (1971, pages 508–511) for further discussion.

FEJER

This function computes the Fejér kernel.

Function Return Value

FEJER — Function value. (Output)

Required Arguments

M — Spectral window parameter. (Input)

RANGLE — Angle in radians. (Input)

EPS — Positive bound on |RANGLE| that determines when an approximation to the Fejér kernel is appropriate. (Input)
EPS must be between 0 and π inclusive. The approximation is used when |RANGLE| is less than EPS.

FORTRAN 90 Interface

Generic: `FEJER(M, RANGLE, EPS)`

Specific: The specific interface names are `S_FEJER` and `D_FEJER`.

FORTRAN 77 Interface

Single: `FEJER(M, RANGLE, EPS)`

Double: The double precision name is `DFEJER`.

Example

In this example, `FEJER` is used to compute the Fejér kernel at $\theta = \pm k\pi/M$ for $k = 0, 1, \ldots, M$ where $M = 11$ and $\varepsilon = 0.01$.

```
!
      USE UMACH_INT
      USE FEJER_INT

      INTEGER    K, M, NOUT
      REAL       EPS, PI, REAL, THETA, WT
      INTRINSIC  REAL
!
      M   = 11
      EPS = .01
      PI = 3.14159265
      CALL UMACH (2, NOUT)
!
      WRITE (NOUT,99998)
99998 FORMAT (’  K     THETA      WEIGHT ’)
      DO 10  K=0, M
         THETA = (PI*REAL(K))/REAL(M)
         WT    = FEJER(M,THETA,EPS)
         WRITE (NOUT,99999) K, THETA, WT
99999    FORMAT (1X, I2, 2(3X,F8.5))
   10 CONTINUE
!
      END
```

Output

```
 K     THETA     WEIGHT
 0    0.00000    1.75070
 1    0.28560    0.71438
 2    0.57120    0.00000
 3    0.85680    0.08384
 4    1.14240    0.00000
 5    1.42800    0.03374
 6    1.71360    0.00000
 7    1.99920    0.02044
 8    2.28479    0.00000
 9    2.57039    0.01572
10    2.85599    0.00000
11    3.14159    0.00000
```

Comments

1. The Fejér kernel is equivalent to the modified Bartlett spectral window. The spectral window parameter denotes the truncation point in the weighted sum of sample autocovariances used to estimate the spectral density.

2. The Fejér kernel produces nonnegative values for all values of RANGLE. Thus, spectral windows based on the Fejér kernel are always nonnegative.

3. The Fejér kernel is defined between $-\pi$ and π, inclusive, and is zero otherwise.

Description

Routine FEJER evaluates the Fejér kernel, $F_M(\theta)$, for a given parameter M, angle θ = RANGLE, and tolerance ε = EPS. The computational form of the function is given by

$$F_M(\theta) = \begin{cases} \dfrac{M}{2\pi}\left(\dfrac{\sin[M\theta/2]}{M\theta/2}\right)^2 & |\theta| < \varepsilon \\ \dfrac{1}{2\pi M}\left(\dfrac{\sin[M\theta/2]}{\sin(\theta/2)}\right)^2 & \varepsilon \le |\theta| \le \pi \\ 0 & |\theta| > \pi \end{cases}$$

The first case is an approximation to $F_M(\theta)$ for small θ, and the second case is the usual theoretical definition.

In spectral analysis, the Fejér kernel corresponds to the modified Bartlett spectral window, and M is called the spectral window parameter. Since the Fejér kernel is nonnegative for all values of θ, the modified Bartlett estimate of the spectral density is also nonnegative. This is a desirable property since the true spectral density is a nonnegative function. See Priestley (1981, pages 439–440) and Anderson (1971, pages 508–511) for further discussion.

ARMA_SPEC

Calculates the rational power spectrum for an ARMA model. It also computes the rational power spectrum for AR and MA models by setting the number of MA or AR coefficients to zero, repectively.

Required Arguments

PAR — Vector of length NPAR. If NPAR > 0 then PAR contains coefficients for the autoregressive terms in the model. If NPAR = 0, then the contents of PAR are ignored. (Input)

PMA — Vector of length NPMA. If NPMA > 0 then PMA contains the coefficients for the moving-average terms in the ARMA model. If NPMA = 0 then the contents of PMA are ignored. (Input)

NF — Number of frequencies at which to evaluate the spectral density estimate. NF must be greater than or equal to one. (Input)

AVAR — Estimate of the random shock variance. AVAR must be greater than zero. (Input)

S — Vector of length NF+1 containing the estimated power spectrum. (Output)

Optional Arguments

NPAR — Number of autoregressive (AR) parameters. NPAR must be greater than or equal to zero. (Input)
Default: NPAR=size(PAR).

NPMA — Number of moving-average (MA) parameters. Must be greater than or equal to zero. (Input)
Default: NPMA= size(PMA).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing
1	Prints final results only
2	Prints intermediate and final results

FORTRAN 90 Interface

Generic: `CALL ARMA_SPEC (PAR, PMA, NF, AVAR, S [,…])`

Specific: The specific interface names are S_ARMA_SPEC and D_ARMA_SPEC.

Example

Consider the Wolfer Sunspot Data (Box and Jenkins 1976, page 530) consisting of the number of sunspots observed each year from 1770 through 1869. These data can be modeled using the following model [ARMA(NPAR=2, NPMA=1)]

$$(1-\phi_1 B-\phi_2 B^2)(W_t-\mu)=(1-\theta_1 B)A_t$$

In this example, estimates of the coeffiecients in this model are obtained using MAX_ARMA. These are then sent to ARMA_SPEC to obtain the estimated power spectrum.

```
USE GDATA_INT
```

```
USE ARMA_SPEC_INT
USE MAX_ARMA_INT
USE WRRRN_INT

IMPLICIT NONE
INTEGER, PARAMETER :: NF=20, LDX=176, NDX=2
INTEGER NCOL, NROW
REAL(KIND(1E0))  PAR(2), PMA(1), RDATA(LDX,NDX), &
     W(100), S(0:NF), AVAR
EQUIVALENCE (W(1), RDATA(22,2))
DATA PAR/-0.5783E0, 0.18594E0/
DATA PMA/-0.1E0/

!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
CALL GDATA (2, RDATA, NROW, NCOL)
CALL MAX_ARMA(W, PAR, PMA, AVAR=AVAR)
CALL ARMA_SPEC(PAR, PMA, NF, AVAR, S)
CALL WRRRN("S", S)
END
```

Output

```
      S
  1    714.8
  2    786.7
  3   1030.2
  4   1450.4
  5   1619.6
  6   1146.4
  7    670.4
  8    407.2
  9    269.3
 10    192.3
 11    146.2
 12    116.7
 13     96.9
 14     83.2
 15     73.4
 16     66.3
 17     61.3
 18     57.7
 19     55.3
 20     53.9
 21     53.5
```

Description

The routine ARMA_SPEC is derived from the rational power spectrum analysis described by Akaike and Nakagawa (1972) and the RASPEC routine published in the original TIMSAC Library.

Using the notation developed in the introduction to this chapter, the stationary time series W_t with mean μ can be represented by the nonseasonal autoregressive moving average model (ARMA) by the following relationship:

$$\phi(B)(W_t - \mu) = \theta(B)A_t ,$$

where

$$t \in ZZ = \{\cdots, -2, -1, 0, 1, 2, \cdots\} ,$$

B is the backward shift operator defined by $B^k W_t = W_{t-k}$,

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_{NPAR} B^{NPAR} \quad NPAR \geq 0 ,$$

and

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_{NPMA} B^{NPMA} \quad NPMA \geq 0 .$$

Routine ARMA_SPEC uses estimates for the coefficients $\phi_1, \phi_2, \ldots, \phi_{NPAR}$, and $\phi_1, \phi_2, \ldots, \phi_{NPMA}$ as input to its algorithm, PAR and PMA respectively. These estimates can be derived from MAX_ARMA or by using NSLSE.

Routine ARMA_SPEC also requires an initial estimate for the variance of the white noise in the series. In MAX_ARMA this is returned as AVAR. This is also returned from the autocovariance procedure ACF as ACV(0).

PFFT

Computes the periodogram of a stationary time series using a fast Fourier transform.

Required Arguments

X — Vector of length NOBS containing the stationary time series. (Input)

PM — ($\lfloor N/2 \rfloor + 1$) by 5 matrix that contains a summarization of the periodogram analysis. (Output)
For $k = 0, 1, \ldots, \lfloor N/2 \rfloor$, the $(k + 1)$-*st* element of the j-th column of *PM* is defined as:

Col.	Description
1	Frequency, ω_k where $\omega_k = 2\pi_k/N$ for IFSCAL = 0 and $\omega_k = k/N$ for IFSCAL = 1.
2	Period, p_k where $p_k = 2\pi/\omega_k$ for IFSCAL = 0 and $p_k = 1/\omega_k$ for IFSCAL = 1. If $\omega_k = 0$, p_k is set to missing.
3	Periodogram ordinate, $I(\omega_k)$.
4	Cosine transformation coefficient, $A(\omega_k)$.
5	Sine transformation coefficient, $B(\omega_k)$.

Optional Arguments

NOBS — Number of observations in the stationary time series X. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the periodogram, and the cosine and sine transformations of the centered and padded time series.

XCNTR — Constant used to center the time series X. (Input)
Default: XCNTR = the arithmetic mean.

NPAD — Number of zeroes used to pad the centered time series. (Input)
NPAD must be greater than or equal to zero. The length of the centered and padded time series is N = NOBS + NPAD.
Default: NPAD = NOBS – 1.

IFSCAL — Option for frequency scale. (Input)
Default: IFSCAL = 0.

IFSCAL	Action
0	Frequency in radians per unit time
1	Frequency in cycles per unit time

IPVER — Option for version of the periodogram. (Input)
Default: IPVER = 0.

IPVER	Action
0	Compute usual periodogram.
1	Compute modified periodogram.

Refer to the "Description" section for further details.

LDPM — Leading dimension of PM exactly as specified in the dimension statement of the calling program. (Input)
LDPM must be greater than or equal to $\lfloor N/2 \rfloor + 1$.
Default: LDPM = size (PM,1).

FORTRAN 90 Interface

Generic: `CALL PFFT (X, PM, [,…])`

Specific: The specific interface names are S_PFFT and D_PFFT.

FORTRAN 77 Interface

Single:
```
CALL PFFT (NOBS, X, IPRINT, XCNTR, NPAD, IFSCAL, IPVER,
PM, LDPM)
```

Double: The double precision name is DPFFT.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Application of routine PFFT to these data produces the following results.

```
      USE GDATA_INT
      USE PFFT_INT
      USE WRRRL_INT
      INTEGER    LDPM, NOBS
      PARAMETER  (LDPM=100, NOBS=100)
!
      INTEGER    IPVER, NCOL, NPAD, NROW
      REAL       PM(LDPM,5), RDATA(176,2), REAL, X(NOBS)
      CHARACTER  CLABEL(6)*9, FMT*20, RLABEL(1)*6
      INTRINSIC  REAL
!
      EQUIVALENCE (X(1), RDATA(22,2))
!
      DATA RLABEL/'NONE'/, CLABEL/' ', 'Frequency', 'Period', &
```

```
           'I(w(k))', 'A(w(k))', 'B(w(k))'/
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Center on arithmetic mean
!                                Pad standard amount
!                                Frequency in radians per unit time
!                                Modified periodogram version
      IPVER = 1
!                                Compute the periodogram
      CALL PFFT (X, PM, IPVER=IPVER)
!                                Print results
      FMT = '(F9.4, F6.2, 3F10.2)'
      CALL WRRRL (' ', PM, RLABEL, CLABEL, 20, 5, FMT=FMT)
!
      END
```

Output

```
Frequency  Period     I(w(k))     A(w(k))     B(w(k))
   0.0000     NaN        0.00        0.00        0.00
   0.0316  199.00      183.97        3.72      -13.04
   0.0631   99.50     1363.37       35.45      -10.32
   0.0947   66.33     2427.09       29.31       39.60
   0.1263   49.75     1346.64      -21.74       29.56
   0.1579   39.80      139.74      -11.69       -1.79
   0.1894   33.17       44.67       -4.65        4.80
   0.2210   28.43      123.47      -11.11       -0.33
   0.2526   24.88      176.04       -4.79      -12.37
   0.2842   22.11      143.06        9.92       -6.69
   0.3157   19.90       44.17        6.43        1.68
   0.3473   18.09       38.95        5.40        3.13
   0.3789   16.58       63.20        7.14        3.49
   0.4105   15.31      537.64        0.89       23.17
   0.4420   14.21      944.68      -30.73       -0.75
   0.4736   13.27      162.02       -0.95      -12.69
   0.5052   12.44      908.09      -24.51      -17.53
   0.5368   11.71     3197.84       34.84      -44.54
   0.5683   11.06     1253.82       19.69       29.43
   0.5999   10.47      850.45       -8.75      -27.82
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `P2FT/DP2FT`. The reference is:

   ```
   CALL P2FT (NOBS, X, IPRINT, XCNTR, NPAD, IFSCAL,
   IPVER, PM, LDPM, CX, COEF, WFFTC, CPY)
   ```

   The additional arguments are as follows:

   CX — Complex work vector of length *N*.

   COEF — Complex work vector of length *N*.

WFFTC — Work vector of length $4N + 15$.

CPY — Work vector of length $2N$.

2. The centered and padded time series is defined by
   CX(j) = X(j) − XCNTR for $j = 1, \ldots,$ NOBS
   CX(j) = 0 for $j =$ NOBS + 1, …, N
   where $N =$ NOBS + NPAD.

3. The periodogram $I(\omega)$ is an even function of the frequency ω. The relation $I(-\omega) = I(\omega)$ for $\omega > 0.0$ recovers the periodogram for negative frequencies.

4. Since $\cos(\omega)$ is an even function of ω and $\sin(\omega)$ is an odd function of ω, the cosine and sine transformations, respectively, satisfy $A(-\omega) = A(\omega)$ and $B(-\omega) = -B(\omega)$ for $\omega > 0.0$. Similarly, the complex Fourier coefficients, stored in COEF, satisfy COEF($-\omega$) = conj(COEF(ω)).

5. Computation of the 2 * NOBS − 1 autocovariances of X using the inverse Fourier transform of the periodogram requires NPAD = NOBS − 1.

Description

Routine PFFT computes the periodogram of a stationary time series given a sample of $n =$ NOBS observations $\{X_t\}$ for $t = 1, 2, \ldots, n$.

Let

$$\{\tilde{X}_t\}$$

for $t = 1, \ldots, N$ represent the centered and padded data where $N =$ NOBS + NPAD,

$$\tilde{X}_t = \begin{cases} X_t - \hat{\mu}_X & t = 1, \ldots, n \\ 0 & t = (n+1), \ldots, N \end{cases}$$

and

$$\hat{\mu}_X = \text{XCNTR}$$

is determined by

$$\hat{\mu}_X = \begin{cases} \mu_X & \mu_X \text{ known} \\ \bar{X} = \dfrac{1}{n}\displaystyle\sum_{t=1}^{n} X_t & \mu_X \text{ unknown} \end{cases}$$

The discrete Fourier transform of

$$\{\tilde{X}_t\}$$

for $t = 1, \ldots, N$ is defined by

$$\zeta_{\tilde{X}}(\omega_k) = \sum_{t=1}^{N} \tilde{X}_t e^{-i\omega_k t}$$

over the discrete set of frequencies

$$\omega_k = \frac{2\pi k}{N} \qquad k = 0, \pm 1, \ldots, \pm \lfloor N/2 \rfloor$$

An alternative representation of

$$\zeta_{\tilde{X}}(\omega_k)$$

in terms of cosine and sine transforms is

$$\zeta_{\tilde{X}}(\omega_k) = \alpha_{\tilde{X}}(\omega_k) - i\beta_{\tilde{X}}(\omega_k)$$

where

$$\alpha_{\tilde{X}}(w_k) = \sum_{t=1}^{n} \tilde{X}_t \cos(\omega_k t)$$

and

$$\beta_{\tilde{X}}(\omega_k) = \sum_{t=1}^{n} \tilde{X}_t \sin(\omega_k t)$$

The fast Fourier transform algorithm is used to compute the discrete Fourier transform. The periodogram of the sample sequence $\{X_t\}$, $t = 1, \ldots, n$ computed with the centered and padded sequence

$$\{\tilde{X}_t\}$$

$t = 1, \ldots, N$ is defined by

$$I_{n,N,\tilde{X}}(\omega_k) = K \left| \sum_{t=1}^{N} \tilde{X}_t e^{-i\omega_k t} \right|^2 = K \left| \zeta_{\tilde{X}}(\omega_k) \right|^2$$

where K is the scale factor

$$K = \begin{cases} \dfrac{2}{n} & \text{for the usual periodogram} \\ \dfrac{1}{2\pi n} & \text{for the modified periodogram} \end{cases}$$

The scale factor of the usual periodogram relates the ordinates to the sum of squares of

$$X_t - \hat{\mu}_X$$

(Fuller 1976, pages 276–277). If the first ordinate (corresponding to $k = 0$) is replaced by one-half of its value, then if N is odd, the sum of the $\lfloor N/2 \rfloor + 1$ ordinates corresponding to $k = 0, 1, \ldots, \lfloor N/2 \rfloor$ is

$$\frac{N}{n}\sum_{t=1}^{n}(X_t - \hat{\mu}_X)^2$$

For N even, if the first ordinate (corresponding to $k = 0$) and the last ordinate (corresponding to $k = N/2$) are each replaced by one-half of their values, then the same relationship holds. The modified periodogram is an asymptotically unbiased estimate of the nonnormalized spectral density function at each frequency ω_k (Priestley 1981, page 417). The argument `IPVER` is used to specify the version of the periodogram.

The alternative representation of the discrete Fourier transform implies

$$I_{n,N,\tilde{X}}(\omega_k) = A^2_{\tilde{X}}(\omega_k) + B^2_{\tilde{X}}(\omega_k)$$

where

$$A_{\tilde{X}}(\omega_k) = K^{1/2}\alpha_{\tilde{X}}(\omega_k)$$

and

$$B_{\tilde{X}}(\omega_k) = K^{1/2}\beta_{\tilde{X}}(\omega_k)$$

represent the (scaled) cosine and sine transforms, respectively. Since the periodogram is an even function of the frequency, it is sufficient to estimate the periodogram at the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N} \qquad k = 0, 1, \ldots, \lfloor N/2 \rfloor$$

Use of the centered data

$$\{\tilde{X}_t\}$$

(without padding) instead of the original data $\{X_t\}$ for $t = 1, \ldots, n$ does not affect the asymptotic sampling properties of the periodogram. In fact,

$$I_{n,n,\tilde{X}}(\omega_k) \equiv I_{n,n,X}(\omega_k) \qquad \omega_k \neq 0$$

For $\omega_k = 0$, both

$$I_{n,n,\tilde{X}}(0) = 0$$

and

$$I_{n,n,X}(0) = K\left(\sum_{t=1}^{n} X_t\right)^2 = Kn^2\bar{X}^2$$

reflect the mean of the data. See Priestley (1981, page 417) for further discussion.

SSWD

Estimates the nonnormalized spectral density of a stationary time series using a spectral window given the time series data.

Required Arguments

X — Vector of length NOBS containing the stationary time series. (Input)

F — Vector of length NF containing the frequencies at which to evaluate the spectral density estimate. (Input)
The units of F correspond to the scale specified by IFSCAL. The elements of F must be in the range $(-\pi/\text{TINT}, \pi/\text{TINT})$, inclusive for IFSCAL = 0 and $(-1/(2 * \text{TINT}), 1/(2 * \text{TINT}))$ inclusive for IFSCAL = 1.

M — Vector of length NM containing the values of the spectral window parameters M. (Input)
For the Parzen spectral window (ISWVER = 5), all values of the spectral window parameters M must be even.

PM — $(\lfloor N/2 \rfloor + 1)$ by 5 matrix that contains a summarization of the periodogram analysis. (Output)
For $k = 0, 1, \ldots, \lfloor N/2 \rfloor$, the $(k + 1)$-st element of the j-th column of PM is defined as

Col.	Description
1	Frequency, ω_k where $\omega_k = 2\pi k/N$ for IFSCAL = 0 or $\omega_k = k/N$ for IFSCAL = 1.
2	Period, p_k where $p_k = 2\pi/\omega_k$ for IFSCAL = 0 and $p_k = 1/\omega_k$ for IFSCAL = 1. If $\omega_k = 0$, p_k is set to missing.
3	Periodogram ordinate, $I(\omega_k)$.
4	Cosine transformation coefficient, $A(\omega_k)$.
5	Sine transformation coefficient, $B(\omega_k)$.

Note N = NOBS + NPAD.

SM — NF by NM + 2 matrix containing a summarization of the spectral analysis. (Output)
The k-th element of the j-th column of SM is defined as

Col.	Description
1	Frequency, F(k).
2	Period, p_k where $p_k = 2\pi/\text{F}(k)$ for IFSCAL = 0 and $p_k = 1/\text{F}(k)$ for IFSCAL = 1. If F(k) = 0, p_k is set to missing.
3	Spectral density estimate at F(k) using the spectral window parameter M(1).
NM + 2	Spectral density estimate at F(k) using the spectral window parameter M(NM).

where $k = 1, \ldots,$ NF.

Optional Arguments

NOBS — Number of observations in the stationary time series X. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the periodogram, cosine transform and sine transform of the centered and padded time series, and the spectral density estimate based on a specified version of a spectral window for a given set of spectral window parameters.

XCNTR — Constant used to center the time series X. (Input)
Default: XCNTR = the arithmetic mean.

NPAD — Number of zeroes used to pad the centered time series. (Input)
NPAD must be greater than or equal to zero.
Default: NPAD = NOBS – 1.

IFSCAL — Option for frequency scale. (Input)
Default: IFSCAL = 0.

IFSCAL	Action
0	Frequency in radians per unit time.
1	Frequency in cycles per unit time.

NF — Number of frequencies at which to evaluate the spectral density estimate. (Input)
Default: NF = size (F,1).

TINT — Time interval at which the series is sampled. (Input)
For a discrete parameter process, usually TINT = 1. For a continuous parameter process, TINT > 0. TINT is used to adjust the spectral density estimate.
Default: TINT = 1.0.

ISWVER — Option for version of the spectral window. (Input)
Default: ISWVER = 1.

ISWVER	Action
1	Modified Bartlett
2	Daniell
3	Tukey-Hamming
4	Tukey-Hanning
5	Parzen
6	Bartlett-Priestley

Refer to the "Algorithm" section for further details.

NM — Number of spectral window parameters `M` used to compute the spectral density estimate for a given spectral window version. (Input)
`NM` must be greater than or equal to one.
Default: `NM` = size (`M`,1).

LDPM — Leading dimension of `PM` exactly as specified in the dimension statement of the calling program. (Input)
`LDPM` must be greater than or equal to $\lfloor N/2 \rfloor + 1$.
Default: `LDPM` = size (`PM`,1).

LDSM — Leading dimension of `SM` exactly as specified in the dimension statement of the calling program. (Input)
`LDSM` must be greater than or equal to `NF`.
Default: `LDSM` = size (`SM`,1).

FORTRAN 90 Interface

Generic: `CALL SSWD (X, F, M, PM, SM [,…])`

Specific: The specific interface names are `S_SSWD` and `D_SSWD`.

FORTRAN 77 Interface

Single:
```
CALL SSWD (NOBS, X, IPRINT, XCNTR, NPAD, IFSCAL, NF, F,
TINT, ISWVER, NM, M, PM, LDPM, SM, LDSM)
```

Double: The double precision name is `DSSWD`.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of

the number of sunspots observed from 1770 through 1869. Application of routine SSWD to these data produces the following results:

```
      USE GDATA_INT
      USE SSWD_INT
      USE WRRRL_INT
      INTEGER    LDPM, LDSM, NF, NM, NOBS
      REAL       PI
      PARAMETER  (NF=20, NM=3, NOBS=100, PI=3.141592654, &
                 LDPM=NOBS, LDSM=NF)
!
      INTEGER    I, ISWVER, M(NM), NCOL, NROW
      REAL       F(NF), PM(LDPM,5), RDATA(176,2), FLOAT, SM(LDSM,5), &
                  X(NOBS)
      CHARACTER  CLABEL(6)*9, FMT*20, RLABEL(1)*6, TITLE*60
      INTRINSIC  REAL
!
      EQUIVALENCE (X(1), RDATA(22,2))
!
      DATA RLABEL/'NONE'/, CLABEL/' ', 'Frequency', 'Period', &
          'M = 10', 'M = 20', 'M = 30'/
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Center on arithmetic mean
!                                Pad standard amount (Default)
!     USE Default Frequency in radians per unit time
!                                Determine frequencies at which
!                                to evaluate spectral density
      DO 10  I=1, NF
         F(I) = PI*FLOAT(I)/FLOAT(NF)
   10 CONTINUE
!     USE Default Time interval for discrete data
!                                Spectral window parameters
      M(1) = 10
      M(2) = 20
      M(3) = 30
!                                Compute spectral density using
!                                the Parzen window
      ISWVER = 5
      CALL SSWD (X, F, M, PM, SM,  ISWVER=ISWVER)
!                                Print results
      TITLE = 'Spectral Density Using the Parzen Window'
      FMT   = '(F9.4, F6.2, 3F10.2)'
      CALL WRRRL (TITLE, SM, RLABEL, CLABEL, FMT=FMT)
!                                Compute spectral density using
!                                the Bartlett-Priestley window
      ISWVER = 6
      CALL SSWD (X, F, M, PM, SM, ISWVER=ISWVER)
!                                Print results
      TITLE = '%/Spectral Density Using the Bartlett-Priestley '// &
            'Window'
      CALL WRRRL (TITLE, SM, RLABEL, CLABEL, FMT=FMT)
!
      END
```

Output

Spectral Density Using the Parzen Window

Frequency	Period	M = 10	M = 20	M = 30
0.1571	40.00	659.64	617.42	619.73
0.3142	20.00	666.95	554.70	339.61
0.4712	13.33	653.73	770.64	860.49
0.6283	10.00	598.77	857.80	1046.13
0.7854	8.00	497.47	582.85	550.77
0.9425	6.67	367.72	266.33	186.98
1.0996	5.71	240.65	121.46	104.79
1.2566	5.00	142.41	76.17	76.74
1.4137	4.44	81.28	54.20	47.19
1.5708	4.00	49.13	40.16	41.39
1.7279	3.64	32.57	27.58	26.46
1.8850	3.33	22.44	16.52	14.40
2.0420	3.08	15.53	10.93	9.87
2.1991	2.86	11.19	8.30	8.32
2.3562	2.67	8.66	6.18	5.86
2.5133	2.50	6.93	4.75	4.22
2.6704	2.35	5.51	4.62	4.35
2.8274	2.22	4.47	4.91	5.24
2.9845	2.11	3.61	4.23	4.75
3.1416	2.00	2.62	2.44	2.27

Spectral Density Using the Bartlett-Priestley Window

Frequency	Period	M = 10	M = 20	M = 30
0.1571	40.00	604.34	712.73	757.61
0.3142	20.00	564.28	176.81	107.08
0.4712	13.33	767.63	927.14	981.10
0.6283	10.00	900.32	1190.30	1172.23
0.7854	8.00	607.45	494.85	571.65
0.9425	6.67	237.16	127.65	87.36
1.0996	5.71	103.34	113.93	135.34
1.2566	5.00	75.74	74.88	57.57
1.4137	4.44	52.64	44.98	38.59
1.5708	4.00	38.50	44.56	50.59
1.7279	3.64	27.35	25.28	21.76
1.8850	3.33	15.68	13.84	13.10
2.0420	3.08	10.33	9.79	7.41
2.1991	2.86	7.95	8.31	8.67
2.3562	2.67	6.04	5.86	7.08
2.5133	2.50	4.56	3.67	2.90
2.6704	2.35	4.44	4.38	4.06
2.8274	2.22	4.99	5.62	5.40
2.9845	2.11	4.31	5.07	5.08
3.1416	2.00	2.43	2.23	2.44

Comments

1. Workspace may be explicitly provided, if desired, by use of `S2WD/DS2WD`. The reference is:

```
CALL S2WD (NOBS, X, IPRINT, XCNTR, NPAD, IFSCAL, NF,
 F, TINT, ISWVER, NM, M, PM, LDPM, SM, LDSM, CX, COEF,
WFFTC, CPY)
```

The additional arguments are as follows:

CX — Complex vector of length N containing the centered and padded time series X. (Output)

COEF — Complex vector of length N containing the Fourier coefficients of the finite Fourier transform of CX. (Output)
Note that COEF(k) is the appropriately scaled Fourier coefficient at frequency ω_k, $k = 0, 1, \ldots, N-1$.

WFFTC — Vector of length $4N + 15$.

CPY — Vector of length $2N$.

2. The normalized spectral density estimate is obtained by dividing the nonnormalized spectral density estimate in matrix SM by an estimate of the variance of X.

Description

Routine SSWD estimates the nonnormalized spectral density function of a stationary time series using a spectral window given a sample of n = NOBS observations $\{X_t\}$ for $t = 1, 2, \ldots, n$.

Let

$$\{\tilde{X}_t\}$$

for $t = 1, \ldots, N$ represent the centered and padded data where N = NOBS + NPAD,

$$\tilde{X}_t = \begin{cases} X_t - \hat{\mu}_X & t = 1, \ldots, n \\ 0 & t = (n+1), v \ldots, N \end{cases}$$

and

$$\hat{\mu}_X = \text{XCNTR}$$

is determined by

$$\hat{\mu}_X = \begin{cases} \mu_X & \mu_X \text{ known} \\ \dfrac{1}{n}\displaystyle\sum_{t=1}^{n} X_t & \mu_X \text{ unknown} \end{cases}$$

The modified periodogram of

$$\{\tilde{X}_t\}$$

for $t = 1, \ldots, N$ is estimated by

$$I_{n,N,\tilde{X}}(\omega_k) = A^2_{\tilde{X}}(\omega_k) + B^2_{\tilde{X}}(\omega_k)$$

where

$$A_{\tilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{X}_t\cos(\omega_k t)$$

and

$$B_{\tilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{X}_t\sin(\omega_k t)$$

represent the

$$\tilde{X}_t$$

cosine and sine transforms, respectively, and K is the scale factor equal to $1/(2\pi n)$. Since the periodogram is an even function of the frequency, it is sufficient to estimate the periodogram over the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N} \qquad k = 0,1,\ldots,\lfloor N/2 \rfloor$$

The routine PFFT (page 825) is used to compute the modified periodogram of

$$\tilde{X}_t$$

The estimate of the nonnormalized spectral density $h_X(\omega)$ is computed according to

$$\hat{h}_X(\omega) = \frac{2\pi}{N}\sum_{k=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} I_{n,N,\tilde{X}}(\omega_k)W_n(\omega-\omega_k)$$

where the spectral window $W_n(\theta)$ is specified by argument ISWVER. The following spectral windows $W_n(\theta)$ are available.

Modified Bartlett

$$W_n(\theta) = \frac{1}{2\pi M}\left\{\frac{\sin(M\theta/2)}{\sin(\theta/2)}\right\}^2 = F_M(\theta)$$

where $F_M(\theta)$ corresponds to the Fejér kernel of order M.

Daniell

$$W_n(\theta) = \begin{cases} M/2\pi & -\pi/M \le \theta \le \pi/M \\ 0 & \text{otherwise} \end{cases}$$

Tukey

$$W_n(\theta) = aD_M\left(\theta - \frac{\pi}{M}\right) + (1-2a)D_M(\theta) + aD_M\left(\theta + \frac{\pi}{M}\right) \quad 0 < a \le 0.25$$

where $D_M(\theta)$ represents the Dirichlet kernel. The Tukey-Hamming window is obtained when $a = 0.23$ and the Tukey-Hanning window is obtained when $a = 0.25$.

Parzen

$$W_n(\theta) = \frac{6\pi}{M}\left[F_{M/2}(\theta)\right]^2\left\{1 - \frac{2}{3}\sin^2(\theta/2)\right\}$$

where M is even. If M is odd, then $M + 1$ is used instead of M in the above formula.

Bartlett-Priestley

$$W_n(\theta) = \begin{cases} \dfrac{3M}{4\pi}\left\{1 - \left(\dfrac{M\theta}{\pi}\right)^2\right\} & |\theta| \le \pi/M \\ 0 & |\theta| > \pi/M \end{cases}$$

The argument NM specifies the number of window parameters M and corresponds to the number of spectral density estimates to be computed for a given spectral window. The nonnormalized spectral density is estimated over the set of frequencies

$$\omega = f_i \qquad i = 1, \ldots, n_f$$

where n_f = NF. These frequencies are in the scale specified by the argument IFSCAL but are transformed to the scale of radians per unit time for computational purposes.

The above formula for

$$\hat{h}_X(\omega)$$

assumes the data $\{X_t\}$ correspond to a realization of a discrete-parameter stationary process observed consecutively in time. In this case, the observations are equally spaced in time with interval Δt = TINT equivalent to one. However, if the data correspond to a realization of a continuous-parameter stationary process recorded at equal time intervals, then the estimate of the nonnormalized spectral density must be adjusted for the effect of aliasing. In general, the estimate of $h_X(\omega)$ is given by

$$\hat{h}_X(\omega) = \Delta t\, \hat{h}_X(\omega) \qquad |\omega| \le \pi/\Delta t$$

Note that the frequency ω of the desired spectral estimate is assumed to be input in a form already adjusted for the time interval Δt. Approximate confidence intervals for $h(\omega)$ can be computed using formulas given in the introduction.

SSWP

Estimates the nonnormalized spectral density of a stationary time series using a spectral window given the periodogram.

Required Arguments

N — Number of observations in the centered and padded time series X. (Input)
N must be greater than or equal to two.

PX — Vector of length $\lfloor N/2 \rfloor + 1$ containing the (modified) periodogram of X. (Input)
The periodogram ordinate evaluated at (angular) frequency $w_k = 2\pi k/N$ is given by PX(k + 1), $k = 0, 1, \ldots, \lfloor N/2 \rfloor$.

F — Vector of length NF containing the (angular) frequencies at which the spectral density is estimated. (Input)

M — Spectral window parameter. (Input)
M must be greater than or equal to one and less than N.

SX — Vector of length NF containing the estimate of the spectral density of the time series X. (Output)

Optional Arguments

NF — Number of (angular) frequencies. (Input)
NF must be greater than or equal to one.
Default: NF = size (F,1).

ISWVER — Option for version of the spectral window. (Input)
Default: ISWVER = 1.

ISWVER	Action
1	Modified Bartlett
2	Daniell
3	Tukey-Hamming
4	Tukey-Hanning
5	Parzen
6	Bartlett-Priestley

Refer to the "Algorithm" section for further details.

FORTRAN 90 Interface

Generic: `CALL SSWP (N, PX, F, M, SX [,…])`

Specific: The specific interface names are `S_SSWP` and `D_SSWP`.

FORTRAN 77 Interface

Single: `CALL SSWP (N, PX, NF, F, ISWVER, M, SX)`

Double: The double precision name is `DSSWP`.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Application of routine `SSWP` to these data produces the following results:

```
      USE IMSL_LIBRARIES
      INTEGER    LDPM, LDSM, NF, NM, NOBS
      REAL       PI
      PARAMETER  (NF=20, NM=3, NOBS=100, PI=3.141592654, &
                 LDPM=NOBS, LDSM=NF)
!
      INTEGER    I, IPVER, ISWVER, J, M(NM), N, NCOL, NROW
      REAL       F(NF), PM(LDPM,5), PX(LDPM), RDATA(176,2), FLOAT, &
                 SM(NF,5), SX(NF), X(NOBS)
      CHARACTER  CLABEL(6)*9, FMT*20, RLABEL(1)*6, TITLE*60
      INTRINSIC  FLOAT
!
      EQUIVALENCE (PX(1), PM(1,3)), (F(1), SM(1,1))
      EQUIVALENCE (X(1), RDATA(22,2))
!
      DATA RLABEL/'NONE'/, CLABEL/' ', 'Frequency', 'Period', &
          'M = 10', 'M = 20', 'M = 30'/
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Center on arithmetic mean
!                                Pad standard amount
      NPAD = NOBS-1
!                                Frequency in radians per unit time
!                                Modified periodogram version
      IPVER = 1
!                                Compute periodogram
      CALL PFFT (X, PM, IPVER=IPVER)
!                                Number of observations used to
!                                compute the periodogram
      N = NOBS + NPAD
!                                Determine frequency and period
!                                at which to evaluate the spectral
!                                density
```

```
      DO 10  I=1, NF
         SM(I,1) = PI*FLOAT(I)/FLOAT(NF)
         SM(I,2) = 2.0*FLOAT(NF)/FLOAT(I)
   10 CONTINUE
!                                  Spectral window parameters
      M(1) = 10
      M(2) = 20
      M(3) = 30
!                                  Compute spectral density using
!                                  the Parzen window
      ISWVER = 5
      DO 20  J=1, NM
         CALL SSWP (N, PX, F, M(J), SX, ISWVER=ISWVER)
!                                  Copy into SM
         CALL SCOPY (NF, SX, 1, SM(1:,2+J), 1)
   20 CONTINUE
!                                  Print results
      TITLE = 'Spectral Density Using the Parzen Window'
      FMT   = '(F9.4, F6.2, 3F10.2)'
      CALL WRRRL (TITLE, SM, RLABEL, CLABEL, FMT=FMT)
!                                  Compute spectral density using
!                                  the Bartlett-Priestley window
      ISWVER = 6
      DO 30  J=1, NM
         CALL SSWP (N, PX, F, M(J), SX, ISWVER=ISWVER)
!                                  Copy into SM
         CALL SCOPY (NF, SX, 1, SM(1:,2+J), 1)
   30 CONTINUE
!                                  Print results
      TITLE = '%/Spectral Density Using the Bartlett-Priestley '// &
            & 'Window'
      CALL WRRRL (TITLE, SM, RLABEL, CLABEL, FMT=FMT)
!
      END
```

Output

```
      Spectral Density Using the Parzen Window
Frequency  Period      M = 10      M = 20      M = 30
   0.1571   40.00      659.64      617.42      619.73
   0.3142   20.00      666.95      554.70      339.61
   0.4712   13.33      653.73      770.64      860.49
   0.6283   10.00      598.77      857.80     1046.13
   0.7854    8.00      497.47      582.85      550.77
   0.9425    6.67      367.72      266.33      186.98
   1.0996    5.71      240.65      121.46      104.79
   1.2566    5.00      142.41       76.17       76.74
   1.4137    4.44       81.28       54.20       47.19
   1.5708    4.00       49.13       40.16       41.39
   1.7279    3.64       32.57       27.58       26.46
   1.8850    3.33       22.44       16.52       14.40
   2.0420    3.08       15.53       10.93        9.87
   2.1991    2.86       11.19        8.30        8.32
   2.3562    2.67        8.66        6.18        5.86
   2.5133    2.50        6.93        4.75        4.22
   2.6704    2.35        5.51        4.62        4.35
```

```
   2.8274    2.22        4.47        4.91        5.24
   2.9845    2.11        3.61        4.23        4.75
   3.1416    2.00        2.62        2.44        2.27

Spectral Density Using the Bartlett-Priestley Window
Frequency  Period      M = 10      M = 20      M = 30
   0.1571   40.00      604.34      712.73      757.61
   0.3142   20.00      564.28      176.81      107.08
   0.4712   13.33      767.63      927.14      981.10
   0.6283   10.00      900.32     1190.30     1172.23
   0.7854    8.00      607.45      494.85      571.65
   0.9425    6.67      237.16      127.65       87.36
   1.0996    5.71      103.34      113.93      135.34
   1.2566    5.00       75.74       74.88       57.57
   1.4137    4.44       52.64       44.98       38.59
   1.5708    4.00       38.50       44.56       50.59
   1.7279    3.64       27.35       25.28       21.76
   1.8850    3.33       15.68       13.84       13.10
   2.0420    3.08       10.33        9.79        7.41
   2.1991    2.86        7.95        8.31        8.67
   2.3562    2.67        6.04        5.86        7.08
   2.5133    2.50        4.56        3.67        2.90
   2.6704    2.35        4.44        4.38        4.06
   2.8274    2.22        4.99        5.62        5.40
   2.9845    2.11        4.31        5.07        5.08
   3.1416    2.00        2.43        2.23        2.44
```

Comments

1. The periodogram of `X` may be computed using the routine `PFFT` (page 825). Estimation of the spectral density of `X` using the modified periodogram preserves the scale of the spectral density up to adjustment for the time sampling interval.

2. The time sampling interval, `TINT`, is assumed to be equal to one. This assumption is appropriate for discrete parameter processes. The adjustment for continuous parameter processes (`TINT` > 0.0) involves multiplication of the frequency vector `F` by 1/`TINT` and multiplication of the spectral density estimate by `TINT`.

3. To convert the frequency scale from radians per unit time to cycles per unit time, multiply `F` by $1/(2\pi)$.

Description

Routine `SSWP` estimates the nonnormalized spectral density function of a stationary time series using a spectral window given the modified periodogram of the appropriately centered and padded data

$$\{\tilde{X}_t\} \qquad \text{for } t = 1, \ldots, N$$

The routine `PFFT` (page 825) may be used to obtain the modified periodogram

$$I_{N,\tilde{X}}(\omega_k)$$

over the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N}, \quad k = 0,1,\ldots,\lfloor N/2 \rfloor$$

The symmetry of the periodogram is used to recover the ordinates at negative frequencies.

The estimate of the nonnormalized spectral density $h_X(\omega)$ is computed according to

$$\hat{h}_X(\omega) = \frac{2\pi}{N} \sum_{k=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} I_{n,N,\bar{X}}(\omega_k) W_n(\omega - \omega_k)$$

where the spectral window $W_n(\theta)$ is specified by argument `ISWVER`. The following spectral windows $W_n(\theta)$ are available.

Modified Bartlett

$$W_n(\theta) = \frac{1}{2\pi M}\left\{\frac{\sin(M\theta/2)}{\sin(\theta/2)}\right\}^2 = F_M(\theta)$$

where $F_M(\theta)$ corresponds to the Fejer kernel of order M.

Daniell

$$W_n(\theta) = \begin{cases} M/2\pi & -\pi/M \le \theta \le \pi/M \\ 0 & \text{otherwise} \end{cases}$$

Tukey

$$W_n(\theta) = aD_M\left(\theta - \frac{\pi}{M}\right) + (1-2a)D_M(\theta) + aD_M\left(\theta + \frac{\pi}{M}\right), \quad 0 < a \le 0.25$$

where $D_M(\theta)$ represents the Dirichlet kernel. The Tukey-Hamming window is obtained when $a = 0.23$, and the Tukey-Hanning window is obtained when $a = 0.25$.

Parzen

$$W_n(\theta) = \frac{6\pi}{M}\left[F_{M/2}(\theta)\right]^2 \left\{1 - \frac{2}{3}\sin^2(\theta/2)\right\}$$

where M is even. If M is odd, then $M + 1$ is used instead of M in the above formula.

Bartlett-Priestley

$$W_n(\theta) = \begin{cases} \frac{3M}{4\pi}\left\{1 - \left(\frac{M\theta}{\pi}\right)^2\right\} & |\theta| \le \pi/M \\ 0 & |\theta| > \pi/M \end{cases}$$

Only one window parameter M may be specified so that only one estimate of $h_X(\omega)$ is computed. The nonnormalized spectral density is estimated over the set of frequencies

$$\omega = f_i, \quad i = 1, \ldots, n_f$$

where n_f = NF. These frequencies are in the scale of radians per unit time. The time sampling interval Δt is assumed to be equal to one.

SWED

Estimations of the nonnormalized spectral density of a stationary time series based on specified periodogram weights given the time series data.

Required Arguments

X — Vector of length NOBS containing the stationary time series. (Input)

F — Vector of length NF containing the frequencies at which to evaluate the spectral density estimate. (Input)
The units of F correspond to the scale specified by IFSCAL. The elements of F must be in the range ($-\pi$/TINT, π/TINT), inclusive, for IFSCAL = 0 and (-1/(2 * TINT), 1/(2 * TINT)), inclusive, for IFSCAL = 1.

WT — Vector of length NWT containing the weights used to smooth the periodogram. (Input)
The actual weights are the values in WT normalized to sum to 1 with the current periodogram ordinate taking the middle weight for NWT odd or the weight to the right of the middle for NWT even.

PM — ($\lfloor N/2 \rfloor + 1$) by 5 matrix that contains a summarization of the periodogram analysis. (Output)
For $k = 0, 1, \ldots, \lfloor N/2 \rfloor$, the $(k + 1)$-st element of the j-th column of PM is defined as

Col.	Description
1	Frequency, ω_k where $\omega_k = 2\pi k/N$ for IFSCAL = 0 or $\omega_k = k/N$ for IFSCAL = 1.
2	Period, p_k where $p_k = 2\pi/\omega_k$ for IFSCAL = 0 and $p_k = 1/\omega_k$ for IFSCAL = 1. If $\omega_k = 0$, p_k is set to the missing value or NaN (not a number).
3	Periodogram ordinate, $I(\omega_k)$.
4	Cosine transformation coefficient, $A(\omega_k)$.
5	Sine transformation coefficient, $B(\omega_k)$.

SM — NF by 3 matrix containing a summarization of the spectral analysis. (Output)
The k-th element of the j-th column of SM is defined as

Col.	Description
1	Frequency, F(k).
2	Period, p_k where $p_k = 2\pi/$F(k) for IFSCAL = 0 and $p_k = 1/$F(k) for IFSCAL = 1. If F(k) = 0, p_k is set to missing.
3	Spectral density estimate at F(k) using the specified weights WT.

where $k = 1, \ldots,$ NF.

Optional Arguments

NOBS — Number of observations in the stationary time series X. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the periodogram, cosine and sine transforms of the centered and padded time series, and the spectral density estimate based on a specified weight sequence.

XCNTR — Constant used to center the time series X. (Input)
Default: XCNTR = the arithmetic mean.

NPAD — Number of zeroes used to pad the centered time series. (Input)
NPAD must be greater than or equal to zero. The length of the centered and padded time series is $N =$ NOBS + NPAD.
Default: NPAD = NOBS –1.

IFSCAL — Option for frequency scale. (Input)
Default: IFSCAL = 0.

IFSCAL	Action
0	Frequency in radians per unit time.
1	Frequency in cycles per unit time.

NF — Number of frequencies at which to evaluate the spectral density estimate. (Input)
NF must be greater than zero.
Default: NF = size (F,1).

TINT — Time interval at which the series is sampled. (Input)
For a discrete parameter process, usually TINT = 1.0. For a continuous parameter process, TINT > 0.0. TINT is used to adjust the spectral density estimate.
Default: TINT = 1.0.

NWT — Number of weights. (Input)
NWT must be greater than or equal to one.
Default: NWT = size (WT,1).

LDPM — Leading dimension of PM exactly as specified in the dimension statement in the calling program. (Input)
LDPM must be greater than or equal to $\lfloor N/2 \rfloor + 1$.
Default: LDPM = size (PM,1).

LDSM — Leading dimension of SM exactly as specified in the dimension statement in the calling program. (Input)
LDSM must be greater than or equal to NF.
Default: LDSM = size (SM,1).

FORTRAN 90 Interface

Generic: `CALL SWED (X, F, WT, PM, SM [,…])`

Specific: The specific interface names are S_SWED and D_SWED.

FORTRAN 77 Interface

Single: `CALL SWED (NOBS, X, IPRINT, XCNTR, NPAD, IFSCAL, NF, F, TINT, NWT, WT, PM, LDPM, SM, LDSM)`

Double: The double precision name is DSWED.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Application of routine SWED to these data produces the following results:

```
      USE IMSL_LIBRARIES
      INTEGER    LDPM, LDRDAT, LDSM, NDRDAT, NF, NOBS, NPAD, NWT
      PARAMETER  (LDRDAT=176, NDRDAT=2, NF=20, NOBS=100, NWT=7, &
                  LDSM=NF, NPAD=NOBS-1, LDPM=(NOBS+NPAD)/2+1)
!
      INTEGER    I, NROW, NVAR
```

```
      REAL       F(NF), PI, PM(LDPM,5), RDATA(LDRDAT,NDRDAT), &
                 REAL, SM(LDSM,3), WT(NWT), X(NOBS)
      CHARACTER  CLABEL(4)*20, FMT*20, RLABEL(1)*4, TITLE*28
      INTRINSIC  FLOAT
!
      EQUIVALENCE (X(1), RDATA(22,2))
!
      DATA WT/1.0, 2.0, 3.0, 4.0, 3.0, 2.0, 1.0/
      DATA IPRINT/0/, IFSCAL/0/, TINT/1.0/
      DATA FMT/'(F9.4, F6.2, F9.4)'/
      DATA RLABEL/'NONE'/
      DATA CLABEL/' ', '%/Frequency', '%/Period', 'Spectral%/Estimates' &
          /
      DATA TITLE/'Results of Spectral Analysis'/
!                                Initializations
      PI = 2.0*ASIN(1.0)
      DO 10  I=1, NF
         F(I) = PI*FLOAT(I)/FLOAT(NF)
   10 CONTINUE
!                                Wolfer Sunspot Data for years
!                                1770 through 1869
      CALL GDATA (2, RDATA, NROW, NVAR)
!                                Center on arithmetic mean
!                                Spectral density
      CALL SWED (X, F, WT, PM, SM)
!                                Print Results
      CALL WRRRL (TITLE, SM, RLABEL, CLABEL, FMT=FMT)
!
      END
```

Output

```
Results of Spectral Analysis
                    Spectral
Frequency  Period  Estimates
   0.1571   40.00   710.8386
   0.3142   20.00   116.3940
   0.4712   13.33   937.1508
   0.6283   10.00  1209.8268
   0.7854    8.00   538.9236
   0.9425    6.67    84.9561
   1.0996    5.71   128.0791
   1.2566    5.00    55.0304
   1.4137    4.44    40.2022
   1.5708    4.00    46.4240
   1.7279    3.64    21.0053
   1.8850    3.33    12.1449
   2.0420    3.08     8.8654
   2.1991    2.86     7.2589
   2.3562    2.67     6.8078
   2.5133    2.50     3.3873
   2.6704    2.35     3.9504
   2.8274    2.22     5.7418
   2.9845    2.11     4.4652
   3.1416    2.00     4.1216
```

Comments

1. Workspace may be explicitly provided, if desired, by use of S2ED/DS2ED. The reference is:

   ```
   CALL S2ED (NOBS, X, IPRINT, XCNTR, NPAD, IFSCAL, NF, F, TINT,
   NWT, WT, PM, LDPM, SM, LDSM,CX, COEF, WFFTC, CPY)
   ```

   The additional arguments are as follows:

   CX — Complex vector of length N containing the centered and padded time series X. (Output)

   COEF — Complex vector of length N containing the Fourier coefficients of the finite Fourier transform of CX. (Output)
   Note that COEF(k + 1) is the appropriately scaled Fourier coefficient at frequency ω_k, $k = 0, 1, \ldots, N-1$.

   WFFTC — Work vector of length $4N + 15$.

   CPY — Work vector of length $2N$.

2. The centered and padded time series is defined by
   CX(j) = X(j) − XCNTR for $j = 1, \ldots,$ NOBS
   CX(j) = 0 for j = NOBS + 1, …, N
   where N = NOBS + NPAD.

3. The normalized spectral density estimate is obtained by dividing the nonnormalized spectral density estimate in matrix SM by an estimate of the variance of X.

Description

Routine SWED estimates the nonnormalized spectral density function of a stationary time series using a fixed sequence of weights, given a sample of n = NOBS observations $\{X_t\}$, for $t = 1, 2, \ldots, n$.
Let

$$\{\tilde{X}_t\} \text{ for } t = 1, \ldots, N$$

represent the centered and padded data where N = NOBS + NPAD,

$$\tilde{X}_t = \begin{cases} X_t - \hat{\mu}_X, & t = 1, \ldots, n \\ 0, & t = (n+1), \ldots, N \end{cases}$$

and

$$\hat{\mu}_X = \text{XCNTR}$$

is determined by

$$\hat{\mu}_X = \begin{cases} \mu_X, & \mu_X \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} X_t, & \mu_X \text{ unknown} \end{cases}$$

The modified periodogram of

$$\{\tilde{X}_t\} \text{ for } t = 1, \ldots, N$$

is estimated by

$$I_{n,N,\tilde{X}}(\omega_k) = A^2_{\tilde{X}}(\omega_k) + B^2_{\tilde{X}}(\omega_k)$$

where

$$A_{\tilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N} \tilde{X}_t \cos(\omega_k t)$$

and

$$B_{\tilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N} \tilde{X}_t \sin(\omega_k t)$$

represent the

$$\tilde{X}_t$$

cosine and sine transforms, respectively, and K is the scale factor equal to $1/(2\pi n)$. Since the periodogram is an even function of the frequency, it is sufficient to estimate the periodogram at the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N}, \quad k = 0, 1, \ldots, \lfloor N/2 \rfloor$$

(Here, $\lfloor a \rfloor$ means the greatest integer less than or equal to a.) The routine `PFFT` (page 825) is used to compute the modified periodogram of

$$\{\tilde{X}_t\}$$

Consider the sequence of m = `NWT` weights

$$\{w_j\} \text{ for } j = -\lfloor m/2 \rfloor, \ldots, (m - \lfloor m/2 \rfloor - 1)$$

where

$$\sum_j w_j = 1$$

These weights are fixed in the sense that they do not depend on the frequency ω at which to estimate the nonnormalized spectral density $h_X(\omega)$. The estimate of the nonnormalized spectral density is computed according to

$$\hat{h}_X(\omega) = \sum_j \omega_j I_{n,N,\tilde{X}}(\omega_{k,j})$$

where

$$\omega_{k,j} = \frac{2\pi\{k(\omega)+j\}}{N}$$

and $k(\omega)$ is the integer such that $\omega_{k,0}$ is closest to ω. The weights specified by argument WT may be relative since they are normalized to sum to one in the actual computation of

$$\hat{h}_X(\omega)$$

Usually, m is odd with the weights symmetric about the middle weight w_0. If m is even, the weight to the right of the middle is considered w_0. Note that periodogram ordinate

$$I_{n,N,\tilde{X}}(0)$$

is replaced by

$$I_{n,N,\tilde{X}}(\omega_1)$$

and the sum reflects at each end. The nonnormalized spectral density is estimated over the set of frequencies

$$\omega = f_i, \qquad i = 1, \ldots, n_f$$

where n_f = NF. These frequencies are in the scale specified by the argument IFSCAL but are transformed to the scale of radians per unit time for computational purposes.

The above formula for

$$\hat{h}_X(\omega)$$

assumes the data $\{X_t\}$ correspond to a realization of a discrete-parameter stationary process observed consecutively in time. In this case, the observations are equally spaced in time with interval Δt = TINT equivalent to one. However, if the data correspond to a realization of a continuous-parameter stationary process recorded at equal time intervals, then the estimate of the nonnormalized spectral density must be adjusted for the effect of aliasing. In general, the estimate of $h_X(\omega)$ is given by

$$\hat{h}_X(\omega) = \Delta t\, \hat{h}_X(\omega), \quad |\omega| \le \pi / \Delta t$$

Note that the frequency ω of the desired spectral estimate is assumed to be input in a form already adjusted for the time interval Δt.

Approximate confidence intervals for $h(\omega)$ can be computed using formulas given in the introduction.

SWEP

Estimations of the nonnormalized spectral density of a stationary time series based on specified periodogram weights given the periodogram.

Required Arguments

N — Number of observations in the appropriately centered and padded time series X. (Input)
N must be greater than or equal to two.

PX — Vector of length $\lfloor N/2 \rfloor + 1$ containing the (modified) periodogram of X. (Input)
The periodogram ordinate evaluated at (angular) frequency $\omega_k = 2\pi k/N$ is given by PX(k + 1), $k = 0, 1, \ldots, \lfloor N/2 \rfloor$.

F — Vector of length NF containing the (angular) frequencies at which the spectral density is estimated. (Input)

WT — Vector of length NWT containing the weights used to smooth the periodogram. (Input)
The actual weights are the values in WT normalized to sum to 1 with the current periodogram ordinate taking the middle weight for NWT odd or the weight to the right of the middle for NWT even.

SX — Vector of length NF containing the estimate of the spectral density of the time series X. (Output)

Optional Arguments

NF — Number of (angular) frequencies. (Input)
NF must be greater than or equal to one.
Default: NF = size (F,1).

NWT — Number of weights. (Input)
NWT must be greater than or equal to one.
Default: NWT = size (WT,1).

FORTRAN 90 Interface

Generic: `CALL SWEP (N, PX, F, WT, SX [,…])`

Specific: The specific interface names are S_SWEP and D_SWEP.

FORTRAN 77 Interface

Single: `CALL SWEP (N, PX, NF, F, NWT, WT, SX)`

Double: The double precision name is DSWEP.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of

the number of sunspots observed from 1770 through 1869. Application of routine SWEP to these data produces the following results:

```
      USE IMSL_LIBRARIES
      INTEGER    LDPM, LDRDAT, N, NDRDAT, NF, NOBS, NPAD, NWT
      PARAMETER  (LDRDAT=176, NDRDAT=2, NF=20, NOBS=100, NWT=7, &
                 NPAD=NOBS-1, LDPM=(NOBS+NPAD)/2+1, N=NOBS+NPAD)
!
      INTEGER    I, IFSCAL, IPVER, NROW, NVAR
      REAL       F(NF), PI, PM(LDPM,5), RDATA(LDRDAT,NDRDAT), &
                 FLOAT, SM(NF,2), SX(NF), WT(NWT), X(NOBS)
      CHARACTER  CLABEL(3)*30, FMT*20, RLABEL(1)*4, TITLE*28
      INTRINSIC  FLOAT
!
      EQUIVALENCE (X(1), RDATA(22,2))
!
      DATA WT/1., 2., 3., 4., 3., 2., 1./
      DATA IPVER/1/, IFSCAL/0/
      DATA FMT/'(F9.4)'/
      DATA CLABEL/'     ', '%/Frequency', 'Spectral%/Estimates'/
      DATA RLABEL/'NONE'/
      DATA TITLE/'Results of Spectral Analysis'/
!                                Initialization
      PI = 2.0*ASIN(1.0)
      DO 10  I=1, NF
         F(I) = PI*FLOAT(I)/FLOAT(NF)
   10 CONTINUE
!                                Wolfer Sunspot Data for years
!                                1770 through 1869
      CALL GDATA (2, RDATA, NROW, NVAR)
!                                Compute modified periodogram
      CALL PFFT (X, PM, IPVER=IPVER)
!
!                                Compute spectral density
      CALL SWEP (N, PM(:,3), F, WT, SX)
!
!                                Print results
!
!                                Copy the frequencies to the output
!                                matrix
      CALL SCOPY (NF, F, 1, SM(1:,1), 1)
!                                Copy the spectral estimates to the
!                                output matrix
      CALL SCOPY (NF, SX, 1, SM(1:,2), 1)
!                                Call printing routine
      CALL WRRRL (TITLE, SM, RLABEL, CLABEL, FMT=FMT)
!
      END
```

Output

```
Results of Spectral Analysis
            Spectral
Frequency  Estimates
   0.1571   710.8386
   0.3142   116.3940
```

```
0.4712   937.1508
0.6283  1209.8268
0.7854   538.9236
0.9425    84.9561
1.0996   128.0791
1.2566    55.0304
1.4137    40.2022
1.5708    46.4240
1.7279    21.0053
1.8850    12.1449
2.0420     8.8654
2.1991     7.2589
2.3562     6.8078
2.5133     3.3873
2.6704     3.9504
2.8274     5.7418
2.9845     4.4652
3.1416     4.1216
```

Comments

1. The periodogram of X may be computed using the routine PFFT (page 825). Estimation of the spectral density of X using the modified periodogram preserves the scale of the spectral density up to adjustment for the time sampling interval.

2 The time sampling interval, TINT, is assumed to be equal to one. This assumption is appropriate for discrete parameter processes. The adjustment for continuous parameter processes (TINT > 0) involves multiplication of the frequency vector F by 1/TINT and multiplication of the spectral density estimate by TINT.

3. To convert the frequency scale from radians per unit time to cycles per unit time, multiply F by 1/(2π).

Description

Routine SWEP estimates the nonnormalized spectral density function of a stationary time series using a fixed sequence of weights given the modified periodogram of the appropriately centered and padded data

$$\{\tilde{X}_t\} \text{ for } t = 1, \ldots, N$$

The routine PFFT (page 825) may be used to obtain the modified periodogram

$$I_{n,N,\tilde{X}}(\omega_k)$$

over the discrete set of nonnegative frequencies

$$w_k = \frac{2\pi k}{N}, \quad k = 0,1,\ldots,\lfloor N/2 \rfloor$$

(Here, $\lfloor a \rfloor$ means the greatest integer less than or equal to a.) The symmetry of the periodogram is used to recover the ordinates at negative frequencies.

Consider the sequence of m = NWT weights $\{w_j\}$ for $j = -\lfloor m/2 \rfloor, \ldots, (m - \lfloor m/2 \rfloor - 1)$ where $\sum_j w_j = 1$. These weights are fixed in the sense that they do not depend on the frequency ω at which to estimate the nonnormalized spectral density $h_X(\omega)$. The estimate of the nonnormalized spectral density is computed according to

$$\hat{h}_X(\omega) = \sum_j \omega_j I_{n,N,\tilde{X}}(\omega_{k,j})$$

where

$$\omega_{k,j} = \frac{2\pi\{k(\omega) + j\}}{N}$$

and $k(\omega)$ is the integer such that $\omega_{k,0}$ is closest to ω. The weights specified by argument WT may be relative since they are normalized to sum to one in the actual computation of

$$\hat{h}_X(\omega)$$

Usually, m is odd with the weights symmetric about the middle weight w_0. If m is even, the weight to the right of the middle is considered w_0. Note that periodogram ordinate

$$I_{n,N,\tilde{X}}(0)$$

is replaced by

$$I_{n,N,\tilde{X}}(\omega_1)$$

and the sum reflects at each end.

The nonnormalized spectral density estimate is computed over the set of frequencies

$$\omega = f_i, \qquad i = 1, \ldots, n_f$$

where n_f = NF. These frequencies are in the scale of radians per unit time. The time sampling interval Δt is assumed to be equal to one.

Approximate confidence intervals for $h(\omega)$ can be computed using formulas given in the introduction.

CPFFT

Computes the cross periodogram of two stationary time series using a fast Fourier transform.

Required Arguments

X — Vector of length NOBS containing the first stationary time series. (Input)

Y — Vector of length NOBS containing the second stationary time series. (Input)

CPM — ($\lfloor N/2 \rfloor + 1$) by 10 matrix containing a summarization of the results of the cross periodogram analysis. (Output)
For $k = 0, 1, \ldots, \lfloor N/2 \rfloor$, the $(k + 1)$-st element of the j-th column of `CPM` is defined as

Col.	Description
1	Frequency, ω_k where $\omega_k = 2\pi k/N$ for `IFSCAL` = 0 or $\omega_k = k/N$ for `IFSCAL` = 1.
2	Period, p_k where $p_k = 2\pi/\omega_k$ for `IFSCAL` = 0 and $p_k = 1/\omega_k$ for `IFSCAL` = 1. If $\omega_k = 0$, p_k is set to missing.
3	`X` periodogram ordinate, $I_X(\omega_k)$
4	`X` cosine transformation coefficient, $A_X(\omega_k)$
5	`X` sine transformation coefficient, $B_X(\omega_k)$
6	`Y` periodogram ordinate, $I_Y(\omega_k)$
7	`Y` cosine transformation coefficient, $A_Y(\omega_k)$
8	`Y` sine transformation coefficient, $B_Y(\omega_k)$
9	Real part of the `XY` cross periodogram ordinate $I_{XY}(\omega_k)$.
10	Imaginary part of the `XY` cross periodogram ordinate $I_{XY}(\omega_k)$.

Optional Arguments

NOBS — Number of observations in each stationary time series `X` and `Y`. (Input)
`NOBS` must be greater than or equal to two.
Default: `NOBS` = size (`X`,1).

IPRINT — Printing option. (Input)
Default: `IPRINT` = 0.

`IPRINT`	Action
0	No printing is performed.
1	Prints the periodogram, cosine and sine series, and the real and imaginary components of the cross periodogram.

XCNTR — Constant used to center the time series `X`. (Input)
Default: `XCNTR` = the arithmetic mean.

YCNTR — Constant used to center the time series `Y`. (Input)
Default: `YCNTR` = the arithmetic mean.

NPAD — Number of zeroes used to pad each centered time series. (Input)
`NPAD` must be greater than or equal to zero. The length of each centered and padded time series is `N` = `NOBS` + `NPAD`.
Default: `NPAD` = `NOBS` – 1.

IFSCAL — Option for frequency scale. (Input)
Default: `IFSCAL`= 0.

`IFSCAL`	Action
0	Frequency in radians per unit time
1	Frequency in cycles per unit time

IPVER — Option for version of the periodogram. (Input)
Default: `IPVER` = 0.

`IPVER`	Action
0	Compute usual periodogram.
1	Compute modified periodogram.

Refer to the "Description" section for further details.

LDCPM — Leading dimension of `CPM` exactly as specified in the dimension statement of the calling program. (Input)
`LDCPM` must be greater than or equal to $\lfloor N/2 \rfloor + 1$.
Default: `LDCPM` = size (`CPM`,1).

FORTRAN 90 Interface

Generic: `CALL CPFFT (X, Y, CPM [,…])`

Specific: The specific interface `names` are `S_CPFFT` and `D_CPFFT`.

FORTRAN 77 Interface

Single:
```
CALL CPFFT (NOBS, X, Y, IPRINT, XCNTR, YCNTR, NPAD, IFSCAL,
IPVER, CPM, LDCPM)
```

Double: The double precision name is `DCPFFT`.

Example

Consider the Robinson Multichannel Time Series Data (Robinson 1967, page 204) where X is the Wölfer sunspot number and Y is the northern light activity for the time period from 1770 through 1869. Application of routine CPFFT to these data produces the following results. Note that CPFFT sets CPM (1, 2) to the missing value code via routine AMACH (page 1334). The printing of CPM (1, 2) depends on the computer.

```
      USE GDATA_INT
      USE CPFFT_INT
      USE WRRRL_INT
      INTEGER    LDCPM, LDRDAT, NDRDAT, NOBS, NPAD
      PARAMETER  (LDRDAT=100, NDRDAT=4, NOBS=100, NPAD=NOBS-1, &
                 LDCPM=(NOBS+NPAD)/2+1)
!
      INTEGER    IPVER, NRCOL, NRROW
      REAL       CPM(LDCPM,10), FLOAT, RDATA(LDRDAT,NDRDAT), &
                 X(NOBS), Y(NOBS)
      CHARACTER  CLABEL1(6)*9, CLABEL2(6)*9, FMT*7, RLABEL(1)*6, &
                  TITLE*41
      INTRINSIC  FLOAT
!
      EQUIVALENCE (X(1), RDATA(1,2)), (Y(1), RDATA(1,3))
!
      DATA TITLE/'Results of the Cross Periodogram Analysis'/
      DATA FMT/'(F10.3)'/
      DATA CLABEL1/'k+1', 'w(k)', 'p(k)', 'IX(w(k))', 'AX(w(k))', &
          'BX(w(k))'/
      DATA CLABEL2/'k+1', 'IY(w(k))', 'AY(w(k))', 'BY(w(k))', &
          'Real IXY', 'Imag. IXY'/
      DATA RLABEL/'NUMBER'/
!
!                                Robinson Data
      CALL GDATA (8, RDATA, NRROW, NRCOL)
!                                Center on arithmetic means
!                                Frequency in radians per unit time
!                                Modified periodogram version
      IPVER = 1
!                                Compute the cross periodogram
      CALL CPFFT (X, Y, CPM, IPVER=IPVER)
!
!                                Print results (First 10 rows)
      CALL WRRRL (TITLE, CPM, RLABEL, CLABEL1, 10, 5, FMT=FMT)
      CALL WRRRL ('%/', CPM(1:,6), RLABEL, CLABEL2, 10, 5, FMT=FMT)
!
      END
```

Output

```
          Results of the Cross Periodogram Analysis
k+1        w(k)        p(k)     IX(w(k))     AX(w(k))     BX(w(k))
  1       0.000         NaN        0.000        0.000        0.000
  2       0.032     199.000      184.159        3.742      -13.044
  3       0.063      99.500     1364.408       35.457      -10.354
  4       0.095      66.333     2433.933       29.411       39.610
```

```
  5       0.126      49.750    1351.002     -21.749      29.631
  6       0.158      39.800     140.421     -11.716      -1.773
  7       0.189      33.167      44.117      -4.671       4.722
  8       0.221      28.429     121.186     -11.003      -0.343
  9       0.253      24.875     176.275      -4.782     -12.386
 10       0.284      22.111     144.867      10.038      -6.642

k+1    IY(w(k))    AY(w(k))    BY(w(k))    Real IXY   Imag. IXY
  1       0.000       0.000       0.000       0.000       0.000
  2    1689.212     -37.480     -16.866      79.776    -552.014
  3    4113.003      41.232     -49.122    1970.577   -1314.779
  4    3255.785      44.214      36.068    2729.031    -690.474
  5    1757.663      -8.162      41.122    1396.006    -652.513
  6    1002.050     -30.107       9.778     335.410    -167.954
  7      62.360      -6.825       3.972      50.636      13.678
  8    1481.396     -38.096       5.487     417.288     -73.451
  9    1274.161     -17.176     -31.291     469.704     -63.095
 10     488.479     -12.442     -18.267      -3.570    -265.992
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `C2FFT/DC2FFT`. The reference is:

   ```
   CALL C2FFT (NOBS, X, Y, IPRINT, XCNTR, YCNTR, NPAD,  IFSCAL,
   IPVER, CPM, LDCPM, CX, COEF, WFFTC, CPY)
   ```

   The additional arguments are as follows:

   CX — Complex work vector of length *N*.

   COEF — Complex work vector of length *N*.

   WFFTC — Work vector of length $4N + 15$.

   CPY — Work vector of length $2N$.

2. The centered and padded time series are defined by

   $CX(j) = X(j) - XCNTR$ for $j = 1, \ldots, NOBS$
   $CX(j) = 0$ for $j = NOBS + 1, \ldots, N$
   and
   $CY(j) = Y(j) - YCNTR$ for $j = 1, \ldots, NOBS$
   $CY(j) = 0$ for $j = NOBS + 1, \ldots, N$
   where $N = NOBS + NPAD$.

3. The cross periodogram $I_{XY}(\omega)$ is complex valued in general. The relation $I_{XY}(-\omega) = \text{conj}(I_{XY}(\omega))$ for $w > 0.0$ recovers the cross periodogram for negative frequencies since $\text{real}(I_{XY}(-\omega)) = \text{real}(I_{XY}(\omega))$ and $\text{imag}(I_{XY}(-\omega)) = -\text{imag}(I_{XY}(\omega))$. The periodogram $I(\omega)$ is an even function of the frequency ω. The relation $I(-\omega) = I(\omega)$ for $\omega > 0.0$ recovers the periodogram for negative frequencies.

4. Since cos(ω) is an even function of ω and sin(ω) is an odd function of ω, the cosine and sine transformations, respectively, satisfy $A(-\omega) = A(\omega)$ and $B(-\omega) = -B(\omega)$ for $\omega > 0.0$. Similarly, the complex Fourier coefficients, stored in `COEF`, satisfy `COEF`$(-\omega)$ = conj(`COEF`(ω)).

5. Computation of the 2 * `NOBS` − 1 cross-covariances of `X` and `Y` using the inverse Fourier transform of the cross periodogram requires `NPAD` = `NOBS` − 1.

Description

Routine `CPFFT` computes the cross periodogram of two jointly stationary time series given a sample of n = `NOBS` observations $\{X_t\}$ and $\{Y_t\}$ for $t = 1, 2, \ldots, n$.

Let

$$\{\tilde{X}_t\} \text{ for } t = 1, \ldots, N$$

represent the centered and padded data where N = `NOBS` + `NPAD`,

$$\tilde{X}_t = \begin{cases} X_t - \hat{\mu}_X & t = 1, \ldots, n \\ 0 & t = (n+1), \ldots, N \end{cases}$$

and

$$\hat{\mu}_X = \text{XCNTR}$$

is determined by

$$\hat{\mu}_X = \begin{cases} \mu_X & \mu_X \text{ known} \\ \dfrac{1}{n}\displaystyle\sum_{t=1}^{n} X_t & \mu_X \text{ unknown} \end{cases}$$

Similarly, let

$$\{\tilde{Y}_t\} \text{ for } t = 1, \ldots, N$$

represent the centered and padded data where

$$\tilde{Y}_t = \begin{cases} Y_t - \hat{\mu}_Y & t = 1, \ldots, n \\ 0 & t = (n+1), \ldots, N \end{cases}$$

and

$$\hat{\mu}_Y = \text{YCNTR}$$

is determined by

$$\hat{\mu}_Y = \begin{cases} \mu_Y & \mu_Y \text{ known} \\ \dfrac{1}{n}\displaystyle\sum_{t=1}^{n} Y_t & \mu_Y \text{ unknown} \end{cases}$$

The periodogram of the sample sequence $\{X_t\}$, $t = 1, \ldots, n$ computed with the padded sequence

$$\{\tilde{X}_t\} \text{ for } t = 1, \ldots, N$$

is defined by

$$I_{n,N,\tilde{X}}(\omega_k) = A^2_{\tilde{X}}(\omega_k) + B^2_{\tilde{X}}(\omega_k)$$

where

$$A_{\tilde{X}}(\omega_k) = K^{1/2} \sum_{t=1}^{N} \tilde{X}_t \cos(\omega_k t)$$

and

$$B_{\tilde{X}}(\omega_k) = K^{1/2} \sum_{t=1}^{N} \tilde{X}_t \sin(\omega_k t)$$

represent the

$$\tilde{X}_t$$

cosine and sine transforms, respectively, and K is the scale factor

$$K = \begin{cases} \dfrac{2}{n} & \text{for the usual periodogram,} \\ \dfrac{1}{2\pi n} & \text{for the modified periodogram} \end{cases}$$

The periodogram of the sample sequence $\{Y_t\}$, $t = 1, \ldots, n$ computed with the padded sequence

$$\{\tilde{Y}_t\} \text{ for } t = 1, \ldots, N$$

is defined by

$$I_{n,N,\tilde{Y}}(\omega_k) = A^2_{\tilde{Y}}(\omega_k) + B^2_{\tilde{Y}}(\omega_k)$$

where

$$A_{\tilde{Y}}(\omega_k) = K^{1/2} \sum_{t=1}^{N} \tilde{Y}_t \cos(\omega_k t)$$

and

$$B_{\tilde{Y}}(\omega_k) = K^{1/2} \sum_{t=1}^{N} \tilde{Y}_t \sin(\omega_k t)$$

represent the

$$\{\tilde{Y}_t\}$$

cosine and sine transforms, respectively. Since the periodogram is an even function of the frequency, it is sufficient to estimate the periodogram at the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N} \qquad k = 0, 1, \ldots, \lfloor N/2 \rfloor$$

(Here, $\lfloor a \rfloor$ means the greatest integer less than or equal to a). The routine PFFT (page 825) is used to compute the periodograms of both

$$\{\tilde{X}_t\} \text{ and } \{\tilde{Y}_t\}$$

according to the version specified by the argument IPVER. The computational formula for the cross periodogram is given by

$$I_{n,N,\tilde{X}\tilde{Y}}(\omega_k) = \Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} + i\Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\}$$

where

$$\Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} = A_{\tilde{X}}(\omega_k)A_{\tilde{Y}}(\omega_k) + B_{\tilde{X}}(\omega_k)B_{\tilde{Y}}(\omega_k)$$

and

$$\Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} = A_{\tilde{X}}(\omega_k)B_{\tilde{Y}}(\omega_k) - B_{\tilde{X}}(\omega_k)A_{\tilde{Y}}(\omega_k)$$

The real part of the (modified) cross periodogram represents the 'raw' sample cospectrum and the negative of the imaginary part of the (modified) cross periodogram represents the 'raw' sample quadrature spectrum (Priestley 1981, page 695). The relationship between the cross periodogram and its complex conjugate is given by

$$I_{n,N,\tilde{X}\tilde{Y}}(-\omega_k) \equiv I^*_{n,N,\tilde{X}\tilde{Y}}(\omega_k), \qquad 0 \le \omega_k \le \pi$$

and may be used to recover the cross periodogram at negative frequencies.

CSSWD

Estimates the nonnormalized cross-spectral density of two stationary time series using a spectral window given the time series data.

Required Arguments

X — Vector of length NOBS containing the first stationary time series. (Input)

Y — Vector of length NOBS containing the second stationary time series. (Input)

F — Vector of length NF containing the frequencies at which to evaluate the cross-spectral density estimate. (Input)
The units of F correspond to the scale specified by IFSCAL. The elements of F must be in the range (–π/TINT, π/TINT), inclusive, for IFSCAL = 0 and (–1/(2 * TINT), 1/(2 * TINT)), inclusive, for IFSCAL = 1.

ISWVER — Option for version of the spectral window. (Input)

ISWVER	Action
1	Modified Bartlett
2	Daniell
3	Tukey-Hamming
4	Tukey-Hanning
5	Parzen
6	Bartlett-Priestley

Refer to the "Algorithm" section for further details.

M — Vector of length NM containing the values of the spectral window parameter M. (Input)
For the Parzen spectral window (ISWVER = 5), all values of the spectral window parameters M must be even.

CPM — ($\lfloor N/2 \rfloor + 1$) by 10 matrix containing a summarization of the cross periodogram analysis. (Output)
For $k = 0, 1, \ldots, \lfloor N/2 \rfloor$, the $(k + 1)$-st element of the j-th column of CPM is defined as

Col.	Description
1	Frequency, ω_k where $\omega_k = 2\pi k/N$ for IFSCAL = 0 or $\omega_k = k/N$ for IFSCAL = 1.
2	Period, p_k where $p_k = 2\pi/\omega_k$ for IFSCAL = 0 and $p_k = 1/\omega_k$ for IFSCAL = 1. If $\omega_k = 0$, p_k is set to missing.
3	X periodogram ordinate, $I_X(\omega k)$
4	X cosine transformation coefficient, $A_X(\omega_k)$
5	X sine transformation coefficient, $B_X(\omega_k)$
6	Y periodogram ordinate, $I_Y(\omega_k)$
7	Y cosine transformation coefficient, $A_Y(\omega_k)$
8	Y sine transformation coefficient, $B_Y(\omega_k)$
9	Real part of the XY cross periodogram ordinate $I_{XY}(\omega_k)$.
10	Imaginary part of the XY cross periodogram ordinate $I_{XY}(\omega_k)$.

Note N = NOBS + NPAD.

CSM — NF by (NM * 7 + 2) matrix containing a summarization of the cross-spectral analysis. (Output)
The k-th element of the j-th column of CSM is defined as

Col.	Description
1	Frequency, F(k).
2	Period, p_k where $p_k = 2\pi/$F(k) for IFSCAL = 0 and $p_k = 1/$F(k) for IFSCAL = 1. If F(k) = 0, p_k is set to missing.
3	X spectral density estimate at F(k) using the spectral window parameter M(1).
4	Y spectral density estimate at F(k) using the spectral window parameter M(1).
5	Cospectrum estimate at F(k) using the spectral window parameter M(1).
6	Quadrature spectrum estimate at F(k) using the spectral window parameter M(1).
7	Cross-amplitude spectrum estimate at F(k) using the spectral window parameter M(1).
8	Phase spectrum estimate at F(k) using the spectral window parameter M(1).
9	Coherence estimate at F(k) using the spectral window parameter M(1).
⋮	
NM * 7 + 2	Coherence estimate at F(k) using the spectral window parameter M(NM).

where k = 1, …, NF.

Optional Arguments

NOBS — Number of observations in each stationary time series X and Y. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the cross periodogram and cross-spectral density estimate based on a specified version of a spectral window for a given set of spectral window parameters.

XCNTR — Constant used to center the time series `X`. (Input)
Default: `XCNTR` = the arithmetic mean.

YCNTR — Constant used to center the time series `Y`. (Input)
Default: `YCNTR` = the arithmetic mean.

NPAD — Number of zeroes used to pad each centered time series. (Input)
`NPAD` must be greater than or equal to zero. The length of each centered and padded time series is N = `NOBS` + `NPAD`.
Default: `NPAD` = `NOBS` – 1.

IFSCAL — Option for frequency scale. (Input)
Default: `IFSCAL` = 0.

IFSCAL	Action
0	Frequency in radians per unit time.
1	Frequency in cycles per unit time.

NF — Number of frequencies at which to evaluate the cross-spectral density estimate. (Input)
Default: `NF` = size (`F`,1).

TINT — Time interval at which the series are sampled. (Input)
For a discrete parameter process, usually `TINT` = 1. For a continuous parameter process, `TINT` > 0. `TINT` is used to adjust the cross-spectral density estimate.
Default: `TINT` = 1.0.

NM — Number of spectral window parameters `M` used to compute the cross-spectral density estimate for a given spectral window version. (Input)
`NM` must be greater than or equal to one.
Default: `NM` = size (`M`,1).

LDCPM — Leading dimension of `CPM` exactly as specified in the dimension statement of the calling program. (Input)
`LDCPM` must be greater than or equal to $\lfloor N/2 \rfloor + 1$.
Default: `LDCPM` = size (`CPM`,1).

LDCSM — Leading dimension of CSM exactly as specified in the dimension statement of the calling program. (Input)
LDCSM must be greater than or equal to NF.
Default: LDCSM = size (CSM,1).

FORTRAN 90 Interface

Generic: `CALL CSSWD (X, Y, F, ISWVER, M, CPM, CSM[,…])`

Specific: The specific interface names are S_CSSWD and D_CSSWD.

FORTRAN 77 Interface

Single:
```
CALL CSSWD (NOBS, X, Y, IPRINT, XCNTR, YCNTR, NPAD, IFSCAL,
NF, F, TINT, ISWVER, NM, M, CPM, LDCPM, CSM, LDCSM)
```

Double: The double precision name is DCSSWD.

Example

Consider the Robinson Multichannel Time Series Data (Robinson 1967, page 204) where *X* is the Wölfer sunspot number and *Y* is the northern light activity for the time period from 1770 through 1869. Application of routine CSSWD to these data produces the following results:

```
      USE UMACH_INT
      USE GDATA_INT
      USE CSSWD_INT
      USE WRRRL_INT
      INTEGER    LDCPM, LDCSM, LDRDAT, N, NDRDAT, NF, NM, &
                 NOBS, NPAD
      PARAMETER  (LDRDAT=100, NDRDAT=4, NF=10, NM=2, &
                  NOBS=100, LDCSM=NF, NPAD=NOBS-1, N=NOBS+NPAD,&
                  LDCPM=N/2+1)
!
      INTEGER    I, ISWVER, J, JPT, M(NM), NOUT, NRCOL, NRROW
      REAL       ASIN, CPM(LDCPM,10), CSM(LDCSM,NM*7+2), F(NF), FLOAT, &
                 PI, RDATA(LDRDAT,NDRDAT), TINT, X(NOBS), Y(NOBS)
      CHARACTER  CLABEL1(3)*9, CLABEL2(6)*16, FMT*7, RLABEL(1)*6, &
                 TITLE*80
      INTRINSIC  ASIN, FLOAT
!
      EQUIVALENCE (X(1), RDATA(1,2)), (Y(1), RDATA(1,3))
!
      DATA FMT/'(F10.4)'/
      DATA CLABEL1/' k', 'Frequency', 'Period'/
      DATA CLABEL2/'%/ k', '%/Cospectrum', '%/Quadrature', &
          'Cross%/Amplitude', '%/Phase', '%/Coherence'/
      DATA RLABEL/'NUMBER'/
!                                Initialization
      CALL UMACH (2, NOUT)
      PI = 2.0*ASIN(1.0)
      DO 10  I=1, NF
```

```
         F(I) = PI*FLOAT(I)/FLOAT(NF)
   10 CONTINUE
!                                  Robinson Data
      CALL GDATA (8, RDATA, NRROW, NRCOL)
!                                  Center on arithmetic means
!                                  Frequency in radians per unit time
!                                  Spectral window parameters
      M(1) = 10
      M(2) = 30
!                                  Time interval for discrete data
!                                  Compute cross-spectral density
!                                  using the Parzen window
      ISWVER = 5
      CALL CSSWD (X, Y, F, ISWVER, M, CPM, CSM)
!                                  Print results
      TITLE = 'Cross-Spectral Analysis Using Parzen Window'
      CALL WRRRL (TITLE, CSM, RLABEL, CLABEL1, NF, 2, FMT=FMT)
      DO 20  J=1, NM
         JPT   = 7*(J-1) + 5
         TITLE = '%/Results of the Cross-Spectral Analysis With '// &
                 'Spectral Window Parameter M = '
         WRITE (TITLE(77:78),'(I2)') M(J)
         CALL WRRRL (TITLE, CSM(1:,JPT:), RLABEL, CLABEL2, NF, 5, FMT=FMT)
   20 CONTINUE
!
      END
```

Output

```
Cross-Spectral Analysis Using Parzen Window
         k   Frequency      Period
         1      0.3142     20.0000
         2      0.6283     10.0000
         3      0.9425      6.6667
         4      1.2566      5.0000
         5      1.5708      4.0000
         6      1.8850      3.3333
         7      2.1991      2.8571
         8      2.5133      2.5000
         9      2.8274      2.2222
        10      3.1416      2.0000

Results of the Cross-Spectral Analysis With Spectral Window Parameter M = 10
                                          Cross
         k  Cospectrum  Quadrature   Amplitude       Phase   Coherence
         1    463.5888    -65.9763    468.2600      0.1414      0.2570
         2    286.5450    -75.0209    296.2029      0.2561      0.1710
         3    150.1073    -57.8263    160.8604      0.3677      0.1438
         4     52.9840    -32.3642     62.0866      0.5483      0.0998
         5     21.5435    -15.0888     26.3020      0.6110      0.0794
         6     21.4228     -9.8188     23.5658      0.4298      0.1716
         7     15.7005     -5.3704     16.5936      0.3296      0.2112
         8      8.0118     -1.8887      8.2314      0.2315      0.1272
         9      2.7682      0.2007      2.7754     -0.0724      0.0446
        10      0.5777      0.1008      0.5864     -0.1727      0.0091
```

```
Results of the Cross-Spectral Analysis With Spectral Window Parameter M = 30
                                        Cross
         k  Cospectrum  Quadrature   Amplitude       Phase   Coherence
         1    169.7542   -193.4384    257.3615      0.8505      0.1620
         2    452.6187     32.3813    453.7755     -0.0714      0.2213
         3     94.5221    -90.8159    131.0800      0.7654      0.2629
         4     -0.2096     -6.1127      6.1163      1.6051      0.0019
         5     27.4711    -22.1946     35.3166      0.6796      0.2492
         6     29.1329     -4.0128     29.4080      0.1369      0.3170
         7     11.2058     -9.3403     14.5881      0.6948      0.2594
         8      8.0017      0.8813      8.0501     -0.1097      0.1928
         9     -0.4199      2.2893      2.3275     -1.7522      0.0468
        10      0.5570     -1.0767      1.2123      1.0934      0.0678
```

Comments

1. Workspace may be explicitly provided, if desired, by use of C2SWD/DC2SWD. The reference is:

   ```
   CALL C2SWD (NOBS, X, Y, IPRINT, XCNTR, YCNTR, NPAD,
   IFSCAL, NF, F, TINT, ISWVER, NM, M, CPM, LDCPM, CSM, LDCSM, CX,
   COEF, WFFTC, CPY)
   ```

   The additional arguments are as follows:

   CX — Complex work vector of length N. (Output)

   COEF — Complex work vector of length N. (Output)

   WFFTC — Vector of length $4N + 15$.

   CPY — Vector of length $2N$.

2. The centered and padded time series are defined by

   $\text{CX}(j) = \text{X}(j)\text{XCNTR}$ for $j = 1, \ldots, \text{NOBS}$
   $\text{CX}(j) = 0$ for $j = \text{NOBS} + 1, \ldots, N$
   and
   $\text{CY}(j) = \text{Y}(j)\text{YCNTR}$ for $j = 1, \ldots, \text{NOBS}$
   $\text{CY}(j) = 0$ for $j = \text{NOBS} + 1, \ldots, N$

   where N = NOBS + NPAD.

3. The normalized cross-spectral density estimate is obtained by dividing the nonnormalized cross-spectral density estimate in matrix CSM by the product of the estimated standard deviation of X and the estimated standard deviation of Y.

Description

Routine CSSWD estimates the nonnormalized cross-spectral density function of two jointly stationary time series using a spectral window given a sample of n = NOBS observations $\{X_t\}$ and $\{Y_t\}$ for $t = 1, 2, \ldots, n$.

Let

$$\{\tilde{X}_t\} \text{ for } t = 1, \ldots, N$$

represent the centered and padded data where N = NOBS + NPAD,

$$\tilde{X}_t = \begin{cases} X_t - \hat{\mu}_X & t = 1, \ldots, n \\ 0 & t = (n+1), \ldots, N \end{cases}$$

and

$$\hat{\mu}_X = \text{XCNTR}$$

is determined by

$$\hat{\mu}_X = \begin{cases} \mu_X & \mu_X \text{ known} \\ \dfrac{1}{n}\sum_{t=1}^{n} X_t & \mu_X \text{ unknown} \end{cases}$$

Similarly, let

$$\{\tilde{Y}_t\} \text{ for } t = 1, \ldots, N$$

represent the centered and padded data where

$$\tilde{Y}_t = \begin{cases} Y_t - \hat{\mu}_Y & t = 1, \ldots, n \\ 0 & t = (n+1), \ldots, N \end{cases}$$

and

$$\hat{\mu}_Y = \text{YCNTR}$$

is determined by

$$\hat{\mu}_Y = \begin{cases} \mu_Y & \mu_Y \text{ known} \\ \dfrac{1}{n}\sum_{t=1}^{n} Y_t & \mu_Y \text{ unknown} \end{cases}$$

The modified periodogram of

$$\{\tilde{X}_t\} \text{ for } t = 1, \ldots, N$$

is estimated by

$$I_{n,N,\tilde{X}}(\omega_k) = A^2_{\tilde{X}}(\omega_k) + B^2_{\tilde{X}}(\omega_k)$$

where

$$A_{\tilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N} \tilde{X}_t \cos(\omega_k t)$$

and

$$B_{\tilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{X}_t \sin(\omega_k t)$$

represent the

$$\tilde{X}_t$$

cosine and sine transforms, respectively, and K is the scale factor equal to $1/(2\pi n)$. The modified periodogram of

$$\{\tilde{Y}_t\} \text{ for } t = 1, \ldots, N$$

is estimated by

$$I_{n,N,\tilde{Y}}(\omega_k) = A^2_{\tilde{Y}}(\omega_k) + B^2_{\tilde{Y}}(\omega_k)$$

where

$$A_{\tilde{Y}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{Y}_t \cos(\omega_k t)$$

and

$$B_{\tilde{Y}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{Y}_t \sin(\omega_k t)$$

represent the

$$\tilde{Y}_t$$

cosine and sine transforms, respectively. Since the periodogram is an even function of the frequency, it is sufficient to estimate the periodogram at the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N}, \quad k = 0, 1, \ldots, \lfloor N/2 \rfloor$$

The routine PFFT (page 825) is used to compute the modified periodograms of both

$$\{\tilde{X}_t\} \text{ and } \{\tilde{Y}_t\}$$

The computational formula for the cross periodogram is given by

$$I_{n,N,\tilde{X}\tilde{Y}}(\omega_k) = \Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} + i\Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\}$$

where

$$\Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} = A_{\tilde{X}}(\omega_k)A_{\tilde{Y}}(\omega_k) + B_{\tilde{X}}(\omega_k)B_{\tilde{Y}}(\omega_k)$$

and

$$\Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} = A_{\tilde{X}}(\omega_k)B_{\tilde{Y}}(\omega_k) - B_{\tilde{X}}(\omega_k)A_{\tilde{Y}}(\omega_k)$$

The routine CPFFT (page 855) is used to compute the modified cross periodogram between

$$\{\tilde{X}_t\} \text{ and } \{\tilde{Y}_t\}$$

The nonnormalized spectral density of X_t is estimated by

$$\hat{h}_X(\omega) = \frac{2\pi}{N} \sum_{k=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} I_{n,N,\tilde{X}}(\omega_k) W_n(\omega - \omega_k)$$

and the nonnormalized spectral density of Y_t is estimated by

$$\hat{h}_Y(\omega) = \frac{2\pi}{N} \sum_{k=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} I_{n,N,\tilde{Y}}(\omega_k) W_n(\omega - \omega_k)$$

where the spectral window $W_n(\theta)$ is specified by argument ISWVER. The following spectral windows $W_n(\theta)$ are available.

Modified Bartlett

$$W_n(\theta) = \frac{1}{2\pi M} \left\{ \frac{\sin(M\theta/2)}{\sin(\theta/2)} \right\}^2 = F_M(\theta)$$

where $F_M(\theta)$ corresponds to the Fejér kernel of order M.

Daniell

$$W_n(\theta) = \begin{cases} M/2\pi & -\pi/M \le \theta \le \pi/M \\ 0 & \text{otherwise} \end{cases}$$

Tukey

$$W_n(\theta) = aD_M\left(\theta - \frac{\pi}{M}\right) + (1-2a)D_M(\theta) + aD_M\left(\theta + \frac{\pi}{M}\right) \quad 0 < a \le 0.25$$

where $D_M(\theta)$ represents the Dirichlet kernel. The Tukey-Hamming window is obtained when $a = 0.23$, and the Tukey-Hanning window is obtained when $a = 0.25$.

Parzen

$$W_n(\theta) = \frac{6\pi}{M}\left[F_{M/2}(\theta)\right]^2 \left\{1 - \frac{2}{3}\sin^2(\theta/2)\right\}$$

where M is even. If M is odd, then $M + 1$ is used instead of M in the above formula.

Bartlett-Priestley

$$W_n(\theta) = \begin{cases} \frac{3M}{4\pi}\left\{1 - \left(\frac{M\theta}{\pi}\right)^2\right\} & |\theta| \le \pi / M \\ 0 & |\theta| > \pi / M \end{cases}$$

The argument NM specifies the number of window parameters M and, hence, corresponds to the number of spectral density estimates to be computed for a given spectral window. Note that the same spectral window $W_n(\theta)$ and set of parameters M are used to obtain both

$$\hat{h}_X(\omega) \text{ and } \hat{h}_Y(\omega)$$

The above spectral density formulas assume the data $\{X_t\}$ and $\{Y_t\}$ correspond to a realization of a bivariate discrete-parameter stationary process observed consecutively in time. In this case, the observations are equally spaced in time with interval Δt = TINT equal to one. However, if the data correspond to a realization of a bivariate continuous-parameter stationary process recorded at equal time intervals, then the spectral density estimates must be adjusted for the effect of aliasing. In general, the estimate of $h_X(\omega)$ is given by

$$\hat{h}_X(\omega) = \Delta t\, \hat{h}_X(\omega), \quad |\omega| \le \pi / \Delta t$$

and the estimate of $h_Y(\omega)$ is given by

$$\hat{h}_Y(\omega) = \Delta t\, \hat{h}_Y(\omega), \quad |\omega| \le \pi / \Delta t$$

The nonnormalized spectral density is estimated over the set of frequencies

$$\omega = f_i, \qquad i = 1, \ldots, n_f$$

where n_f = NF. These frequencies are in the scale specified by the argument IFSCAL but are transformed to the scale of radians per unit time for computational purposes. The frequency ω of the desired spectral estimate is assumed to be input in a form already adjusted for the time interval Δt.

The cross-spectral density function is complex-valued in general and may be written in the following form:

$$h_{XY}(\omega) = c_{XY}(\omega) - iq_{XY}(\omega)$$

The *cospectrum* is estimated by

$$\hat{c}_{XY}(\omega) = \frac{2\pi}{N} \sum_{k=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} \Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} W_n(\omega - \omega_k)$$

and the *quadrature spectrum* is estimated by

$$\hat{q}_{XY}(\omega) = \frac{2\pi}{N} \sum_{k=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} \Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} W_n(\omega - \omega_k)$$

Note that the same spectral window $W_n(\theta)$ and window parameter M used to derive

$$\hat{h}_X(\omega) \text{ and } \hat{h}_Y(\omega)$$

are also used to compute

$$\hat{h}_{XY}(\omega)$$

The nonnormalized cross-spectral density estimate is computed over the same set of frequencies as the nonnormalized spectral density estimates with a similar adjustment for Δt.

An equivalent representation of $h_{XY}(\omega)$ is the *polar form* defined by

$$h_{XY}(\omega) = \alpha_{XY}(\omega) e^{i\phi_{XY}(\omega)}$$

The *cross-amplitude spectrum* is estimated by

$$\hat{\alpha}_{XY}(\omega) = \left\{\hat{c}^2_{XY}(\omega) + \hat{q}^2_{XY}(\omega)\right\}^{1/2}$$

and the *phase spectrum* is estimated by

$$\hat{\phi}_{XY}(\omega) = \tan^{-1}\left\{-\hat{q}_{XY}(\omega) / \hat{c}_{XY}(\omega)\right\}$$

Finally, the *coherency spectrum* is estimated by

$$\left|\hat{w}_{XY}(\omega)\right| = \left\{\frac{\hat{c}^2_{XY}(\omega) + \hat{q}^2_{XY}(\omega)}{\hat{h}_X(\omega)\hat{h}_Y(\omega)}\right\}^{1/2}$$

The *coherence* or squared coherency is output.

CSSWP

Estimates the nonnormalized cross-spectral density of two stationary time series using a spectral window given the spectral densities and cross periodogram.

Required Arguments

N — Number of observations in each of the appropriately centered and padded time series `X` and `Y`. (Input)
`N` must be greater than or equal to two.

SX — Vector of length `NF` containing the estimate of the spectral density of the first time series `X`. (Input)

SY – Vector of length `NF` containing the estimate of the spectral density of the second time series `Y`. (Input)

CPREAL — Vector of length $\lfloor N/2 \rfloor + 1$ containing the real part of the cross periodogram between `X` and `Y`. (Input)
The real part of the cross periodogram evaluated at (angular) frequency $w_k = 2\pi k/N$ is given by `CPREAL`$(k + 1)$, $k = 0, 1, \ldots, \lfloor N/2 \rfloor$.

CPIMAG — Vector of length $\lfloor N/2 \rfloor + 1$ containing the imaginary part of the cross periodogram between `X` and `Y`. (Input)
The imaginary part of the cross periodogram evaluated at (angular) frequency $w_k = 2\pi k/N$ is given by `CPIMAG`$(k + 1)$, $k = 0, 1, \ldots, \lfloor N/2 \rfloor$.

F — Vector of length `NF` containing the (angular) frequencies at which the spectral and cross-spectral densities are estimated. (Input)

ISWVER — Option for version of the spectral window. (Input)

`SWVER`	Action
1	Modified Bartlett
2	Daniell
3	Tukey-Hamming
4	Tukey-Hanning
5	Parzen
6	Bartlett-Priestley

Refer to the "Description" section for further details.

M — Spectral window parameter. (Input)
`M` must be greater than or equal to one and less than `N`. For the Parzen spectral window (`ISWVER` = 5), the spectral window parameter `M` must be even.

COSPEC — Vector of length `NF` containing the estimate of the cospectrum. (Output)

QUADRA — Vector of length `NF` containing the estimate of the quadrature spectrum. (Output)

CRAMPL — Vector of length `NF` containing the estimate of the cross-amplitude spectrum. (Output)

PHASE — Vector of length `NF` containing the estimate of the phase spectrum. (Output)

COHERE – Vector of length `NF` containing the estimate of the coherence or squared coherency. (Output)

Optional Arguments

NF — Number of (angular) frequencies. (Input) `NF` must be greater than or equal to one. Default: `NF` = size (`F`,1).

FORTRAN 90 Interface

Generic: `CALL CSSWP (N, SX, SY, CPREAL, CPIMAG, F, ISWVER, M, COSPEC, QUADRA, CRAMPL, PHASE, COHERE [,…])`

Specific: The specific interface names are `S_CSSWP` and `D_CSSWP`.

FORTRAN 77 Interface

Single: `CALL CSSWP (N, SX, SY, CPREAL, CPIMAG, NF, F, ISWVER, M, COSPEC, QUADRA, CRAMPL, PHASE, COHERE)`

Double: The double precision name is `DCSSWP`.

Example

Consider the Robinson Multichannel Time Series Data (Robinson 1967, page 204) where X is the Wölfer sunspot number and Y is the northern light activity for the years 1770 through 1869. Application of routine `CSSWP` to these data produces the following results.

```
      USE IMSL_LIBRARIES
      INTEGER    LDCPM, LDCSM, LDRDAT, N, NDRDAT, NF, NM, &
                 NOBS, NPAD
      PARAMETER  (LDRDAT=100, NDRDAT=4, NF=10, NM=2, &
                 NOBS=100, LDCSM=NF, NPAD=NOBS-1, N=NOBS+NPAD, &
                 LDCPM=N/2+1)
!
      INTEGER    I, IPVER, ISWVER, J, JPT, JST, M(NM), NRCOL, NRROW
      REAL       COHERE(NF), COSPEC(NF), CPIMAG(LDCPM), &
                 CPM(LDCPM,10), CPREAL(LDCPM), CRAMPL(NF), &
                 CSM(LDCSM,7*NM+2), F(NF), FLOAT, P(NF), PHASE(NF), &
                 PI, PX(LDCPM), PY(LDCPM), QUADRA(NF), &
                 RDATA(LDRDAT,NDRDAT), SX(NF), SY(NF), X(NOBS), Y(NOBS)
      CHARACTER  CLABEL1(3)*9, CLABEL2(6)*16, FMT*8, RLABEL(1)*6,&
                 TITLE*80
      INTRINSIC  FLOAT
!
      EQUIVALENCE (X(1), RDATA(1,2)), (Y(1), RDATA(1,3))
      EQUIVALENCE (PX(1), CPM(1,3)), (PY(1), CPM(1,6))
      EQUIVALENCE (CPREAL(1), CPM(1,9)), (CPIMAG(1), CPM(1,10))
      EQUIVALENCE (CSM(1,1), F(1)), (CSM(1,2), P(1))
!
      DATA FMT/'(F12.4)'/
      DATA CLABEL1/' k', 'Frequency', 'Period'/
      DATA CLABEL2/'%/ k', '%/Cospectrum', '%/Quadrature',&
          'Cross%/Amplitude', '%/Phase', '%/Coherence'/
      DATA RLABEL/'NUMBER'/
!                                  Initialization
      PI = 2.0*ASIN(1.0)
      DO 10  I=1, NF
         F(I) = PI*FLOAT(I)/FLOAT(NF)
         P(I) = 2.0*FLOAT(NF)/FLOAT(I)
   10 CONTINUE
```

```
!                                  Robinson Data
      CALL GDATA (8, RDATA, NRROW, NRCOL)
!                                  Center on arithmetic means
!                                  Frequency in radians per unit time
!                                  Modified periodogram version
      IPVER = 1
!                                  Compute cross periodogram
      CALL CPFFT (X, Y, CPM, IPVER=IPVER)
!                                  Spectral window parameters
      M(1) = 10
      M(2) = 30
!                                  Compute cross-spectral density
!                                  using the Parzen window
!
!                                  Print frequency and period
      TITLE = 'Cross-Spectral Analysis Using Parzen Window'
      CALL WRRRL (TITLE, CSM, RLABEL, CLABEL1, NF, 2, FMT=FMT)
      ISWVER = 5
      DO 20  J=1, NM
!                                  Estimate the spectral densities
         CALL SSWP (N, PX, F, M(J), SX, ISWVER=ISWVER)
         CALL SSWP (N, PY, F, M(J), SY, ISWVER=ISWVER)
!                                  Estimate the cross-spectral density
         CALL CSSWP (N, SX, SY, CPREAL, CPIMAG, F, ISWVER, M(J), &
                     COSPEC, QUADRA, CRAMPL, PHASE, COHERE)
!                                  Copy results to output matrices
         JPT = 7*(J-1) + 2
         JST = 7*(J-1) + 5
         CALL SCOPY (NF, SX, 1, CSM(1:,JPT+1), 1)
         CALL SCOPY (NF, SY, 1, CSM(1:,JPT+2), 1)
         CALL SCOPY (NF, COSPEC, 1, CSM(1:,JPT+3), 1)
         CALL SCOPY (NF, QUADRA, 1, CSM(1:,JPT+4), 1)
         CALL SCOPY (NF, CRAMPL, 1, CSM(1:,JPT+5), 1)
         CALL SCOPY (NF, PHASE, 1, CSM(1:,JPT+6), 1)
         CALL SCOPY (NF, COHERE, 1, CSM(1:,JPT+7), 1)
!                                  Print results
         TITLE = '%/Results of the Cross-Spectral Analysis With '// &
                 'Spectral Window Parameter M = '
         WRITE (TITLE(77:78),'(I2)') M(J)
         CALL WRRRL (TITLE, CSM(1:,JST), RLABEL, CLABEL2, NF, 5, FMT=FMT)
   20 CONTINUE
!
      END
```

Output

```
Cross-Spectral Analysis Using Parzen Window
       k     Frequency        Period
       1        0.3142       20.0000
       2        0.6283       10.0000
       3        0.9425        6.6667
       4        1.2566        5.0000
       5        1.5708        4.0000
       6        1.8850        3.3333
       7        2.1991        2.8571
       8        2.5133        2.5000
```

```
       9        2.8274        2.2222
      10        3.1416        2.0000

Results of the Cross-Spectral Analysis With Spectral Window Parameter M = 10
                                          Cross
   k    Cospectrum    Quadrature     Amplitude         Phase     Coherence
   1      463.5888      -65.9763      468.2600        0.1414        0.2570
   2      286.5450      -75.0209      296.2029        0.2561        0.1710
   3      150.1073      -57.8263      160.8604        0.3677        0.1438
   4       52.9840      -32.3642       62.0866        0.5483        0.0998
   5       21.5435      -15.0888       26.3020        0.6110        0.0794
   6       21.4228       -9.8188       23.5658        0.4298        0.1716
   7       15.7005       -5.3704       16.5936        0.3296        0.2112
   8        8.0118       -1.8887        8.2314        0.2315        0.1272
   9        2.7682        0.2007        2.7754       -0.0724        0.0446
  10        0.5777        0.1008        0.5864       -0.1727        0.0091

Results of the Cross-Spectral Analysis With Spectral Window Parameter M = 30
                                          Cross
   k    Cospectrum    Quadrature     Amplitude         Phase     Coherence
   1      169.7542     -193.4384      257.3615        0.8505        0.1620
   2      452.6187       32.3813      453.7755       -0.0714        0.2213
   3       94.5221      -90.8159      131.0800        0.7654        0.2629
   4       -0.2096       -6.1127        6.1163        1.6051        0.0019
   5       27.4711      -22.1946       35.3166        0.6796        0.2492
   6       29.1329       -4.0128       29.4080        0.1369        0.3170
   7       11.2058       -9.3403       14.5881        0.6948        0.2594
   8        8.0017        0.8813        8.0501       -0.1097        0.1928
   9       -0.4199        2.2893        2.3275       -1.7522        0.0468
  10        0.5570       -1.0767        1.2123        1.0934        0.0678
```

Comments

1. The periodograms of X and Y and cross periodogram between X and Y may be computed using the routine CPFFT (page 855). The spectral densities of X and Y may then be estimated using any of the routines SSWD (page 831), SWED (page 845), SSWP (page 840), or SWEP (page 851). Thus, different window types and/or weight sequences may be used to estimate the spectral and cross-spectral densities given either the series or their periodograms. Note that use of the modified periodograms and modified cross periodogram ensures that the scale of the spectral and cross-spectral densities and their estimates is equivalent.

2 The time sampling interval, TINT, is assumed to be equal to one. This assumption is appropriate for discrete parameter processes. The adjustment for continuous parameter processes (TINT > 0.0) involves multiplication of the frequency vector F by 1/TINT and multiplication of the spectral and cross-spectral density estimates by TINT.

3. To convert the frequency scale from radians per unit time to cycles per unit time, multiply F by $1/(2\pi)$.

Description

Routine CSSWP estimates the nonnormalized cross-spectral density function of two jointly stationary time series using a spectral window given the modified cross-periodogram and spectral densities of the appropriately centered and padded data

$$\{\tilde{X}_t\} \text{ and } \{\tilde{Y}_t\}$$

for $t = 1, \ldots, N$.

The routine CPFFT (page 855) may be used to compute the modified periodograms

$$I_{n,N,\tilde{X}}(\omega_k) \text{ and } I_{n,N,\tilde{Y}}(\omega_k)$$

and cross periodogram

$$I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)$$

over the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N}, \quad k = 0, 1, \ldots, \lfloor N/2 \rfloor$$

(Here, $\lfloor a \rfloor$ means the greatest integer less than or equal to a.) Either routine SSWP (page 840) or routine SWEP (page 851) may be applied to the periodograms to obtain nonnormalized spectral density estimates

$$\hat{h}_X(\omega) \text{ and } \hat{h}_Y(\omega)$$

over the set of frequencies

$$\omega = f_i, \qquad i = 1, \ldots, n_f$$

where n_f = NF. These frequencies are in the scale of radians per unit time. The time sampling interval Δt is assumed to be equal to one. Note that the spectral window or weight sequence used to compute

$$\hat{h}_X(\omega)$$

may differ from that used to compute

$$\hat{h}_Y(\omega)$$

The cross-spectral density function is complex-valued in general and may be written as

$$h_{XY}(\omega) = c_{XY}(\omega) - iq_{XY}(\omega)$$

The *cospectrum* is estimated by

$$\hat{c}_{XY}(\omega) = \frac{2\pi}{N} \sum_{k=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} \Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} W_n(\omega - \omega_k)$$

and the *quadrature spectrum* is estimated by

$$\hat{q}_{XY}(\omega) = \frac{2\pi}{N} \sum_{k=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} \Im\left\{ I_{n,N,\tilde{X}\tilde{Y}}(\omega_k) \right\} W_n(\omega - \omega_k)$$

where the spectral window $W_n(\theta)$ is specified by argument `ISWVER`. The following spectral windows $W_n(\theta)$ are available.

Modified Bartlett

$$W_n(\theta) = \frac{1}{2\pi M} \left\{ \frac{\sin(M\theta/2)}{\sin(\theta/2)} \right\}^2 = F_M(\theta)$$

where $F_M(\theta)$ corresponds to the Fejér kernel of order M.

Daniell

$$W_n(\theta) = \begin{cases} M/2\pi & -\pi/M \le \theta \le \pi/M \\ 0 & \text{otherwise} \end{cases}$$

Tukey

$$W_n(\theta) = aD_M\left(\theta - \frac{\pi}{M}\right) + (1-2a)D_M(\theta) + aD_M\left(\theta + \frac{\pi}{M}\right), \quad 0 < a \le 0.25$$

where $D_M(\theta)$ represents the Dirichlet kernel. The Tukey-Hamming window is obtained when $a = 0.23$, and the Tukey-Hanning window is obtained when $a = 0.25$.

Parzen

$$W_n(\theta) = \frac{6\pi}{M}\left[F_{M/2}(\theta)\right]^2 \left\{1 - \frac{2}{3}\sin^2(\theta/2)\right\}$$

where M is even. If M is odd, then $M + 1$ is used instead of M in the above formula.

Bartlett-Priestley

$$W_n(\theta) = \begin{cases} \frac{3M}{4\pi}\left\{1 - \left(\frac{M\theta}{\pi}\right)^2\right\} & |\theta| \le \pi/M \\ 0 & |\theta| > \pi/M \end{cases}$$

Only one window parameter M may be specified so that only one estimate of $h_{XY}(\omega)$ is computed. The nonnormalized cross-spectral density estimate is computed over the same set of frequencies as the nonnormalized spectral density estimates discussed above. However, the particular spectral window used to compute

$$\hat{h}_{XY}(\omega)$$

need not correspond to either the spectral window or the weight sequence used to compute either

$$\hat{h}_X(\omega) \text{ or } \hat{h}_Y(\omega)$$

An equivalent representation of $h_{XY}(\omega)$ is the *polar form* defined by

$$h_{XY}(\omega) = \alpha_{XY}(\omega)e^{i\phi_{XY}(\omega)}$$

The *cross-amplitude* spectrum is estimated by

$$\hat{\alpha}_{XY}(\omega) = \left\{\hat{c}^2_{XY}(\omega) + \hat{q}^2_{XY}(\omega)\right\}^{1/2}$$

and the *phase spectrum* is estimated by

$$\hat{\phi}_{XY}(\omega) = \tan^{-1}\left\{-\hat{q}_{XY}(\omega)/\hat{c}_{XY}(\omega)\right\}$$

Finally, the *coherency spectrum* is estimated by

$$\left|\hat{w}_{XY}(\omega)\right| = \left\{\frac{\hat{c}^2_{XY}(\omega) + \hat{q}^2_{XY}(\omega)}{\hat{h}_X(\omega)\hat{h}_Y(\omega)}\right\}^{1/2}$$

The *coherence* or squared coherency is output.

CSWED

Estimates the nonnormalized cross-spectral density of two stationary time series using a weighted cross periodogram given the time series data.

Required Arguments

X — Vector of length NOBS containing the first stationary time series. (Input)

Y — Vector of length NOBS containing the second stationary time series. (Input)

F — Vector of length NF containing the frequencies at which to evaluate the cross-spectral density estimate. (Input)
The units of F correspond to the scale specified by IFSCAL. The elements of F must be in the range ($-\pi$/TINT, π/TINT) inclusive, for IFSCAL = 0 and (-1/(2 * TINT), 1/(2 * TINT)) inclusive, for IFSCAL = 1.

WT — Vector of length NWT containing the weights used to smooth the periodogram. (Input)
The actual weights are the values in WT normalized to sum to 1 with the current periodogram ordinate taking the middle weight for NWT odd or the weight to the right of the middle for NWT even.

CPM — ($\lfloor N/2 \rfloor + 1$) by 10 matrix containing a summarization of the cross periodogram analysis. (Output)
For $k = 0, 1, \ldots, \lfloor N/2 \rfloor$, the $(k + 1)$-st element of the j-th column of CPM is defined as

Col.	Description
1	Frequency, ω_k where $\omega_k = 2\pi k/N$ for IFSCAL = 0 or $\omega_k = k/N$ for IFSCAL = 1
2	Period, p_k where $p_k = 2\pi/\omega k$ for IFSCAL = 0 and $p_k = 1/\omega_k$ for IFSCAL = 1. If $\omega_k = 0$, p_k is set to missing.
3	X periodogram ordinate, $I_X(\omega_k)$
4	X cosine transformation coefficient, $A_X(\omega_k)$
5	X sine transformation coefficient, $B_X(\omega_k)$
6	Y periodogram ordinate, $I_Y(\omega_k)$
7	Y cosine transformation coefficient, $A_Y(\omega_k)$
8	Y sine transformation coefficient, $B_Y(\omega_k)$
9	Real part of the XY cross periodogram ordinate $I_{XY}(\omega_k)$.
10	Imaginary part of the XY cross periodogram ordinate $I_{XY}(\omega_k)$.

CSM — NF by 9 matrix containing a summarization of the cross-spectral analysis. (Output) The k-th element of the j-th column of CSM is defined as

Col.	Description
1	Frequency, F(k).
2	Period, p_k where $p_k = 2\pi/$F(k) for IFSCAL = 0 and $p_k = 1/$F(k) for IFSCAL = 1. If F(k) = 0, p_k is set to missing.
3	X spectral density estimate at F(k) using the specified relative weights contained in WT.
4	Y spectral density estimate at F(k) using the specified relative weights contained in WT.
5	Co-spectrum estimate at F(k) using the specified relative weights contained in WT.
6	Quadrature spectrum estimate at F(k) using the specified relative weights contained in WT.

7 Cross-amplitude spectrum estimate at F(k).

8 Phase spectrum estimate at F(k).

9 Coherence estimate at F(k).
where k = 1, …, NF.

Optional Arguments

NOBS — Number of observations in each stationary time series X and Y. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the periodogram, cosine and sine transformations of each centered and padded time series, the real and imaginary components of the cross periodogram, and the cross-spectral density estimate based on a specified weight sequence.

XCNTR — Constant used to center the time series X. (Input)
Default: XCNTR = the arithmetic mean.

YCNTR — Constant used to center the time series Y. (Input)
Default: YCNTR = the arithmetic mean.

NPAD — Number of zeroes used to pad each centered time series. (Input)
NPAD must be greater than or equal to zero. The length of each centered and padded time series is N = NOBS + NPAD.
Default: NPAD = NOBS – 1.

IFSCAL — Option for frequency scale. (Input)
Default: IFSCAL = 0.

IFSCAL	Action
0	Frequency in radians per unit time.
1	Frequency in cycles per unit time.

NF — Number of frequencies at which to evaluate the cross-spectral density estimate. (Input)
Default: NF = size (F,1).

TINT — Time interval at which the series are sampled. (Input)
For a discrete parameter process, usually TINT = 1.0. For a continuous parameter process, TINT > 0.0. TINT is used to adjust the cross-spectral density estimate.
Default: TINT = 1.0.

NWT — Number of weights. (Input)
NWT must be greater than or equal to one.
Default: NWT= size (WT,1).

LDCPM — Leading dimension of CPM exactly as specified in the dimension statement of the calling program. (Input)
LDCPM must be greater than or equal to $\lfloor N/2 \rfloor + 1$.
Default: LDCPM = size (CPM,1).

LDCSM — Leading dimension of CSM exactly as specified in the dimension statement of the calling program. (Input)
LDCSM must be greater than or equal to NF.
Default: LDCSM = size (CSM,1).

FORTRAN 90 Interface

Generic: `CALL CSWED (X, Y, F, WT, CPM, CSM [,…])`

Specific: The specific interface names are S_CSWED and D_CSWED.

FORTRAN 77 Interface

Single:
```
CALL CSWED (NOBS, X, Y, IPRINT, XCNTR, YCNTR, NPAD, IFSCAL,
NF, F, TINT, NWT, WT, CPM, LDCPM, CSM, LDCSM)
```

Double: The double precision name is DCSWED.

Example

Consider the Robinson Multichannel Time Series Data (Robinson 1967, page 204) where X is the Wölfer sunspot number and Y is the northern light activity for the years 1770 through 1869. Application of routine CSWED to these data produces the following results.

```
      USE IMSL_LIBRARIES
      INTEGER    LDCPM, LDCSM, LDRDAT, N, NDRDAT, NF, NOBS, &
                 NPAD, NWT
      PARAMETER  (LDRDAT=100, NDRDAT=4, NF=10, NOBS=100, &
                 NWT=7, LDCSM=NF, NPAD=NOBS-1, N=NPAD+NOBS, &
                 LDCPM=N/2+1)
!
      INTEGER    I, NRCOL, NRROW
      REAL       CPM(LDCPM,10), CSM(LDCSM,9), F(NF), FLOAT, PI, &
                 RDATA(LDRDAT,NDRDAT), WT(NWT), X(NOBS), Y(NOBS)
      CHARACTER  CLABEL1(5)*24, CLABEL2(6)*16, FMT*7, RLABEL(1)*6, &
                 TITLE1*32, TITLE2*40
```

```
      INTRINSIC  FLOAT
!
      EQUIVALENCE (X(1), RDATA(1,2)), (Y(1), RDATA(1,3))
!
      DATA WT/1.0, 2.0, 3.0, 4.0, 3.0, 2.0, 1.0/
      DATA FMT/'(F12.4)'/
      DATA CLABEL1/'%/%/ k', '%/%/Frequency', '%/%/Period', &
          'Spectral%/Estimate%/of X', 'Spectral%/Estimate%/of Y'/
      DATA CLABEL2/'%/ k', '%/Cospectrum', '%/Quadrature', &
          'Cross%/Amplitude', '%/Phase', '%/Coherence'/
      DATA RLABEL/'NUMBER'/
      DATA TITLE1/'Results of the Spectral Analyses'/
      DATA TITLE2/'%/Results of the Cross-Spectral Analysis'/
!                                Initialization
      PI = 2.0*ASIN(1.0)
      DO 10  I=1, NF
         F(I) = PI*FLOAT(I)/FLOAT(NF)
   10 CONTINUE
!                                Robinson data
      CALL GDATA (8, RDATA, NRROW, NRCOL)
!                                Center on arithmetic means
!                                Frequency in radians per unit time
!                                Time interval for discrete data
!                                Compute the cross periodogram
      CALL CSWED (X, Y, F, WT, CPM, CSM)
!                                Print results
      CALL WRRRL (TITLE1, CSM, RLABEL, CLABEL1, NF, 4, FMT=FMT)
      CALL WRRRL (TITLE2, CSM(1:,5), RLABEL, CLABEL2, NF, 5, FMT=FMT)
!
      END
```

Output

Results of the Spectral Analyses

k	Frequency	Period	Spectral Estimate of X	Spectral Estimate of Y
1	0.3142	20.0000	116.9550	1315.8370
2	0.6283	10.0000	1206.6086	1005.1219
3	0.9425	6.6667	84.8369	317.2589
4	1.2566	5.0000	55.2120	270.2111
5	1.5708	4.0000	46.5748	115.6768
6	1.8850	3.3333	12.4050	250.0125
7	2.1991	2.8571	7.0934	82.6773
8	2.5133	2.5000	3.4091	62.3267
9	2.8274	2.2222	5.6828	12.8970
10	3.1416	2.0000	4.0346	17.6441

Results of the Cross-Spectral Analysis

k	Cospectrum	Quadrature	Cross Amplitude	Phase	Coherence
1	94.0531	-254.0125	270.8659	1.2162	0.4767
2	702.5118	21.9823	702.8557	-0.0313	0.4073
3	70.2379	-31.4431	76.9547	0.4209	0.2200
4	-1.8715	-36.1639	36.2123	1.6225	0.0879
5	36.6366	-18.5925	41.0843	0.4696	0.3133

```
 6       32.7071       -6.6569       33.3776        0.2008        0.3592
 7        9.4887       -9.1692       13.1950        0.7683        0.2969
 8        9.2534       -0.3000        9.2583        0.0324        0.4034
 9       -0.5568        2.9455        2.9977       -1.7576        0.1226
10        1.7640       -1.8321        2.5433        0.8043        0.0909
```

Comments

1. Workspace may be explicitly provided, if desired, by use of C2WED/DC2WED. The reference is:

   ```
   CALL C2WED (NOBS, X, Y, IPRINT, XCNTR, YCNTR, NPAD,
   IFSCAL, NF, F, TINT, NWT, WT, CPM, LDCPM, CSM, LDCSM, CWK,
   COEFWK, WFFTC, CPY)
   ```

   The additional arguments are as follows:

   CWK — Complex work vector of length *N*. (Output)

   COEFWK — Complex work vector of length *N*. (Output)

   WFFTC — Vector of length $4N + 15$.

   CPY — Vector of length $2N$.

2. The normalized cross-spectral density estimate is obtained by dividing the nonnormalized cross-spectral density estimate in matrix CSM by the product of the estimated standard deviation of X and the estimated standard deviation of Y.

Description

Routine CSWED estimates the nonnormalized cross-spectral density function of two jointly stationary time series using a fixed sequence of weights given a sample of n = NOBS observations $\{X_t\}$ and $\{Y_t\}$ for $t = 1, 2, \ldots, n$. Let

$$\left\{\tilde{X}_t\right\}$$

for $t = 1, \ldots, N$ represent the centered and padded data where N = NOBS + NPAD,

$$\tilde{X}_t = \begin{cases} X_t - \hat{\mu}_X & t = 1, \ldots, n \\ 0 & t = (n+1), \ldots, N \end{cases}$$

and

$$\hat{\mu}_X = \text{XCNTR}$$

is determined by

$$\hat{\mu}_X = \begin{cases} \mu_{X,} & \mu_X \text{ known} \\ \dfrac{1}{n}\sum_{t=1}^{n} X_{t,} & \mu_X \text{ unknown} \end{cases}$$

Similarly, let

$$\{\tilde{Y}_t\}$$

for $t = 1, \ldots, N$ represent the centered and padded data where

$$\tilde{Y}_t = \begin{cases} Y_{t,} - \hat{\mu}_Y, & t = 1, \ldots, n \\ 0, & t = (n+1), \ldots, N \end{cases}$$

and

$$\hat{\mu}_Y = \text{ YCNTR}$$

is determined by

$$\hat{\mu}_Y = \begin{cases} \mu_{Y,} & \mu_Y \text{ known} \\ \dfrac{1}{n}\sum_{t=1}^{n} Y_{t,} & \mu_Y \text{ unknown} \end{cases}$$

The modified periodogram of

$$\{\tilde{X}_t\}$$

for $t = 1, \ldots, N$ is estimated by

$$I_{n,N,\tilde{X}}(\omega_k) = A^2_{\tilde{X}}(\omega_k) + B^2_{\tilde{X}}(\omega_k)$$

where

$$A_{\tilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N} \tilde{X}_t \cos(\omega_k t)$$

and

$$B_{\tilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N} \tilde{X}_t \sin(\omega_k t)$$

represent the

$$\tilde{X}_t$$

cosine and sine transforms, respectively, and K is the scale factor equal to $1/(2\pi n)$. The modified periodogram of $\{Y_t\}$ for $t = 1, \ldots, N$ is estimated by

$$I_{n,N,\tilde{Y}}(\omega_k) = A^2_{\tilde{Y}}(\omega_k) + B^2_{\tilde{Y}}(\omega_k)$$

where

$$A_{\tilde{Y}}(\omega_k) = K^{1/2} \sum_{t=1}^{N} \tilde{Y}_t \cos(\omega_k t)$$

and

$$B_{\tilde{Y}}(\omega_k) = K^{1/2} \sum_{t=1}^{N} \tilde{Y}_t \sin(\omega_k t)$$

represent the

$$\tilde{Y}_t$$

cosine and sine transforms, respectively. Since the periodogram is an even function of the frequency, it is sufficient to estimate the periodogram at the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N}, \quad k = 0, 1, \ldots, \lfloor N/2 \rfloor$$

(Here, $\lfloor a \rfloor$ means the greatest integer less than or equal to a). The routine PFFT (page 825) is used to compute the modified periodograms of both

$$\{\tilde{X}_t\} \text{ and } \{\tilde{Y}_t\}$$

The computational formula for the cross periodogram is given by

$$I_{n,N,\tilde{X}\tilde{Y}}(\omega_k) = \Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} + i\Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\}$$

where

$$\Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} = A_{\tilde{X}}(\omega_k)A_{\tilde{Y}}(\omega_k) + B_{\tilde{X}}(\omega_k)B_{\tilde{Y}}(\omega_k)$$

and

$$\Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} = A_{\tilde{X}}(\omega_k)B_{\tilde{Y}}(\omega_k) - B_{\tilde{X}}(\omega_k)A_{\tilde{Y}}(\omega_k)$$

The routine CPFFT (page 855) is used to compute the modified cross periodogram between

$$\{\tilde{X}_t\} \text{ and } \{\tilde{Y}_t\}$$

The nonnormalized spectral density of X_t is estimated by

$$\hat{h}_X(\omega) = \sum_j \omega_j I_{n,N,\tilde{X}}(\omega_{k,j})$$

and the nonnormalized spectral density of Y_t is estimated by

$$\hat{h}_Y(\omega) = \sum_j \omega_j I_{n,N,\tilde{Y}}(\omega_{k,j})$$

where

$$\omega_{k,j} = \frac{2\pi\{k(\omega)+j\}}{N}$$

and $k(\omega)$ is the integer such that $\omega_{k,0}$ is closest to ω. The sequence of m = NWT weights $\{w_j\}$ for $j = -\lfloor m/2 \rfloor, \ldots, (m - \lfloor m/2 \rfloor - 1)$ satisfies $\sum_j w_j = 1$. These weights are fixed in the sense that they do not depend on the frequency ω at which to estimate the spectral density. Usually, m is odd with the weights symmetric about the middle weight w_0. If m is even, the weight to the right of the middle is considered w_0. The argument WT may contain relative weights since they are normalized to sum to one in the actual computations. The above spectral density formulas assume the data $\{X_t\}$ and $\{Y_t\}$ correspond to a realization of a bivariate discrete-parameter stationary process observed consecutively in time. In this case, the observations are equally spaced in time with interval Δt = TINT equivalent to one. However, if the data correspond to a realization of a bivariate continuous-parameter stationary process recorded at equal time intervals, then the spectral density estimates must be adjusted for the effect of aliasing. In general, the estimate of $h_X(\omega)$ is given by

$$\hat{h}_X(\omega) = \Delta t\, \hat{h}_X(\omega), \quad |\omega| \le \pi / \Delta t$$

and the estimate of $h_Y(\omega)$ is given by

$$\hat{h}_Y(\omega) = \Delta t\, \hat{h}_Y(\omega), \quad |\omega| \le \pi / \Delta t\ .$$

The nonnormalized spectral density is estimated over the set of frequencies

$$\omega = f_i, \qquad i = 1, \ldots, n_f$$

where nf = NF. These frequencies are in the scale specified by the argument IFSCAL but are transformed to the scale of radians per unit time for computational purposes. The frequency ω of the desired spectral estimate is assumed to be input in a form already adjusted for the time interval Δt. The cross-spectral density function is complex-valued in general and may be written as

$$h_{XY}(\omega) = c_{XY}(\omega) - iq_{XY}(\omega)$$

The *cospectrum* is estimated by

$$\hat{c}_{XY}(\omega) = \sum_j w_j \Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_{k,j})\right\}$$

and the *quadrature spectrum* is estimated by

$$\hat{q}_{XY}(\omega) = \sum_j w_j \Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_{k,j})\right\}$$

Note that the same sequence of weights $\{w_j\}$ used to estimate

$$\hat{h}_X(\omega) \text{ and } \hat{h}_Y(\omega)$$

is used to estimate

$$\hat{c}_{XY}(\omega) \text{ and } \hat{q}_{XY}(\omega)$$

The nonnormalized cross-spectral density estimate is computed over the same set of frequencies as the nonnormalized spectral density estimates discussed above with a similar adjustment for Δt. An equivalent representation of $h_{XY}(\omega)$ is the polar form defined by

$$h_{XY}(\omega) = \alpha_{XY}(\omega)e^{i\phi_{XY}(\omega)}$$

The *cross-amplitude* spectrum is estimated by

$$\hat{\alpha}_{XY}(\omega) = \left\{\hat{c}^2_{XY}(\omega) + \hat{q}^2_{XY}(\omega)\right\}^{1/2}$$

and the *phase spectrum* is estimated by

$$\hat{\phi}_{XY}(\omega) = \tan^{-1}\left\{-\hat{q}_{XY}(\omega)/\hat{c}_{XY}(\omega)\right\}$$

Finally, the *coherency spectrum* is estimated by

$$\left|\hat{w}_{XY}(\omega)\right| = \left\{\frac{\hat{c}^2_{XY}(w) + \hat{q}^2_{XY}(\omega)}{\hat{h}_X(\omega)\hat{h}_Y(\omega)}\right\}^{1/2}$$

The *coherence* or squared coherency is output.

CSWEP

Estimates the nonnormalized cross-spectral density of two stationary time series using a weighted cross periodogram given the spectral densities and cross periodogram.

Required Arguments

N — Number of observations in each of the appropriately centered and padded time series `X` and `Y`. (Input)
`N` must be greater than or equal to two.

SX — Vector of length `NF` containing the estimate of the spectral density of the first time series `X`. (Input)

SY — Vector of length `NF` containing the estimate of the spectral density of the second time series `Y`. (Input)

CPREAL – Vector of length $\lfloor N/2 \rfloor + 1$ containing the real part of the cross periodogram between `X` and `Y`. (Input)
The real part of the cross periodogram evaluated at (angular) frequency $\omega_k = 2\pi k/N$ is given by `CPREAL`$(k + 1)$, $k = 0, 1, \ldots, \lfloor N/2 \rfloor$.

CPIMAG — Vector of length $\lfloor N/2 \rfloor + 1$ containing the imaginary part of the cross periodogram between `X` and `Y`. (Input)
The imaginary part of the cross periodogram evaluated at (angular) frequency $\omega_k = 2\pi k/N$ is given by `CPIMAG`$(k + 1)$, $k = 0, 1, \ldots, \lfloor N/2 \rfloor$.

F — Vector of length NF containing the (angular) frequencies at which the spectral density is estimated. (Input)

WT — Vector of length NWT containing the weights used to smooth the periodogram. (Input) The actual weights are the values in WT normalized to sum to 1 with the current periodogram ordinate taking the middle weight for NWT odd or the weight to the right of the middle for NWT even.

COSPEC — Vector of length NF containing the estimate of the cospectrum. (Output)

QUADRA — Vector of length NF containing the estimate of the quadrature spectrum. (Output)

CRAMPL — Vector of length NF containing the estimate of the cross-amplitude spectrum. (Output)

PHASE — Vector of length NF containing the estimate of the phase spectrum. (Output)

COHERE — Vector of length NF containing the estimate of the coherence. (Output)

Optional Arguments

NF — Number of (angular) frequencies. (Input)
F must be greater than or equal to one.
Default: NF = size (F,1).

NWT — Number of weights. (Input)
NWT must be greater than or equal to one.
Default: NWT = size (WT,1).

FORTRAN 90 Interface

Generic:
```
CALL CSWEP (N, SX, SY, CPREAL, CPIMAG, F, WT, COSPEC,
QUADRA, CRAMPL, PHASE, COHERE [,…])
```

Specific: The specific interface names are S_CSWEP and D_CSWEP.

FORTRAN 77 Interface

Single:
```
CALL CSWEP (N, SX, SY, CPREAL, CPIMAG, NF, F, NWT, WT,
COSPEC, QUADRA, CRAMPL, PHASE, COHERE)
```

Double: The double precision name is DCSWEP.

Example

Consider the Robinson Multichannel Time Series Data (Robinson 1967, page 204) where X is the Wölfer sunspot number and Y is the northern light activity for the years 1770 through 1869. Application of routine CSWEP to these data produces the following results.

```
      USE IMSL_LIBRARIES
      INTEGER    LDCPM, LDCSM, LDRDAT, N, NDRDAT, NF, NOBS, &
                 NPAD, NWT
      PARAMETER  (LDRDAT=100, NDRDAT=4, NF=10, NOBS=100, &
                 NWT=7, LDCSM=NF, NPAD=NOBS-1, N=NOBS+NPAD, &
                 LDCPM=N/2+1)
!
      INTEGER    I, IPVER, NROW, NVAR
      REAL       COHERE(NF), COSPEC(NF), CPIMAG(LDCPM), &
                 CPM(LDCPM,10), CPREAL(LDCPM), CRAMPL(NF), &
                 CSM(LDCSM,9), F(NF), FLOAT, PHASE(NF), PI, PX(LDCPM), &
                 PY(LDCPM), QUADRA(NF), RDATA(LDRDAT,NDRDAT), &
                 SX(NF), SY(NF), WT(NWT), X(NOBS), Y(NOBS)
      CHARACTER  CLABEL1(5)*24, CLABEL2(6)*16, FMT*8, RLABEL(1)*6, &
                 TITLE1*32, TITLE2*40
      INTRINSIC  FLOAT

!
      EQUIVALENCE (X(1), RDATA(1,2)), (Y(1), RDATA(1,3))
      EQUIVALENCE (PX(1), CPM(1,3)), (PY(1), CPM(1,6))
      EQUIVALENCE (CPREAL(1), CPM(1,9)), (CPIMAG(1), CPM(1,10))
!
      DATA WT/1.0, 2.0, 3.0, 4.0, 3.0, 2.0, 1.0/
      DATA FMT/'(F12.4)'/
      DATA CLABEL1/'%/%/ k', '%/%/Frequency', '%/%/Period', &
          'Spectral%/Estimate%/of X', 'Spectral%/Estimate%/of Y'/
      DATA CLABEL2/'%/ k', '%/Cospectrum', '%/Quadrature', &
          'Cross%/Amplitude', '%/Phase', '%/Coherence'/
      DATA RLABEL/'NUMBER'/
      DATA TITLE1/'Results of the Spectral Analyses'/
      DATA TITLE2/'%/Results of the Cross-Spectral Analysis'/
!                                Initialization
      PI = 2.0*ASIN(1.0)
      DO 10  I=1, NF
         F(I) = PI*FLOAT(I)/FLOAT(NF)
         CALL SCOPY (NF, F, 1, CSM(1:,1), 1)
         CSM(I,2) = 2.0*FLOAT(NF)/FLOAT(I)
   10 CONTINUE
!                                Robinson data
      CALL GDATA (8, RDATA, NROW, NVAR)
!                                Center on arithmetic means
!                                Frequency in radians per unit time
!                                Modified periodogram version
      IPVER = 1
!                                Compute the cross periodogram
      CALL CPFFT (X, Y, CPM, IPVER=IPVER)
!                                Estimate the spectral densities
      CALL SWEP (N, PX, F, WT, SX)
      CALL SWEP (N, PY, F, WT, SY)
```

```
!                                  Estimate the cross-spectral density
      CALL CSWEP (N, SX, SY, CPREAL, CPIMAG, F, WT, COSPEC, &
                  QUADRA, CRAMPL, PHASE, COHERE)
!                                  Print results
!
!                                  Copy results to output matrices
      CALL SCOPY (NF, SX, 1, CSM(1:,3), 1)
      CALL SCOPY (NF, SY, 1, CSM(1:,4), 1)
      CALL SCOPY (NF, COSPEC, 1, CSM(1:,5), 1)
      CALL SCOPY (NF, QUADRA, 1, CSM(1:,6), 1)
      CALL SCOPY (NF, CRAMPL, 1, CSM(1:,7), 1)
      CALL SCOPY (NF, PHASE, 1, CSM(1:,8), 1)
      CALL SCOPY (NF, COHERE, 1, CSM(1:,9), 1)
!                                  Call printing routines
      CALL WRRRL (TITLE1, CSM, RLABEL, CLABEL1, NF, 4, FMT=FMT)
      CALL WRRRL (TITLE2, CSM(1:,5), RLABEL, CLABEL2, NF, 5, FMT=FMT)
!
      END
```

Output

Results of the Spectral Analyses

k	Frequency	Period	Spectral Estimate of X	Spectral Estimate of Y
1	0.3142	20.0000	116.9550	1315.8370
2	0.6283	10.0000	1206.6086	1005.1219
3	0.9425	6.6667	84.8369	317.2589
4	1.2566	5.0000	55.2120	270.2111
5	1.5708	4.0000	46.5748	115.6768
6	1.8850	3.3333	12.4050	250.0125
7	2.1991	2.8571	7.0934	82.6773
8	2.5133	2.5000	3.4091	62.3267
9	2.8274	2.2222	5.6828	12.8970
10	3.1416	2.0000	4.0346	17.6441

Results of the Cross-Spectral Analysis

k	Cospectrum	Quadrature	Cross Amplitude	Phase	Coherence
1	94.0531	-254.0125	270.8659	1.2162	0.4767
2	702.5118	21.9823	702.8557	-0.0313	0.4073
3	70.2379	-31.4431	76.9547	0.4209	0.2200
4	-1.8715	-36.1639	36.2123	1.6225	0.0879
5	36.6366	-18.5925	41.0843	0.4696	0.3133
6	32.7071	-6.6569	33.3776	0.2008	0.3592
7	9.4887	-9.1692	13.1950	0.7683	0.2969
8	9.2534	-0.3000	9.2583	0.0324	0.4034
9	-0.5568	2.9455	2.9977	-1.7576	0.1226
10	1.7640	-1.8321	2.5433	0.8043	0.0909

Comments

1. The periodograms of X and Y and cross periodogram between X and Y may be computed via the routine CPFFT (page 855). The spectral densities of X and Y may then be estimated using any of the routines SSWD (page 831), SWED (page 845), SSWP (page

840), or SWEP (page 851). Thus, different window types and/or weight sequences may be used to estimate the spectral and cross-spectral densities given either the series or their periodograms. Note that use of the modified periodograms and modified cross periodogram ensures that the scales of the spectral and cross-spectral densities and their estimates are equivalent.

2 The time sampling interval, TINT, is assumed to be equal to one. This assumption is appropriate for discrete parameter processes. The adjustment for continuous parameter processes (TINT > 0.0) involves multipication of the frequency vector F by 1/TINT and multiplication of the spectral and cross-spectral density estimates by TINT.

3. To convert the frequency scale from radians per unit time to cycles per unit time, multiply F by 1/(2π).

Description

Routine CSWEP estimates the nonnormalized cross-spectral density function of two jointly stationary time series using a fixed sequence of weights given the modified cross-periodogram and spectral densities of the appropriately centered and padded data

$$\{\tilde{X}_t\}$$

and

$$\{\tilde{Y}_t\}$$

for $t = 1, \ldots, N$. The routine CPFFT (page 855) may be used to compute the modified periodograms

$$I_{n,N,\tilde{X}}(\omega_k) \text{ and } I_{n,N,\tilde{Y}}(\omega_k)$$

and cross-periodogram

$$I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)$$

over the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N}, \quad k = 0, 1, \ldots, \lfloor N/2 \rfloor$$

(Here, $\lfloor a \rfloor$ means the greatest integer less than or equal to a.) Either routine SSWP (page 840) or routine SWEP (page 851) may be applied to the periodograms to obtain nonnormalized spectral density estimates

$$\hat{h}_X(\omega) \text{ and } \hat{h}_Y(\omega)$$

over the set of frequencies

$$\omega = f_i, \qquad i = 1, \ldots, n_f$$

where n_f = NF. These frequencies are in the scale of radians per unit time. The time sampling interval Δt is assumed to be equal to one. Note that the spectral window or weight sequence used to compute

$$\hat{h}_X(\omega)$$

may differ from that used to compute

$$\hat{h}_Y(\omega)$$

The cross-spectral density function is complex-valued in general and may be written as

$$h_{XY}(\omega) = c_{XY}(\omega) - iq_{XY}(\omega)$$

The *cospectrum* is estimated by

$$\hat{c}_{XY}(\omega) = \sum_j w_j \Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_{k,j})\right\}$$

and the *quadrature spectrum* is estimated by

$$\hat{q}_{XY}(\omega) = \sum_j w_j \Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_{k,j})\right\}$$

where

$$\omega_{k,j} = \frac{2\pi\{k(\omega) + j\}}{N}$$

and $k(\omega)$ is the integer such that $\omega_{k,0}$ is closest to ω. The sequence of m = NWT weights $\{w_j\}$ for $j = -\lfloor m/2 \rfloor, \ldots, (m - \lfloor m/2 \rfloor - 1)$ satisfies $\sum_j w_j = 1$. These weights are fixed in the sense that they do not depend on the frequency ω at which to estimate $h_{XY}(\omega)$. Usually, m is odd with the weights symmetric about the middle weight w_0. If m is even, the weight to the right of the middle is considered w_0. The argument WT may contain relative weights since they are normalized to sum to one in the actual computations. The nonnormalized cross-spectral density estimate is computed over the same set of frequencies as the nonnormalized spectral density estimates. However, the particular weight sequence used to compute

$$\hat{h}_{XY}(\omega)$$

need not correspond to either the weight sequence or spectral window used to compute either

$$\hat{h}_X(\omega) \text{ or } \hat{h}_Y(\omega)$$

An equivalent representation of $h_{XY}(\omega)$ is the *polar form* defined by

$$h_{XY}(\omega) = \alpha_{XY}(\omega)e^{i\phi_{XY}(\omega)}$$

The *cross-amplitude* spectrum is estimated by

$$\hat{\alpha}_{XY}(\omega) = \left\{\hat{c}^2_{XY}(\omega) + \hat{q}^2_{XY}(\omega)\right\}^{1/2}$$

and the *phase spectrum* is estimated by

$$\hat{\phi}_{XY}(\omega) = \tan^{-1}\left\{-\hat{q}_{XY}(\omega)/\hat{c}_{XY}(\omega)\right\}$$

Finally, the *coherency spectrum* is estimated by

$$\left|\hat{w}_{XY}(\omega)\right| = \left\{\frac{\hat{c}_{XY}^2(\omega) + \hat{q}_{XY}^2(\omega)}{\hat{h}_X(\omega)\hat{h}_Y(\omega)}\right\}^{1/2}$$

The coherence or squared coherency is output.

Chapter 9: Covariance Structures and Factor Analysis

Routines

Usage Notes

Notation that is consistently used throughout this chapter is given in the following table. The FORTRAN equivalent of the symbols used are also given.

Notation Used

Symbol	FORTRAN Symbol	Meaning
p	NVAR	Number of variables in the observed variables
k	NF	Number of factors
Σ	COV	Population (or sample) covariance (correlation) matrix
A	A	Unrotated factor loadings
B	B	Rotated factor loadings
T	T	Factor rotation matrix
	TI	Image transformation matrix
β	SCOEF	Factor score coefficients

The routines in this chapter can generally be used for one or more of several purposes. Among these purposes are the following:

1. Data description: The information in the data is summarized by the factor loadings or by the eigenvectors and eigenvalues.
2. Data reduction: The information in a multivariate sample is reduced to a much smaller number of factors or principal components.
3. Variable clustering: The principal component coefficients or factor loadings lead to a grouping (clustering) of the variables.
4. Model building: Linear models relating the variables to the factors or principal components are estimated. Hypothesis tests may be used to obtain parsimonious and/or other descriptions of the data.

Principal Components

The idea in principal components is to find a small number of linear combinations of the original variables that maximize the variance accounted for in the original data. This amounts to an eigensystem analysis of the covariance (or correlation) matrix. In addition to the eigensystem analysis, routine PRINC (page 901) computes standard errors for the eigenvalues. Correlations of the original variables with the principal component scores are also computed.

The computation of common principal components via routine KPRIN (page 905) is equivalent to finding the "eigenvectors" that best simultaneously diagonalize one or more variance-covariance

matrices. For only one input variance-covariance matrix, the vectors computed actually are the eigenvectors of the matrix.

Factor Analysis

Factor analysis and principal component analysis, while quite different in assumptions, often serve the same ends. Unlike principal components in which linear combinations yielding the highest possible variances are obtained, factor analysis generally obtains linear combinations of the observed variables according to a model relating the observed variables to hypothesized underlying factors, plus a random error term called the unique error or uniqueness. In factor analysis, the unique errors associated with each variable are usually assumed to be independent of the factors. In addition, in the common factor model, the unique errors are assumed to be mutually independent. The factor analysis model is

$$x - \mu = \Lambda f + e$$

where x is the p vector of observed variables, μ is the p vector of variable means, Λ is the $p \times k$ matrix of factor loadings, f is the k vector of hypothesized underlying random factors, and e is the p vector of hypothesized unique random errors.

Because much of the computation in factor analysis was originally done by hand or was expensive on early computers, quick (but dirty) algorithms that made the calculations possible were developed. One result is the many factor extraction methods available today. Generally speaking, in the exploratory or model building phase of a factor analysis, a method of factor extraction that is not computationally intensive (such as principal components, principal factor, or image analysis) is used. If desired, a computationally intensive method is then used to obtain (what is hoped will be) the final factors.

In exploratory factor analysis, the unrotated factor loadings obtained from the factor extraction are generally transformed (rotated) to simplify the interpretation of the factors. Rotation is possible because of the overparameterization in the factor analysis model. The method used for rotation may result in factors that are independent (orthogonal rotations) or correlated (oblique rotations). Prior information may be available (or hypothesized) in which case a Procrustes rotation could be used. When no prior information is available, an analytic rotation can be performed.

Once the factor loadings have been extracted and rotated (if desired), estimates for the hypothesized underlying factors can be computed. First, one of several available methods in routine `FCOEF` (page 944) is used to compute the factor score coefficients. Routine `FSCOR` (page 949) is then called with these factor score coefficients to compute the factor scores.

The steps generally used in a factor analysis are summarized as follows:

Steps in a Factor Analysis

Step 1

Calculate Covariance (Correlation) Matrix
IMSL routine CORVC (page 314)

Step 2

Initial Factor Extraction
FACTR, page 909

Step 3

Factor Rotation

Orthogonal		Oblique	
No Prior Info. FROTA, page 918	Prior Info. FOPCS, page 921	No Prior Info. FPRMX, page 927 FDOBL, page 924 FHARR, page 932 FGCRF, page 936	Prior Info. FPRMX, page 927

Step 3a

Image Analysis
FIMAG, page 940

Step 3b

Factor Variances
FRVAR, page 942

Step 4

Factor Coefficients
FCOEF, page 944

Step 5

Factor Scores
FSCOR, page 949

Independence of Sets of Variables and Canonical Correlation Analysis

Routine MVIND (page 954) computes the Wilks likelihood-ratio test of independence among several sets of variables. Routines CANCR (page 957) and CANVC (page 971) compute some other tests of independence between exactly two sets of variables. Routine CANCR uses the raw data as input while CANVC uses the sample variance-covariance matrix. Furthermore, CANCR and CANVC perform a canonical correlation analysis. Since CANCR uses a better algorithm in terms of numerical stability (it does not compute the covariance matrix), CANCR should be used if possible. However, if the raw data is not available, or if there is too much data for all of it to reside in memory at the same time, or if multiple canonical correlation analyses are to be performed based

on the same pre-computed sample variance-covariance matrix, then the use of CANVC may be necessary. Canonical correlation analysis is useful for characterizing the independent linear statistical relationships that exist between the two sets of variables. This involves computing linear combinations of the variables in the two separate sets and their associated correlation. The coefficients of the variables in the linear combinations are called the "canonical coefficients," and the correlations are called "canonical correlations." Evaluation of the linear combinations using the canonical coefficients gives the "canonical scores." Routine CANCR computes the canonical scores for the observed data. Routine FSCOR can be used to compute the canonical scores for new data or for the observed data if CANVC is used.

PRINC

Computes principal components from a variance-covariance matrix or a correlation matrix.

Required Arguments

NDF — Number of degrees of freedom in COV. (Input)
If NDF is less than or equal to 0, 100 degrees of freedom are assumed.

COV — NVAR by NVAR matrix containing the covariance or correlation matrix. (Input)
Only the upper triangular part of COV is referenced.

EVAL — Vector of length NVAR containing the eigenvalues from matrix COV ordered from largest to smallest. (Output)

Optional Arguments

NVAR — Order of matrix COV. (Input)
Default: NVAR = size (COV,2).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

ICOV — Covariance/Correlation matrix option parameter. (Input)
ICOV = 0 means that a covariance matrix is input. Otherwise, a correlation matrix is input.
Default: ICOV = 0.

PCT — Vector of length NVAR containing the cumulative percent of the total variance explained by each principal component. (Output)

STD — Vector of length NVAR containing the estimated asymptotic standard errors of the eigenvalues. (Output)

EVEC — NVAR by NVAR matrix containing the eigenvectors of COV, stored columnwise. (Output)
Each vector is normalized to have Euclidean length equal to the value one. Also, the

sign of each vector is set so that the largest component in magnitude (the first of the largest if there are ties) is made positive.

LDEVEC — Leading dimension of EVEC exactly as specified in the dimension statement in the calling program. (Input)
Default: LDEVEC = size(EVEC, 1).

A — NVAR by NVAR matrix containing the correlations of the principal components (the columns) with the observed/standardized variables (the rows). (Output)
If ICOV = 0, then the correlations are with the observed variables. Otherwise, the correlations are with the standardized (to a variance of 1.0) variables. In the principal component model for factor analysis, matrix A is the matrix of unrotated factor loadings.

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size(A, 1).

FORTRAN 90 Interface

Generic: `CALL PRINC (NDF, COV, EVAL [,…])`

Specific: The specific interface names are `S_PRINC` and `D_PRINC`.

FORTRAN 77 Interface

Single:
```
CALL PRINC (NDF, NVAR, COV, LDCOV, ICOV, EVAL, PCT, STD,
EVEC, LDEVEC, A, LDA)
```

Double: The double precision name is `DPRINC`.

Example

Principal components are computed for a nine-variable matrix.

```
      USE PRINC_INT
      USE WRRRN_INT
      INTEGER    ICOV, LDA, LDCOV, LDEVEC, NDF, NVAR
      PARAMETER  (ICOV=1, LDA=9, LDCOV=9, LDEVEC=9, NDF=100, NVAR=9)
!
      REAL       A(LDA,NVAR), COV(LDCOV,NVAR), EVAL(NVAR), &
                 EVEC(LDEVEC,NVAR), PCT(NVAR), STD(NVAR)
!
      DATA COV/&
       1.000, 0.523, 0.395, 0.471, 0.346, 0.426, 0.576, 0.434, 0.639, &
       0.523, 1.000, 0.479, 0.506, 0.418, 0.462, 0.547, 0.283, 0.645, &
       0.395, 0.479, 1.000, 0.355, 0.270, 0.254, 0.452, 0.219, 0.504, &
       0.471, 0.506, 0.355, 1.000, 0.691, 0.791, 0.443, 0.285, 0.505, &
       0.346, 0.418, 0.270, 0.691, 1.000, 0.679, 0.383, 0.149, 0.409, &
       0.426, 0.462, 0.254, 0.791, 0.679, 1.000, 0.372, 0.314, 0.472, &
```

```
       0.576, 0.547, 0.452, 0.443, 0.383, 0.372, 1.000, 0.385, 0.680, &
       0.434, 0.283, 0.219, 0.285, 0.149, 0.314, 0.385, 1.000, 0.470, &
       0.639, 0.645, 0.504, 0.505, 0.409, 0.472, 0.680, 0.470, 1.000/
!
      CALL PRINC (NDF, COV, EVAL, ICOV=ICOV, PCT=PCT, STD=STD, &
                  EVEC=EVEC, A=A)
!
      CALL WRRRN ('EVAL', EVAL, 1, NVAR, 1)
      CALL WRRRN ('PCT', PCT, 1, NVAR, 1)
      CALL WRRRN ('STD', STD, 1, NVAR, 1)
      CALL WRRRN ('EVEC', EVEC)
      CALL WRRRN ('A', A)
      END
```

Output

```
                                EVAL
    1       2       3       4       5       6       7       8       9
4.677   1.264   0.844   0.555   0.447   0.429   0.310   0.277   0.196

                                PCT
    1       2       3       4       5       6       7       8       9
0.520   0.660   0.754   0.816   0.865   0.913   0.947   0.978   1.000

                                STD
     1        2        3        4        5        6        7        8
0.6498   0.1771   0.0986   0.0879   0.0882   0.0890   0.0944   0.0994

     9
0.1113

                                     EVEC
         1        2        3        4        5        6        7        8
1   0.3462  -0.2354   0.1386  -0.3317  -0.1088   0.7974   0.1735  -0.1240
2   0.3526  -0.1108  -0.2795  -0.2161   0.7664  -0.2002   0.1386  -0.3032
3   0.2754  -0.2697  -0.5585   0.6939  -0.1531   0.1511   0.0099  -0.0406
4   0.3664   0.4031   0.0406   0.1196   0.0017   0.1152  -0.4022  -0.1178
5   0.3144   0.5022  -0.0733  -0.0207  -0.2804  -0.1796   0.7295   0.0075
6   0.3455   0.4553   0.1825   0.1114   0.1202   0.0696  -0.3742   0.0925
7   0.3487  -0.2714  -0.0725  -0.3545  -0.5242  -0.4355  -0.2854  -0.3408
8   0.2407  -0.3159   0.7383   0.4329   0.0861  -0.1969   0.1862  -0.1623
9   0.3847  -0.2533  -0.0078  -0.1468   0.0459  -0.1498  -0.0251   0.8521

         9
1  -0.0488
2  -0.0079
3  -0.0997
4   0.7060
5   0.0046
6  -0.6780
7  -0.1089
8   0.0505
9   0.1225

                                     A
         1        2        3        4        5        6        7        8
```

```
1   0.7487  -0.2646   0.1274  -0.2471  -0.0728   0.5224   0.0966  -0.0652
2   0.7625  -0.1245  -0.2568  -0.1610   0.5124  -0.1312   0.0772  -0.1596
3   0.5956  -0.3032  -0.5133   0.5170  -0.1024   0.0990   0.0055  -0.0214
4   0.7923   0.4532   0.0373   0.0891   0.0012   0.0755  -0.2240  -0.0620
5   0.6799   0.5646  -0.0674  -0.0154  -0.1875  -0.1177   0.4063   0.0039
6   0.7472   0.5119   0.1677   0.0830   0.0804   0.0456  -0.2084   0.0487
7   0.7542  -0.3051  -0.0666  -0.2641  -0.3505  -0.2853  -0.1589  -0.1794
8   0.5206  -0.3552   0.6784   0.3225   0.0576  -0.1290   0.1037  -0.0854
9   0.8319  -0.2848  -0.0072  -0.1094   0.0307  -0.0981  -0.0140   0.4485

          9
1  -0.0216
2  -0.0035
3  -0.0442
4   0.3127
5   0.0021
6  -0.3003
7  -0.0482
8   0.0224
9   0.0543
```

Comments

Informational errors

Type	Code	
3	1	Because NDF is zero or less, 100 degrees of freedom will be used.
3	2	One or more eigenvalues much less than zero are computed. The matrix COV is not nonnegative definite. In order to continue computations of STD and A, these eigenvalues are treated as zero.

Description

Routine PRINC finds the principal components of a set of variables from a sample covariance or correlation matrix. The characteristic roots, characteristic vectors, standard errors for the characteristic roots, and the correlations of the principal component scores with the original variables are computed. Principal components obtained from correlation matrices are the same as principal components obtained from standardized (to unit variance) variables.

The principal component scores are the elements of the vector $y = \Gamma^T x$ where Γ is the matrix whose columns are the characteristic vectors (eigenvectors) of the sample covariance (or correlation) matrix and x is the vector of observed (or standardized) random variables. The variances of the principal component scores are the characteristic roots (eigenvalues) of the the covariance (correlation) matrix.

Asymptotic variances for the characteristic roots were first obtained by Girshick (1939) and are given more recently by Kendall, Stuart, and Ord (1983, page 331). These variances are computed either for covariance matrices (ICOV = 0) or for correlation matrices (ICOV $\neq$ 0).

The correlations of the principal components with the observed (or standardized) variables are given in the matrix A. When the principal components are obtained from a correlation matrix, A

is the same as the matrix of unrotated factor loadings obtained for the principal components model for factor analysis.

KPRIN

Maximums likelihood or least-squares estimates for principal components from one or more matrices.

Required Arguments

COV — NVAR by NVAR by NMAT array containing the NMAT covariance or correlation matrices. (Input)
Only the upper triangular elements of each matrix are referenced.

ANI — Vector of length NMAT containing the number of observations in each of the covariance matrices. (Input)
For least-squares estimation, the square root of ANI(I) is the weight to be used for the I-th covariance matrix. Since the elements of ANI are used as weights, they need not be integers.

EVEC — NVAR by NVAR matrix containing the estimated principal components. (Output)
Each column of EVEC contains a principal component vector (an "eigenvector"). The ordering of the eigenvectors is such that the sum of the corresponding eigenvalues are ordered from largest to smallest. Each vector is normalized to have Euclidean length equal to the value one.

Optional Arguments

NVAR — Number of variables in each matrix. (Input)
NVAR must be 2 or greater.
Default: NVAR = size (COV,2).

NMAT — Number of matrices. (Input)
Default: NMAT = size (COV,3).

LDCOV — Leading and second dimensions of COV exactly as specified in the dimension statement of the calling program. (Input)
The first two dimensions of COV must be equal.
Default: LDCOV = size (COV,1).

IMETH — Method to be used for extracting the estimated principal components. (Input)
For IMETH = 0, maximum likelihood estimation is used. For IMETH = 1, least-squares estimation is used.
Default: IMETH = 0.

LDEVEC — Leading dimension of EVEC exactly as specified in the dimension statement in the calling program. (Input)
Default: LDEVEC = size (EVEC,1).

FORTRAN 90 Interface

Generic: `CALL KPRIN (COV, ANI, EVEC [,…])`

Specific: The specific interface names are `S_KPRIN` and `D_KPRIN`.

FORTRAN 77 Interface

Single: `CALL KPRIN (NVAR, NMAT, COV, LDCOV, ANI, IMETH, EVEC, LDEVEC)`

Double: The double precision name is `DKPRIN`.

Example

The following example is taken from Flury and Constantine (1985). It involves two 4 by 4 covariance matrices. The two covariance matrices are given as:

$$\begin{pmatrix} 1.3 & -0.3 & -0.6 & 0 \\ -0.3 & 2.1 & 0 & -0.6 \\ -0.6 & 0 & 2.9 & -0.3 \\ 0 & -0.6 & -0.3 & 3.7 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}$$

```
      USE KPRIN_INT
      USE WRRRN_INT
!                                  SPECIFICATIONS FOR PARAMETERS

      INTEGER    LDCOV, LDEVEC, NGROUP, NVAR
      PARAMETER  (LDCOV=4, LDEVEC=4, NGROUP=2, NVAR=4)
!
      REAL       ANI(NGROUP), COV(LDCOV,LDCOV,NGROUP), &
      EVEC(LDEVEC,NVAR)
!
      DATA COV/1.3, -0.3, -0.6, 0, -0.3, 2.1, 0, -0.6, -0.6, 0, 2.9, &
      -0.3, 0, -0.6, -0.3, 3.7, 1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 3, &
      0, 0, 0, 0, 4/
!
      ANI(1) = 1
      ANI(2) = 1
!
      CALL KPRIN (COV, ANI, EVEC)
```

```
!
      CALL WRRRN ('EVEC', EVEC)
!
      END
```

Output

```
                EVEC
         1         2         3         4
1   0.9743   -0.1581   -0.1581    0.0257
2   0.1581    0.9743   -0.0257   -0.1581
3   0.1581   -0.0257    0.9743   -0.1581
4   0.0257    0.1581    0.1581    0.9743
```

Comments

1. Workspace may be explicitly provided, if desired, by use of K2RIN/DK2RIN. The reference is:

   ```
   CALL K2RIN (NVAR, NMAT, COV, LDCOV, ANI, IMETH, EVEC, LDEVEC, H,
   AUX, BOLD, G, T)
   ```

   The additional arguments are as follows:

   H — Work vector of length NVAR^2 * NMAT. On return from K2RIN, H may be treated as an array dimensioned as H(NVAR, NVAR, NMAT). Each NVAR by NVAR matrix in H is computed as (EVEC)T * COV(I) * EVEC, i.e., H contains the "eigenvalues" and the "residuals" for each covariance matrix. Here, COV(I) is the I-th covariance matrix.

   AUX — Work vector of length max(3 * NMAT, NVAR^2).

   BOLD — Work vector of length NVAR^2.

   G — Work vector of length 2 * NVAR.

   T — Work vector of length 4 * NMAT.

2. Informational errors

Type	Code	
3	1	Convergence did not occur within 50 iterations. Convergence is assumed.
4	2	An input matrix is singular. Singular input matrices are not allowed in maximum likelihood estimation.

3. If user specified initial estimates for EVEC are desired (and argument error checking is not needed), then the routine K3RIN (DK3RIN) may be used. The calling sequence is

   ```
   CALL K3RIN (NVAR, NMAT, COV, LDCOV, ANI, IMETH, EVEC, LDEVEC,
   AUX, BOLD, G, T)
   ```

On input, EVEC contains the initial estimates of the common principal components (EVEC must be an orthogonal matrix). On output, COV contains the NMAT matrices (EVEC)T * COV (I) * EVEC. The user should be wary of stationary points in the likelihood if K3RIN is used.

Description

Routine KPRIN is the IMSL version of the F-G diagonalization routine of Flury and Constantine (1985) with modifications as discussed by Clarkson (1988a, 1988b). Let k = NMAT. Routine KPRIN computes the common principal components of $k \geq 1$ covariance (or correlation) matrices using either a least-squares or a maximum likelihood criterion. Computing common principal components is equivalent to finding the "eigenvectors" that best simultaneously diagonalize k symmetric matrices. (Note that when $k = 1$, both least-squares and maximum likelihood estimation yield the eigenvectors of the input matrix.) See Flury (1988) for applications of common principal components.

The algorithm proceeds by accumulating simple rotations as follows: Initial estimates of the diagonalizing principal components are found as the eigenvectors of the summed covariance matrices (unless K3RIN is used, see Comment 3 above). The covariance matrices are then pre- and post-multiplied by the initial estimates to obtain approximately diagonal matrices. Let

$$w_l^{ij}$$

denote the l-th 2×2 matrix obtained from the (i, j), (i, i), and (j, j) elements of S_l, where S_l is the l-th covariance matrix in COV. Then, for each i and j, a Jacobi rotation is found and applied such that the least-squares or maximum likelihood criterion is optimized over all k matrices in

$$w_l^{ij}$$

An iteration consists of computing and applying a Jacobi rotation for all $p(p-1)/2$ possible off-diagonal elements (i, j) where p = NVAR. A maximum of 50 iterations are allowed before convergence. Convergence is assumed when the maximum change in the any element in the eigenvectors from one iteration to the next is less than 0.0001.

Let Γ denote the current estimates of the optimizing principal components. Then, maximizing the multivariate normal likelihood is equivalent to minimizing the criterion

$$L = \prod_{i=1}^{k} \left(\frac{\det \hat{\Sigma}_i}{\det S_i} \right)^{n_i}$$

where S_i is the i-th covariance matrix, n_i is its degrees of freedom, and

$$\hat{\Sigma}_i$$

is the estimate of the covariance matrix under the common principal components model.

During each Jacobi iteration, an optimal orthogonal matrix T_{ij} is found that rotates the two vectors in columns i and j of Γ. When restricted to T_{ij}, the criterion above becomes

$$L = \prod_{i=1}^{k} \left(\frac{\det(\text{diag}(T_{ij}^T \Gamma^T S_i \Gamma) T_{ij})}{\det(S_i)} \right)^{n_i}$$

Γ is updated as ΓT_{ij}. When convergence has been reached (the maximum change in Γ is less than 0.0001), Γ contains the optimizing principal components. Initially, Γ is taken as the eigenvectors of the matrix $\sum_i S_i$.

In least-squares estimation, the matrices T_{ij} are found such that the sum of the squared off-diagonal elements in the resulting "diagonalized" matrices are minimized. That is, T_{ij} is found to minimize

$$v_{ij}^T v_{ij}$$

where v_{ij} is the vector of length k containing the off-diagonal elements in the matrices

$$w_l^{ij}$$

See Flury and Gautschi (1986) for further details on the general algorithm, especially in maximum likelihood estimation. See Clarkson (1988b) for details of the least-squares algorithm.

If the "residual" matrices $\Gamma^T S_l \Gamma$ are desired, they may be obtained in the work vector `H` returned from `K2RIN` or from the matrix `COV` returned from `K3RIN`. If the least-squares criterion is needed, it is easily computed as the sum of the squared off-diagonal elements in `H` (or `COV`). To compute the likelihood ratio criterion, the eigenvalues of each matrix in `COV` first need to be computed. Denote the eigenvalues from the l-th matrix by λ_{lj}, and let

$$\hat{\lambda}_{lj}$$

be the eigenvalues obtained under the common principal component model (and returned as the diagonal elements of `H` or, from `K3RIN`, `COV`). Then, the log-likelihood-ratio statistic for testing,

$$H_o : \Gamma^T \sum_l \Gamma$$

is diagonal, $l = 1, \ldots, k$, is computed as:

$$\ell = \sum_{l=1}^{k} \sum_{j=1}^{p} n_l \log \frac{\hat{\lambda}_{lj}}{\lambda_{lj}}$$

The distribution of ℓ under H_0 is asymptotically χ^2 with $(k-1)p(p-1)/2$ degrees of freedom (see Flury 1984).

FACTR

Extracts initial factor loading estimates in factor analysis.

Required Arguments

COV — `NVAR` by `NVAR` matrix containing the variance-covariance or correlation matrix. (Input)

NF — Number of factors in the model. (Input)

UNIQ — Vector of length NVAR containing the unique variances. (Input/Output, if INIT = 1; output, otherwise)
If INIT = 1, UNIQ contains the initial estimates of these variances on input. On output, UNIQ contains the estimated unique variances. For IMTH = 0, the unique variances are assumed to be known and are not changed from the input values when INIT = 1.

A — NVAR by NF matrix of unrotated factor loadings. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (COV,2).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

IMTH — Method used to obtain the estimates. (Input)
Default: IMTH = 0.

IMTH	Method
0	Principal component (principal component model) or principal factor (common factor model). If INIT = 1 and UNIQ contains zeros, then this option results in the principal component method. Otherwise, the principal factor method is used.
1	Unweighted least squares (common factor model).
2	Generalized least squares (common factor model).
3	Maximum likelihood (common factor model).
4	Image factor analysis (common factor model).
5	Alpha factor analysis (common factor model).

NDF — Number of degrees of freedom in COV. (Input)
NDF is not required when IMTH = 0, 1, or 4. NDF defaults to 100 if NDF = 0.
Default: NDF = 0.

INIT — Method used to obtain initial estimates of the unique variances. (Input)
Default: INIT = 0.

INIT **Method**

0 Initial estimates are taken as the constant 1 – NF/(2 * NVAR) divided by the diagonal elements of the inverse of COV.

1 Initial estimates are input in vector UNIQ.

MAXIT — Maximum number of iterations in the iterative procedure. (Input)
Typical for methods 1 to 3 is 30, while 60 is typical for method 5. MAXIT is not referenced when IMTH = 0 or 4.
Default: MAXIT = 30.

MAXSTP — Maximum number of step halvings allowed during any one iteration. (Input)
Typical is 8. MAXSTP is not referenced when IMTH = 0, 4, or 5.
Default: MAXSTP = 8.

EPS — Convergence criterion used to terminate the iterations. (Input)
For methods 1 to 3, convergence is assumed when the relative change in the criterion is less than EPS. For method 5, convergence is assumed when the maximum change (relative to the variance) of a uniqueness is less than EPS. EPS is not referenced when IMTH = 0 or 4. EPS = 0.0001 is typical.
Default: EPS = 0.0001.

EPSE — Convergence criterion used to switch to exact second derivatives. (Input)
When the largest relative change in the unique standard deviation vector is less than EPSE, exact second derivative vectors are used. Typical is 0.1. EPSE is not referenced when IMTH = 0, 4, or 5.
Default: EPSE = 0.1.

IPRINT — Printing option. (Input)
If IPRINT = 0, then no printing is performed. If IPRINT = 1, then printing of the final results is performed. If IPRINT = 2, then printing of an iteration summary and the final results is performed.
Default: IPRINT = 0.

LDA — Leading dimension of A exactly as specified in the dimension statement of the calling program. (Input)
Default: LDA = size (A,1).

EVAL — Vector of length NVAR containing the eigenvalues of the matrix from which the factors were extracted. (Output)
If IMTH = 5, then the first NF positions of EVAL contain the ALPHA coefficients. Note that EVAL does not usually contain eigenvalues for matrix COV.

STAT — Vector of length 6 containing some output statistics. (Output)

I	STAT(I)
1	Value of the function minimum.
2	Tucker reliability coefficient.
3	Chi-squared test statistic for testing that NF common factors are adequate for the data.
4	Degrees of freedom in chi-squared. This is computed as $((NVAR - NF)^2 - NVAR - NF)/2$.
5	Probability of a greater chi-squared statistic.
6	Number of iterations.

STAT is not used when IMTH = 0, 4, or 5.

DER — Vector of length NVAR containing the parameter updates when convergence was reached (or the iterations terminated). (Output)

FORTRAN 90 Interface

Generic: `CALL FACTR (COV, NF, UNIQ, A [,…])`

Specific: The specific interface names are `S_FACTR` and `D_FACTR`.

FORTRAN 77 Interface

Single:
```
CALL FACTR (NVAR, COV, LDCOV, NF, IMTH, NDF, INIT, MAXIT,
MAXSTP, EPS, EPSE, IPRINT, UNIQ, A, LDA, EVAL, STAT, DER)
```

Double: The double precision name is `DFACTR`.

Example

The following data were originally analyzed by Emmett (1949). There are 211 observations on 9 variables. Following Lawley and Maxwell (1971), three factors will be obtained by the method of maximum likelihood.

```
      USE FACTR_INT
      INTEGER    IMTH,IPRINT, LDA, LDCOV, MAXSTP, NDF, NF, NVAR
      REAL       EPS, EPSE
      PARAMETER  (EPS=0.000001, EPSE=0.01, IMTH=3, IPRINT=1, &
                  LDA=9, LDCOV=9, MAXSTP=10, NDF=210, NF=3, NVAR=9)
!
      REAL       A(LDA,NF), COV(LDCOV,NVAR), DER(NVAR), EVAL(NVAR), &
                 STAT(6), UNIQ(NVAR)
!
```

```
      DATA COV/ &
       1.000, 0.523, 0.395, 0.471, 0.346, 0.426, 0.576, 0.434, 0.639, &
       0.523, 1.000, 0.479, 0.506, 0.418, 0.462, 0.547, 0.283, 0.645, &
       0.395, 0.479, 1.000, 0.355, 0.270, 0.254, 0.452, 0.219, 0.504, &
       0.471, 0.506, 0.355, 1.000, 0.691, 0.791, 0.443, 0.285, 0.505, &
       0.346, 0.418, 0.270, 0.691, 1.000, 0.679, 0.383, 0.149, 0.409, &
       0.426, 0.462, 0.254, 0.791, 0.679, 1.000, 0.372, 0.314, 0.472, &
       0.576, 0.547, 0.452, 0.443, 0.383, 0.372, 1.000, 0.385, 0.680, &
       0.434, 0.283, 0.219, 0.285, 0.149, 0.314, 0.385, 1.000, 0.470, &
       0.639, 0.645, 0.504, 0.505, 0.409, 0.472, 0.680, 0.470, 1.000/
!
      CALL FACTR (COV, NF, UNIQ, A, IMTH=IMTH, MAXSTP=MAXSTP, EPS=EPS,&
                  EPSE=EPSE, IPRINT=IPRINT, NDF=NDF, EVAL=EVAL, &
                  STAT=STAT, DER=DER)
      END
```

Output

```
                        Unique Error Variances
     1        2        3        4        5        6        7        8
0.4505   0.4271   0.6166   0.2123   0.3805   0.1769   0.3995   0.4615

     9
0.2309

     Unrotated Loadings
          1        2        3
1   0.6642  -0.3209   0.0735
2   0.6888  -0.2471  -0.1933
3   0.4926  -0.3022  -0.2224
4   0.8372   0.2924  -0.0354
5   0.7050   0.3148  -0.1528
6   0.8187   0.3767   0.1045
7   0.6615  -0.3960  -0.0777
8   0.4579  -0.2955   0.4913
9   0.7657  -0.4274  -0.0117

                            Eigenvalues
    1       2       3       4       5       6       7       8       9
0.063   0.229   0.541   0.865   0.894   0.974   1.080   1.117   1.140

                            STAT
     1             2             3             4             5
0.0350        1.0000        7.1494       12.0000        0.8476

     6
5.0000

                     Final Parameter Updates
          1             2             3             4             5
2.02042E-07   2.95010E-07   1.80908E-07   6.38808E-08   2.00809E-07

          6             7             8             9
1.48762E-07   1.73797E-08   3.95484E-07   1.42415E-07
```

Comments

1. Workspace may be explicitly provided, if desired, by use of F2CTR/DF2CTR. The reference is:

```
CALL F2CTR (NVAR, COV, LDCOV, NF, IMTH, NDF, INIT, MAXIT,
MAXSTP, EPS, EPSE, IPRINT, UNIQ, A, LDA, EVAL, STAT, DER, IS,
COVI, WK, OLD, EVEC, HESS)
```

   The additional arguments are as follows:

   IS — Integer work vector of length equal to NVAR.

   COVI — Real work vector of length equal to $NVAR^2$.

   WK — Real work vector of length equal to NVAR.

   OLD — Real work vector of length equal to NVAR.

   EVEC — Real work vector of length equal to $NVAR^2$.

   HESS — Real work vector of length equal to $NVAR^2$.

2. Informational errors

Type	Code	
3	1	Too many iterations. Convergence is assumed.
3	2	Too many step halvings. Convergence is assumed.
3	4	There are no degrees of freedom for the significance testing.

Description

Routine FACTR computes unrotated factor loadings in exploratory factor analysis models. Models available in FACTR are the principal component model for factor analysis and the common factor model with additions to the common factor model in alpha factor analysis and image analysis. Methods of estimation include principal components, principal factor, image analysis, unweighted least squares, generalized least squares, and maximum likelihood.

In the factor analysis model used for factor extraction, the basic model is given as $\sum = \Lambda\Lambda^T + \Psi$ where Σ is the $p \times p$ population covariance matrix, Λ is the $p \times k$ matrix of factor loadings relating the factors f to the observed variables x, and Ψ is the $p \times p$ matrix of covariances of the unique errors e. Here, p = NVAR and k = NF. The relationship between the factors, the unique errors, and the observed variables is given as $x = \Lambda f + e$ where, in addition, it is assumed that the expected values of e, f, and x are zero. (The sample means can be subtracted from x if the expected value of x is not zero.) It is also assumed that each factor has unit variance, the factors are independent of each other, and that the factors and the unique errors are mutually independent. In the common factor model, the elements of the vector of unique errors e are also assumed to be independent of one another so that the matrix Ψ is diagonal. This is not the case in the principal component model in which the errors may be correlated.

Further differences between the various methods concern the criterion that is optimized and the amount of computer effort required to obtain estimates. Generally speaking, the least-squares and maximum likelihood methods, which use iterative algorithms, require the most computer time with the principal factor, principal component and the image methods requiring much less time since the algorithms in these methods are not iterative. The algorithm in alpha factor analysis is also iterative, but the estimates in this method generally require somewhat less computer effort than the least-squares and maximum likelihood estimates. In all algorithms, one eigensystem analysis is required on each iteration.

The Principal Component and Principal Factor Methods

Both the principal component and the principal factor methods compute the factor loading estimates as

$$\hat{\Gamma}\hat{\Delta}^{-1/2}$$

where Γ and the diagonal matrix Δ are the eigenvectors and eigenvalues of a matrix. In the principal component model, the eigensystem analysis is performed on the sample covariance (correlation) matrix S while in the principal factor model the matrix $(S - \Psi)$ is used. If the unique error variances Ψ are not known (i.e., if `INIT` = 0) in the principal factor model, then `FACTR` obtains estimates for them as discussed in Comment 3. If the principal components model is to be used, then the `INIT` = 1 option should be set, and the vector `UNIQ` should be set so that all elements are zero. If `UNIQ` is not set, principal factor model estimates are computed.

The basic idea in the principal component method is to find factors that maximize the variance in the original data that is explained by the factors. Because this method allows the unique errors to be correlated, some factor analysts insist that the principal component method is not a factor analytic method. Usually however, the estimates obtained via the principal component model and other models in factor analysis will be quite similar.

It should be noted that both the principal component and the principal factor methods give different results when the correlation matrix is used in place of the covariance matrix. Indeed, any rescaling of the sample covariance matrix can lead to different estimates with either of these methods. A further difficulty with the principal factor method is the problem of estimating the unique error variances. Theoretically, these must be known in advance and passed to `FACTR` through `UNIQ`. In practice, the estimates of these parameters produced via the `INIT` = 0 option in `FACTR` are often used. In either case, the resulting adjusted covariance (correlation) matrix

$$(S - \hat{\Psi})$$

may not yield the `NF` positive eigenvalues required for `NF` factors to be obtained. If this occurs, the user must either lower the number of factors to be estimated or give new unique error variance values.

The Least-Squares and Maximum Likelihood Methods

Unlike the previous two methods, the algorithm used to compute estimates in this section is iterative (see Joreskog 1977). As with the principal factor model, the user may either initialize `UNIQ` or allow `FACTR` to compute initial estimates for the unique error variances. Unlike the principal factor method, `FACTR` then optimizes the criterion function with respect to both Ψ and

Γ. (In the principal factor method, Ψ is assumed to be known. Given Ψ, estimates for Λ may be obtained.)

The major differences between the methods discussed in this section are in the criterion function that is optimized. Let S denote the sample covariance (correlation) matrix, and let Σ denote the covariance matrix that is to be estimated by the factor model. In the unweighted least-squares method, also called the iterated principal factor method or the minres method (see Harman 1976, page 177), the function minimized is the sum of the squared differences between S and Σ. This is written as $\Phi_{ul} = .5 \text{ trace}((S - \Sigma)^2)$.

Generalized least-squares and maximum likelihood estimates are asymptotically equivalent methods. Maximum likelihood estimates maximize the (normal theory) likelihood $\{\Phi_{ml} = \text{trace}(\Sigma^{-1}S) - \log(|\Sigma^{-1}S|)\}$ while generalized least squares optimizes the function $\Phi_{gs} = \text{trace}(\Sigma S^{-1} - I)^2)$.

In all three methods, a two-stage optimization procedure is used. This proceeds by first solving the likelihood equations for Λ in terms of Ψ and substituting the solution into the likelihood. This gives a criterion $\phi(\Psi, \Lambda(\Psi))$, which is optimized with respect to Ψ. In the second stage, the estimates

$$\hat{\Lambda}$$

are obtained from the estimates for Ψ.

The generalized least-squares and the maximum likelihood methods allow for the computation of a statistic (STAT(3)) for testing that NF common factors are adequate to fit the model. This is a chi-squared test that all remaining parameters associated with additional factors are zero. If the probability of a larger chi-squared is small (see STAT(5)) so that the null hypothesis is rejected, then additional factors are needed (although these factors may not be of any practical importance). Failure to reject does not legitimize the model. The statistic STAT(3) is a likelihood ratio statistic in maximum likelihood estimation. As such, it asymptotically follows a chi-squared distribution with degrees of freedom given in STAT(4).

The Tucker and Lewis (1973) reliability coefficient, ρ, is returned in STAT(2) when the maximum likelihood or generalized least-squares methods are used. This coefficient is an estimate of the ratio of explained to the total variation in the data. It is computed as follows:

$$\rho = \frac{mM_o - mM_k}{mM_o - 1}$$

$$m = d - \frac{2p+5}{6} - \frac{2k}{6}$$

$$M_o = \frac{-\ln(|S|)}{p(p-1)/2}$$

$$M_k = \frac{\phi}{((p-k)^2 - p - k)/2}$$

where $|S|$ is determinant of COV, p = NVAR, k = NVAR, ϕ is the optimized criterion, and d = NDF.

Image Analysis

The term "image analysis" is used here to denote the noniterative image method of Kaiser (1963). It is not the image factor analysis discussed by Harman (1976, page 226). The image method (as well as the alpha factor analysis method) begins with the notion that only a finite number from an infinite number of possible variables have been measured. The image factor pattern is calculated under the assumption that the ratio of the number of factors to the number of observed variables is near zero so that a very good estimate for the unique error variances (for standardized variables) is given as one minus the squared multiple correlation of the variable under consideration with all variables in the covariance matrix.

First, the matrix $D^2 = (\text{diag}(S^{-1}))^{-1}$ is computed where the operator "diag" results in a matrix consisting of the diagonal elements of its argument, and S is the sample covariance (correlation) matrix. Then, the eigenvalues Λ and eigenvectors Γ of the matrix $D^{-1}S\,D^{-1}$ are computed. Finally, the unrotated image factor pattern matrix is computed as $\texttt{A} = D\Gamma[(\Lambda - I)^2\Lambda^{-1}]^{1/2}$.

Alpha Factor Analysis

The alpha factor analysis method of Kaiser and Caffrey (1965) finds factor-loading estimates to maximize the correlation between the factors and the complete universe of variables of interest. The basic idea in this method is as follows: only a finite number of variables out of a much larger set of possible variables is observed. The population factors are linearly related to this larger set while the observed factors are linearly related to the observed variables. Let f denote the factors obtainable from a finite set of observed random variables, and let ξ denote the factors obtainable from the universe of observable variables. Then, the alpha method attempts to find factor-loading estimates so as to maximize the correlation between f and ξ. In order to obtain these estimates, the iterative algorithm of Kaiser and Caffrey (1965) is used.

Comments

1. FACTR makes no attempt to solve for NF, the number of factors. In general, if NF is not known in advance, several different values of NF should be used, and the most reasonable value kept in the final solution.

2. The iterative methods are generally thought to be superior from a theoretical point of view but, in practice, often lead to solutions which differ little from the noniterative methods. For this reason, it is usually suggested that a non-iterative method be used in the initial stages of the factor analysis, and that the iterative methods be used when issues such as the number of factors have been resolved.

3. Initial estimates for the unique variances are input when INIT = 1. If the iterative methods fail for these values, new initial estimates should be tried. These may be obtained by use of another factoring method (use the final estimates from the new method as initial estimates in the old method).

Another alternative is to let FACTR compute initial estimates of the unique error variances. When INIT = 0, the initial estimates are taken as a constant

$$\left(1-\frac{k}{2p}\right)$$

divided by the diagonal elements of the

$$\hat{\Sigma}^{-1}$$

matrix. When the correlation matrix is factor analyzed, this is a constant times one minus the squared multiple correlation coefficient.

FROTA

Computes an orthogonal rotation of a factor loading matrix using a generalized orthomax criterion, including quartimax, varimax, and equamax rotations.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

W — Nonnegative constant used to define the rotation. (Input)
W = 0.0 results in quartimax rotations, W = 1.0 results in varimax rotations, and W = NF/2.0 results in equamax rotations. Other nonnegative values of W may also be used, but the best values for W are in the range (0.0, 5 * NF).

B — NVAR by NF matrix of rotated factor loadings. (Output)
If A is not needed, A and B may share the same storage locations.

T — NF by NF matrix containing the rotation transformation matrix. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
Default: NF = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A, 1).

NRM — Row normalization option. (Input)
If NRM = 1, then row (i.e., Kaiser) normalization is performed. Otherwise, row normalization is not performed.
Default: NRM = 1.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 30 is typical. MAXIT ≤ 30 defaults to 30 iterations.
Default: MAXIT = 30.

EPS — Convergence constant. (Input)
When the relative change in the criterion function is less than EPS from one iteration to the next, convergence is assumed. EPS = 0.0001 is typical. EPS ≤ 0.0 defaults to 0.0001.
Default: EPS = 0.0001.

LDB — Leading dimension of B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

FORTRAN 90 Interface

Generic: `CALL FROTA (A, W, B, T [,…])`

Specific: The specific interface names are S_FROTA and D_FROTA.

FORTRAN 77 Interface

Single:
```
CALL FROTA (NVAR, NF, A, LDA, NRM, MAXIT, W, EPS, B, LDB,
T, LDT)
```

Double: The double precision name is DFROTA.

Example

The example is taken from Emmett (1949) and involves factors derived from nine variables. In this example, the varimax method is chosen with row normalization by using W = 1.0 and NRM = 1, respectively. The results correspond to those given by Lawley and Maxwell (1971, page 84).

```
      USE FROTA_INT
      USE WRRRN_INT
      INTEGER    LDA, LDB, LDT, NF, NVAR
      REAL       W
      PARAMETER  (LDA=9, LDB=9, LDT=3, NF=3, NVAR=9, W=1.0)
!
      REAL       A(LDA,NF), B(LDB,NF), T(LDT,NF)
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960, &
      -.2955, -.4274, .0735, -.1933, -.2224, -.0354, -.1528, &
      .1045, -.0778, .4914, -.0117/
```

```
!
      CALL FROTA (A, W, B, T)
!
      CALL WRRRN ('B', B)
      CALL WRRRN ('T', T)
      END
```

Output

```
                B
          1         2         3
1    0.2638   -0.5734    0.3888
2    0.3423   -0.6610    0.1370
3    0.1625   -0.5943    0.0622
4    0.8124   -0.3197    0.1594
5    0.7356   -0.2800    0.0036
6    0.8510   -0.1890    0.2513
7    0.2164   -0.6906    0.2768
8    0.1144   -0.2431    0.6828
9    0.2687   -0.7431    0.3804

                T
          1         2         3
1    0.7307   -0.5939    0.3367
2    0.6816    0.6623   -0.3112
3   -0.0382    0.4569    0.8887
```

Comments

Workspace may be explicitly provided, if desired, by use of F2OTA/DF2OTA. The reference is

```
CALL F2OTA (NVAR, NF, A, LDA, NRM, MAXIT, W, EPS, B, LDB, T, LDT,
WORK)
```

The additional argument is:

WORK — Real work vector of length equal to NVAR.

Description

Routine FROTA performs an orthogonal rotation according to an orthomax criterion. In this analytic method of rotation, the criterion function

$$Q = \sum_i \sum_r \lambda_{ir}^4 - \frac{\gamma}{p} \sum_r \left[\sum_i \lambda_{ir}^2 \right]^2$$

is minimized by finding an orthogonal rotation matrix T such that $(\lambda_{ij}) = \Lambda = AT$ where A is the matrix of unrotated factor loadings. Here, $\gamma \geq 0$ is a user-specified constant (W) yielding a family of rotations, and p is the number of variables.

Kaiser (row) normalization can be performed on the factor loadings prior to rotation via the option parameter NRM. In Kaiser normalization, the rows of A are first "normalized" by dividing each row by the square root of the sum of its squared elements (Harman 1976). After the

rotation is complete, each row of B is "denormalized" by multiplication by its initial normalizing constant.

The method for optimizing Q proceeds by accumulating simple rotations where a simple rotation is defined to be one in which Q is optimized for two columns in Λ and for which the requirement that T be orthogonal is satisfied. A single iteration is defined to be such that each of the NF(NF − 1)/2 possible simple rotations is performed where NF is the number of factors. When the relative change in Q from one iteration to the next is less than EPS (the user-specified convergence criterion), the algorithm stops. EPS = 0.0001 is usually sufficient. Alternatively, the algorithm stops when the user-specified maximum number of iterations, MAXIT, is reached. MAXIT = 30 is usually sufficient.

The parameter in the rotation, γ, is used to provide a family of rotations. When $\gamma = 0.0$, a direct quartimax rotation results. Other values of γ yield other rotations.

FOPCS

Computes an orthogonal Procrustes rotation of a factor-loading matrix using a target matrix.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

X — NVAR by NF target matrix of the rotation. (Input)

B — NVAR by NF matrix of rotated factor loadings. (Output)

T — NF by NF factor rotation matrix. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
Default: NF = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A, 1).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

LDB — Leading dimension of B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

FORTRAN 90 Interface

Generic: `CALL FOPCS (A, X, B, T [,…])`

Specific: The specific interface names are S_FOPCS and D_FOPCS.

FORTRAN 77 Interface

Single: `CALL FOPCS (NVAR, NF, A, LDA, X, LDX, B, LDB, T, LDT )`

Double: The double precision name is DFOPCS.

Example

The following example is taken from Harman (1976, page 355). It involves the orthogonal Procrustes rotation of an 8 × 2 unrotated factor loading matrix. The original variables are measures of physical features ("lankiness" and "stockiness"). The target matrix *X* is also printed. Note that because different methods are used, Harman (1976) gets slightly different results.

```
      USE FOPCS_INT
      USE WRRRN_INT
      INTEGER    LDA, LDB, LDT, LDX, NF, NVAR
      PARAMETER  (LDA=8, LDB=8, LDT=2, LDX=8, NF=2, NVAR=8)
!
      REAL       A(LDA,NF), B(LDB,NF), T(LDT,NF), X(LDX,NF)
!
      DATA A/0.856, 0.848, 0.808, 0.831, 0.750, 0.631, 0.569, 0.607, &
      -0.324, -0.412, -0.409, -0.342, 0.571, 0.492, 0.510, 0.351/
      DATA X/0.9, 0.9, 0.9, 0.9, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, &
      0.0, 0.9, 0.9, 0.9, 0.9/
!
      CALL FOPCS (A, X, B, T)
!
      CALL WRRRN ('A', A)
      CALL WRRRN ('X', X)
      CALL WRRRN ('B', B)
      CALL WRRRN ('T', T)
      END
```

Output

```
          A
          1        2
1    0.8560  -0.3240
2    0.8480  -0.4120
3    0.8080  -0.4090
4    0.8310  -0.3420
5    0.7500   0.5710
```

```
6   0.6310   0.4920
7   0.5690   0.5100
8   0.6070   0.3510

          X
          1        2
1   0.9000   0.0000
2   0.9000   0.0000
3   0.9000   0.0000
4   0.9000   0.0000
5   0.0000   0.9000
6   0.0000   0.9000
7   0.0000   0.9000
8   0.0000   0.9000

          B
1         2
1   0.8763   0.2643
2   0.9235   0.1896
3   0.8900   0.1677
4   0.8674   0.2348
5   0.2471   0.9096
6   0.2009   0.7745
7   0.1407   0.7510
8   0.2677   0.6481

          T
          1        2
1   0.7932   0.6090
2  -0.6090   0.7932
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `F2PCS/DF2PCS`. The reference is:

   ```
   CALL F2PCS (NVAR, NF, A, LDA, X, LDX, B, LDB, T, LDT, WK, S)
   ```

   The additional arguments are as follows:

   WK — Work vector of length NF * (2 * NF + 3) – 1.

   S — Work vector of length NF * (NF + 1).

2. Informational errors

Type	Code	
4	1	`NF` = 1. No rotation is possible.
4	2	The rank of $\mathtt{A}^T * \mathtt{X}$ is less than `NF`.

3. The target matrix is a hypothesized rotated factor loading matrix with loadings chosen (based on prior knowledge) to enhance interpretability. A simple structure solution will have most of the elements in `X` near zero or one (for correlation matrix loadings).

4. This routine may also be used to refine a solution obtained by analytic rotation in routine FROTA (page 918). Choose the target matrix so that it closely resembles the analytic solution but modified to have a simple structure.

Description

Routine FOPCS performs orthogonal Procrustes rotation according to a method proposed by Schöneman (1966). Let k = NF denote the number of factors, p = NVAR denote the number of variables, A denote the $p \times k$ matrix of unrotated factor loadings, T denote the $k \times k$ orthogonal rotation matrix (orthogonality requires that $T^T T$ be a $k \times k$ identity matrix), and let X denote the target matrix. The basic idea in orthogonal Procrustes rotation is to find an orthogonal rotation matrix T such that $B = AT$ and T provides a least-squares fit between the target matrix X and the rotated loading matrix B. Schöneman's algorithm proceeds by finding the singular value decomposition of the matrix $A^T X = U\Sigma V^T$. The rotation matrix is computed as $T = UV^T$.

FDOBL

Computes a direct oblimin rotation of a factor loading matrix.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

W — Nonpositive constant used to define the rotation. (Input)

B — NVAR by NF matrix of rotated factor loadings. (Output)
If A is not needed, A and B may share the same storage locations.

T — NF by NF matrix containing the rotation transformation matrix. (Output)

FCOR — NF by NF matrix of factor correlations. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
Default: Nf = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A, 1).

NRM — Row normalization option. (Input)
If NRM = 1, then row (i.e., Kaiser) normalization is performed. Otherwise, row

normalization is not performed.
Default: NRM = 1.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 30 is typical. MAXIT = 0 defaults to 30 iterations.
Default: MAXIT = 30.

EPS — Convergence constant. (Input)
When the relative change in the criterion function is less than EPS from one iteration to the next, convergence is assumed. EPS = 0.0001 is typical. EPS = 0 defaults to 0.0001.
Default: EPS = 0.0.

LDB — Leading dimension of B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

LDFCOR — Leading dimension of FCOR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDFCOR = size (FCOR,1).

FORTRAN 90 Interface

Generic: `CALL FDOBL (A, W, B, T, FCOR [,…])`

Specific: The specific interface names are S_FDOBL and D_FDOBL.

FORTRAN 77 Interface

Single:
```
CALL FDOBL (NVAR, NF, A, LDA, NRM, W, MAXIT, EPS, B, LDB,
T, LDT, FCOR, LDFCOR)
```

Double: The double precision name is DFDOBL.

Example

The example is a continuation of the example given in routine FACTR. It involves factors derived from nine variables and uses $\gamma = -1$.

```
      USE FDOBL_INT
      USE WRRRN_INT

      INTEGER    LDA, LDB, LDFCOR, LDT, NF, NVAR
      REAL EPS, W
       PARAMETER  (EPS=0.00001, LDA=9, LDB=9, LDFCOR=3, LDT=3, &
       NF=3, NVAR=9, W=-1.0)
!
```

```
      REAL A(LDA,NF), B(LDB,NF), FCOR(LDFCOR,NF), T(LDT,NF)
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960, &
      -.2955, -.4274, .0735, -.1933, -.2224, -.0354, -.1528, &
      .1045, -.0778, .4914, -.0117/
!
      CALL FDOBL (A, W, B, T, FCOR, EPS=EPS)
!
      CALL WRRRN ('B', B)
      CALL WRRRN ('T', T)
      CALL WRRRN ('FCOR', FCOR)
      END
```

Output

```
             B
          1         2         3
1    0.1127   -0.5145    0.2917
2    0.1847   -0.6602   -0.0019
3    0.0128   -0.6354   -0.0585
4    0.7797   -0.1751    0.0598
5    0.7147   -0.1813   -0.0959
6    0.8520    0.0038    0.1820
7    0.0355   -0.6845    0.1509
8    0.0276   -0.0941    0.6824
9    0.0729   -0.7100    0.2493

         T
         1       2       3
1    0.611  -0.462   0.203
2    0.923   0.813  -0.249
3    0.042   0.728   1.050

           FCOR
         1       2       3
1    1.000  -0.427   0.217
2   -0.427   1.000  -0.411
3    0.217  -0.411   1.000
```

Comments

1. Workspace may be explicitly provided, if desired, by use of F2OBL/DF2OBL. The reference is:

   ```
   CALL F2OBL (NVAR, NF, A, LDA, NRM, W, MAXIT, EPS, B, LDB, T,
   LDT, FCOR, LDFCOR, WK1, WK2, WK3)
   ```

   The additional arguments are as follows:

   WK1 — Real work vector of length equal to NVAR.

   WK2 — Real work vector of length equal to NF.

   WK3 — Real work vector of length equal to NVAR.

2 Informational errors

Type	Code	
3	1	The algorithm did not converge within MAXIT iterations.
4	1	NF = 1. No rotation is possible.

3. The parameter W determines the type of direct OBLIMIN rotation to be performed. In general, W must be negative. W = 0.0 yields direct quartimin rotation. As W approaches negative infinity, the orthogonality among the factors will increase.

Description

Routine FDOBL performs direct oblimin rotation. In this analytic method of rotation, the criterion function

$$Q = \sum_{r \neq s} \left[\sum_i \lambda_{ir}^2 \lambda_{is}^2 - \frac{\gamma}{p} \sum_i \lambda_{ir}^2 \sum_i \lambda_{is}^2 \right]$$

is minimized by finding a rotation matrix T such that $(\lambda_{ir}) = \Lambda = AT$ and $(T^T T)^{-1}$ is a correlation matrix. Here, $\gamma \leq 0$ is a user-specified constant (W) yielding a family of rotations, and p is the number of variables. The rotation is said to be direct because it minimizes Q with respect to the factor loadings directly, ignoring the reference structure.

Kaiser normalization can be performed on the factor loadings prior to rotation via the option parameter NRM. In Kaiser normalization (see Harman 1976), the rows of A are first "normalized" by dividing each row by the square root of the sum of its squared elements. After the rotation is complete, each row of B is "denormalized" by multiplication by its initial normalizing constant.

The method for optimizing Q is essentially the method first proposed by Jennrich and Sampson (1966). It proceeds by accumulating simple rotations where a simple rotation is defined to be one in which Q is optimized for a given factor in the plane of a second factor, and for which the requirement that $(T^T T)^{-1}$ be a correlation matrix is satisfied. An iteration is defined to be such that each of the NF(NF − 1) possible simple rotations is performed, where NF is the number of factors. When the relative change in Q from one iteration to the next is less than EPS (the user-specified convergence criterion), the algorithm stops. EPS = .0001 is usually sufficient. Alternatively, the algorithm stops when the user-specified maximum number of iterations, MAXIT, is reached. MAXIT = 30 is usually sufficient.

The parameter in the rotation, γ, is used to provide a family of rotations. Harman (1976) recommends that γ be strictly less than or equal to zero. When $\gamma = 0.0$, a direct quartimin rotation results. Other values of γ yield other rotations. Harman (1976) suggests that the direct quartimin rotations yield the most highly correlated factors while more orthogonal factors result as γ approaches $-\infty$.

FPRMX

Computes an oblique Promax or Procrustes rotation of a factor loading matrix using a target matrix, including pivot and power vector options.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

W — Constant used to define the orthomax orthogonal rotation. (Input)
Values for W are discussed in the Comments. W must be nonnegative. Not used if IMTH = 3.

F — Vector of length NF containing the power vector or the pivot constants depending upon whether IMTH = 1 or IMTH = 2, respectively. (Input)
Not used if IMTH = 3.

X — NVAR by NF target matrix for the rotation. (Output, if IMTH = 1 or 2; input, if IMTH = 3)
For IMTH = 1 or 2, X is the target matrix derived from the orthomax rotation. For IMTH = 3, X is input.

B — NVAR by NF matrix of rotated factor loadings. (Output)

T — NF by NF factor rotation matrix. (Output)

FCOR — NF by NF matrix of factor correlations. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
NVAR must be greater than or equal to 2.
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
NF must be greater than or equal to 2.
Default: Nf = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A, 1).

IMTH — Method used for rotation. (Input)
Default: IMTH = 1.

IMTH	Method
1	The Promax method.
2	The pivotal Promax method.
3	Oblique Procrustes method.

NRM — Normalization option parameter. (Input)
NRM = 0 indicates that no row (Kaiser) normalization is to be performed in the orthomax orthogonal rotation. Otherwise, row normalization is performed. Not used when IMTH = 3.
Default: NRM = 1.

MAXIT — Maximum number of iterations. (Input)
Thirty is typical. Not used if IMTH = 3.
Default: MAXIT = 30.

EPS — Convergence constant for the orthogonal rotation. (Input)
When the relative change in the orthomax criterion function is less than EPS from one iteration to the next, convergence is assumed. EPS = 0.0001 is typical. EPS nonpositive defaults to EPS = 0.0001.
Default: EPS = 0.0001.

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

LDB — Leading dimension of B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

LDFCOR — Leading dimension of FCOR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDFCOR = size (FCOR,1).

FORTRAN 90 Interface

Generic: `CALL FPRMX (A, W, F, X, B, T, FCOR [,…])`

Specific: The specific interface names are S_FPRMX and D_FPRMX.

FORTRAN 77 Interface

Single:
```
CALL FPRMX (NVAR, NF, A, LDA, IMTH, NRM, W, MAXIT, EPS, F,
X, LDX, B, LDB, T, LDT, FCOR, LDFCOR)
```

Double: The double precision name is DFPRMX.

Example

The following example is a continuation of the example in the FACTR (page 909) procedure. It involves nine variables and three factors. The pivotal Promax method is illustrated.

```
      USE FPRMX_INT
      USE WRRRN_INT
      INTEGER    IMTH, LDA, LDB, LDFCOR, LDT, LDX, NF, NVAR
      REAL       W
      PARAMETER  (IMTH=2, LDA=9, LDB=9, LDFCOR=3, LDT=3, LDX=9, NF=3, &
                  NVAR=9, W=1.0)
!
      REAL       A(LDA,NF), B(LDB,NF), F(NF), FCOR(LDFCOR,NF), &
                  T(LDT,NF), X(LDX,NF)
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960, &
      -.2955, -.4274, .0735, -.1933, -.2224, -.0354, -.1528, &
      .1045, -.0778, .4914, -.0117/
!
      DATA F/0.5, 0.5, 0.5/
!
      CALL FPRMX (A, W, F, X, B, T, FCOR, IMTH=IMTH)
!
      CALL WRRRN ('X', X)
      CALL WRRRN ('B', B)
      CALL WRRRN ('T', T)
      CALL WRRRN ('FCOR', FCOR)
      END
```

Output

```
                X
         1        2        3
1   0.0800  -0.6157   0.2967
2   0.2089  -0.7311   0.0007
3   0.0037  -0.6454   0.0000
4   0.8800  -0.1681   0.0032
5   0.8116  -0.1030   0.0000
6   0.9096  -0.0122   0.0640
7   0.0291  -0.7649   0.0982
8   0.0001  -0.0546   0.7563
9   0.0866  -0.8189   0.2807

                B
         1        2        3
1   0.0997  -0.5089   0.3038
2   0.1900  -0.6463   0.0077
3   0.0163  -0.6270  -0.0421
4   0.7991  -0.1469   0.0285
5   0.7408  -0.1531  -0.1256
6   0.8668   0.0308   0.1436
7   0.0280  -0.6777   0.1699
8  -0.0094  -0.1017   0.6911
9   0.0611  -0.7031   0.2683
```

```
            T
         1        2        3
1    0.617   -0.439    0.189
2    0.963    0.839   -0.318
3   -0.015    0.707    1.039

           FCOR
         1        2        3
1    1.000   -0.464    0.316
2   -0.464    1.000   -0.395
3    0.316   -0.395    1.000
```

Comments

1. Workspace may be explicitly provided, if desired, by use of F2RMX/DF2RMX. The reference is:

   ```
   CALL F2RMX (NVAR, NF, A, LDA, IMTH, NRM, W, MAXIT, EPS, F, X,
   LDX, B, LDB, T, LDT, FCOR, LDFCOR, QR, QRAUX, IPVT, WORK)
   ```

   The additional arguments are as follows:

   QR — Work vector of length NVAR * NF.

   QRAUX — Work vector of length NF.

   IPVT — Work vector of length NF.

   WORK — Work vector of length 2 * NF.

2. Arguments W, EPS, and NRM are input arguments to routine FROTA (page 918) when IMTH = 1 or 2. (They are not used when IMTH = 3.) See FROTA for common values of W. Generally, W can be any positive real number, but the best values lie in the range (1.0, 5.0 * NF). Generally, the variances accounted for by the factors approach the same value as W increases.

3. For IMTH = 1, all F(*j*) should be greater than 1.0, typically 4.0. Generally, the larger the values of F(*j*), the more oblique the solution will be. For IMTH = 2, F(*j*) should be in the interval (0.0, 1.0).

4. When IMTH = 3, the target matrix, X, is a hypothesized rotated factor loading matrix based upon prior knowledge with loadings chosen to enhance interpretability. A simple structure solution will have most of the weights X(*i, j*) either zero or large in magnitude. Note that the two options IMTH = 1 or 2 attempt to achieve this simple structure based upon an initial orthogonal rotation.

Description

Routine FPRMX performs oblique rotations via the Promax, the pivotal Promax, or the oblique Procrustes methods. In all of these methods, a target matrix *X* is first either computed or

specified by the user. The differences in the methods relate to how the target matrix is first obtained.

Given a $p \times k$ target matrix, X, and a $p \times k$ orthogonal matrix of unrotated factor loadings, A, compute the rotation matrix T as follows: First regress each column of A on X yielding a $k \times k$ matrix β. Then, let $\gamma = \text{diag}(\beta^T \beta)$ where diag denotes the diagonal matrix obtained from the diagonal of the square matrix. Standardize β to obtain $T = \gamma^{-1/2} \beta$. The rotated loadings are computed as $B = AT$ while the factor correlations can be computed as the inverse of the $T^T T$ matrix.

In the Promax method, the unrotated factor loadings are first rotated according to an orthomax criterion via routine FROTA (page 918). The target matrix X is taken as the elements of the B raised to a power greater than one but retaining the same sign as the original loadings. In FPRMX, column i of the rotated matrix B is raised to the power F(i). A power of four is commonly used. Generally, the larger the power, the more oblique the solution.

In the pivotal Promax method, the unrotated matrix is first rotated to an orthomax orthogonal solution as in the Promax case. Then, rather than raising the i-th column in B to the power F(i), the elements x_{ij} of X are obtained from the elements b_{ij} of B by raising the ij element of B to the power F(i)/b_{ij}. This has the effects of greatly increasing in X those elements in B that are greater in magnitude than the pivot elements F(i), and of greatly decreasing those elements that are less than F(i).

In the oblique Procrustes method, the elements of X are specified by the user as input to the FPRMX routine. No orthogonal rotation is performed in the oblique Procrustes method.

FHARR

Computes an oblique rotation of an unrotated factor loading matrix using the Harris-Kaiser method.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

W — Constant used to define the rotation. (Input)
The value of W must be nonnegative. See Comments.

C — Constant between zero and one used to define the rotation. (Input)
See Comments.

B — NVAR by NF matrix containing the rotated factor loadings. (Output)

T — NF by NF factor rotation matrix. (Output)

FCOR — NF by NF matrix containing the factor correlations. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
Default: NF = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A, 1).

NRM — Row normalization option. (Input)
If NRM = 1, then row (i.e., Kaiser) normalization is performed. Otherwise, row normalization is not performed.
Default: NRM = 1.

MAXIT — Maximum number of iterations. (Input)
A typical value is 30.
Default: MAXIT = 30.

EPS — Convergence constant for the rotation angle. (Input)
EPS = 0.0001 is typical. If EPS is less that or equal to 0.0, then EPS = 0.0001 is used.
Default: EPS = 0.0.

SCALE — Vector of length NVAR containing a scaling vector. (Input)
All elements in SCALE should be set to one if principal components or unweighted least squares was used to obtain the unrotated factor loadings. The elements of SCALE should be set to the unique error variances (vector UNIQ in subroutine FACTR) if the principal factor, generalized least squares, maximum likelihood, or the image method was used. Finally, in alpha factor analysis, the elements of SCALE should be set to the communalities (one minus the uniquenesses in standardized data).
Default: SCALE = 1.0.

LDB — Leading dimension of B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

LDFCOR — Leading dimension of FCOR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDFCOR = size (FCOR,1).

FORTRAN 90 Interface

Generic: `CALL FHARR (A, W, C, B, T, FCOR [,…])`

Specific: The specific interface names are S_FHARR and D_FHARR.

FORTRAN 77 Interface

Single:
```
CALL FHARR (NVAR, NF, A, LDA, NRM, MAXIT, W, C, EPS,
SCALE, B, LDB, T, LDT, FCOR, LDFCOR)
```

Double: The double precision name is DFHARR.

Example

The example is a continuation of the example in routine FACTR (page 909). It involves 9 variables. A rotation with row normalization and 3 factors is performed.

```
      USE FHARR_INT
      USE WRRRN_INT
      INTEGER    LDA, LDB, LDFCOR, LDT, NF, NVAR
      REAL       C, W
      PARAMETER  (C=0.5, LDA=9, LDB=9, LDFCOR=3, LDT=3, NF=3, &
                  NVAR=9, W=1.0)
!
      REAL A(LDA,NF), B(LDB,NF), FCOR(LDFCOR,NF), SCALE(NVAR), &
      T(LDT,NF)
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960,  &
      -.2955, -.4274, .0735, -.1933, -.2224, -.0354, -.1528, &
      .1045, -.0778, .4914, -.0117/
!
      DATA SCALE/.4505, .4271, .6165, .2123, .3805, .1769, .3995, &
      .4616, .2309/
!
      CALL FHARR (A, W, C, B, T, FCOR, SCALE=SCALE)
!
      CALL WRRRN ('B', B)
      CALL WRRRN ('T', T)
      CALL WRRRN ('FCOR', FCOR)
      END
```

Output

```
             B
         1        2        3
1   0.1542  -0.5103   0.2749
2   0.2470  -0.6477  -0.0233
3   0.0744  -0.6185  -0.0750
4   0.7934  -0.1897   0.0363
5   0.7329  -0.1909  -0.1175
6   0.8456  -0.0194   0.1610
7   0.0966  -0.6713   0.1320
8   0.0198  -0.1067   0.6773
9   0.1340  -0.6991   0.2285

            T
```

```
        1       2       3
1   0.649  -0.469   0.175
2   0.850   0.777  -0.249
3  -0.053   0.687   1.065

          FCOR
        1       2       3
1   1.000  -0.335   0.250
2  -0.335   1.000  -0.413
3   0.250  -0.413   1.000
```

Comments

1. Workspace may be explicitly provided, if desired, by use of F2ARR/DF2ARR. The reference is:

   ```
   CALL F2ARR (NVAR, NF, A, LDA, NRM, MAXIT, W, C, EPS, SCALE, B,
   LDB, T, LDT, FCOR, LDFCOR, RWK1, RWK2)
   ```

   The additional arguments are as follows:

   RWK1 — Real work vector of length equal to 2 * NF.

   RWK2 — Real work vector of length equal to NVAR.

2. Argument C must be between 0.0 and 1.0. The larger C is, the more orthogonal the rotated factors are. Rarely, should C be greater than 0.5.

3. Arguments W, EPS, and NRM are arguments to routine FROTA (page 918). See FROTA for common values of W in orthogonal rotations. For FHARR, the best values of W are in the range (0.0, 5.0 * NF). Generally, the variances of the factors converge to the same value as W increases.

Description

Routine FHARR performs an oblique analytic rotation of unrotated factor loadings via a method proposed by Harris and Kaiser (1964). In this method of rotation, the eigenvectors obtained from the factor extraction are weighted by a factor $\Delta^{c/2}$ where Δ is the diagonal matrix of eigenvalues obtained in the factor extraction and c is a specified constant. These transformed eigenvectors are then rotated according to an orthomax criterion.

The transformation used to obtain the weighted eigenvectors, Γ^*, from the unrotated loadings, A, is given as $\Gamma^* = \Psi^{-1/2} A\Delta^{(c-1)/2}$ where Ψ is the matrix of unique error variances output by routine FACTR (page 909). The matrix should be set to an identity matrix if the principal component, unweighted least squares, or alpha factor analysis method is used in routine FACTR to obtain the unrotated factor loadings (IMTH = 0,1, or 5). This is required because in these methods of factor analysis, the eigenvectors are not premultiplied by a diagonal matrix when obtaining the unrotated factor loadings.

After Γ^* has been computed, it is rotated according to a user-selected orthomax criterion. The member of the orthomax family to be used is selected via a constant W. (See the description of routine FROTA, page 918.) Because Γ^* is used in place of A (the unrotated factor loadings in routine FROTA), the matrix resulting from the rotation is (after standardizing by preand postmultiplication by the diagonal matrices U^{-1} and Δ^{1-c}) a matrix of obliquely rotated loadings.

Note that the effect of W is less pronounced than the effect of C. Using $c = 1.0$ yields an orthogonal orthomax rotation while $c = 0.0$ yields the most oblique factors. A common choice for c is given by $c = 0.5$. One good choice for W is 1.0. W = 1.0 yields a varimax rotation on the weighted eigenvectors.

FGCRF

Computes direct oblique rotation according to a generalized fourth-degree polynomial criterion.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

W — Vector of length 4 containing the constants ω_1, ω_2, ω_3, ω_4 necessary to define the rotation. (Input)

Some common rotations are

Rotation	W(1)	W(2)	W(3)	W(4)
Quartimin	0	1	0	–1
Covarimin	–1/NVAR	1	1/NVAR	–1
Oblimin	–γ/NVAR	1	γ/NVAR	–1
Crawford-Ferguson	0	K_1	K_2	$-K_1 - K_2$

where K_1, K_2, and γ are constants (determined by the user).

B — NVAR by NF matrix of rotated factor loadings. (Output)
If A is not needed, A and B can share the same storage locations.

T — NF by NF matrix containing the rotation transformation matrix. (Output)

FCOR — NF by NF matrix of factor correlations. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
Default: NF = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A, 1).

NRM — Row normalization option. (Input)
If NRM = 1, then row (i.e., Kaiser) normalization is performed. If NRM = 0, row normalization is not performed.
Default: NRM = 1.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 30 is typical. MAXIT ≤ 30 defaults to 30 iterations.
Default: MAXIT = 30.

EPS — Convergence constant. (Input)
When the relative change in the criterion function is less than EPS from one iteration to the next, convergence is assumed. EPS = 0.0001 is typical. EPS ≤ 0.0 defaults to 0.0001.
Default: EPS = 0.0.

LDB — Leading dimension of B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

LDFCOR — Leading dimension of FCOR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDFCOR = size (FCOR,1).

FORTRAN 90 Interface

Generic: `CALL FGCRF (A, W, B, T, FCOR [,…])`

Specific: The specific interface names are S_FGCRF and D_FGCRF.

FORTRAN 77 Interface

Single:
```
CALL FGCRF (NVAR, NF, A, LDA, NRM, W, MAXIT, EPS, B, LDB,
T, LDT, FCOR, LDFCOR)
```

Double: The double precision name is DFGCRF.

Example

The example is a continuation of the example in routine FACTR (page 909). It involves nine variables. A Crawford-Ferguson rotation with row normalization and 3 factors is performed.

```
      USE FGCRF_INT
      USE WRRRN_INT
      INTEGER    LDA, LDB, LDFCOR, LDT, NF, NVAR
      PARAMETER  (LDA=9, LDB=9, LDFCOR=3, LDT=3, NF=3, NVAR=9)
!
      REAL       A(LDA,NF), B(LDB,NF), FCOR(LDFCOR,NF), T(LDT,NF), W(4)
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960, &
      -.2955, -.4274, .0735, -.1933, -.2224, -.0354, -.1528, &
      .1045, -.0778, .4914, -.0117/
      DATA W/0.0, 7.0, 1.0, -8.0/
!
      CALL FGCRF (A, W, B, T, FCOR)
!
      CALL WRRRN ('B', B)
      CALL WRRRN ('T', T)
      CALL WRRRN ('FCOR', FCOR)
      END
```

Output

```
              B
         1         2         3
1   0.1156  -0.3875   0.3992
2   0.2161  -0.5831   0.0924
3   0.0422  -0.5859   0.0264
4   0.8051  -0.0906   0.0886
5   0.7495  -0.1373  -0.0839
6   0.8639   0.1045   0.1987
7   0.0527  -0.5792   0.2671
8  -0.0162   0.0779   0.7748
9   0.0852  -0.5765   0.3803

             T
         1       2       3
1   0.632  -0.327   0.290
2   0.935   0.737  -0.399
3  -0.060   0.907   1.066

           FCOR
         1       2       3
1   1.000  -0.434   0.365
2  -0.434   1.000  -0.498
3   0.365  -0.498   1.000
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `F2CRF/DF2CRF`. The reference is:

   ```
   CALL F2CRF (NVAR, NF, A, LDA, NRM, W, MAXIT, EPS, B, LDB, T,
   LDT, FCOR, LDFCOR, RWK1, RWK2, RWK3)
   ```

   The additional arguments are as follows:

RWK1 — Work vector of length NVAR.

RWK2 — Work vector of length NVAR * (NF + 1).

RWK3 — Work vector of length NF^2.

2. Informational error

Type	Code	
3	1	The algorithm did not converge within MAXIT iterations.

Description

Routine FGCRF performs direct oblique factor rotation for an arbitrary fourth-degree polynomial criterion function. Let p = NVAR denote the number of variables, and let k = NF denote the number of factors. Then, the criterion function

$$Q = \omega_1 \left(\sum_{i=1}^{p} \sum_{r=1}^{k} \lambda_{ir}^2 \right)^2 + \omega_2 \sum_{i=1}^{p} \left(\sum_{r=1}^{k} \lambda_{ir}^2 \right)^2$$
$$+ \omega_3 \sum_{r=1}^{k} \left(\sum_{i=1}^{p} \lambda_{ir}^2 \right)^2 + \omega_4 \sum_{i=1}^{p} \sum_{r=1}^{k} \lambda_{ir}^4$$

is minimized by finding a rotation matrix T such that $(\lambda_{ij}) = \Lambda = AT$ and $T^{-1} (T^{-1})^T$ is a correlation matrix. Here, ω_i= W(i), i = 1, …, 4 are user specified constants. The rotation is said to be direct because it minimizes Q with respect to the factor loadings directly, ignoring the reference structure (see, e.g., Harman, 1976).

Kaiser normalization (Harman, 1976) is specified when option parameter NRM = 1. When Kaiser normalization is performed, the rows of A are first "normalized" by dividing each row by the square root of the sum of its squared elements. The rotation is then performed. The rows of B are then "denormalized" by multiplying each row by the initial row normalizing constant.

The criterion function Q was first proposed by Jennrich (1973). It generalizes the oblimin criterion function and the criterion function proposed by Crawford and Ferguson (1970) to an arbitrary fourth degree criterion. Q is optimized by accumulating simple rotations where a simple rotation is defined to be an optimal factor rotation (with respect to Q) for two columns of Λ, and for which the requirement that $T^{-1} (T^{-1})^T$ be a correlation matrix is satisfied. FGCRF determines the optimal simple rotation by finding the roots of a cubic polynomial equation. The details are contained in Clarkson and Jennrich (1988).

Table 1: Specific Criteria in the General Symmetric Family

Criterion	ω_1	ω_2	ω_3	ω_4
Quartimin	0	1	0	–1
Covarimin	$-1/p$	1	$1/p$	–1
Oblimin	$-\gamma/p$	1	γ/p	–1
Crawford-Ferguson	0	K_1	K_2	$-K_1 - K_2$

An iteration is complete after all possible $k(k-1)$ simple rotations have been performed. When the relative change in Q from one iteration to the next is less than EPS, the algorithm stops. EPS = .0001 is usually sufficient. Alternatively, the algorithm stops when the user specified maximum number of iterations, MAXIT, is reached. MAXIT = 30 is typical.

Notes

The parameters in the rotation, ω_1, provide for a two-dimensional family of rotations. When $\omega_1 = -\gamma/p$, $\omega_2 = 1$, $\omega_3 = \gamma/p$, and $\omega_4 = -1$, then a direct oblimin rotation with parameter γ is performed. Direct oblimin rotations are also performed by routine FDOBL (page 924), which is somewhat faster. For $\omega_1 = 0$, $\omega_2 = K_1$, $\omega_3 = K_2$, and $\omega_4 = -(K_1 + K_2)$ direct Crawford-Ferguson rotation with parameters K_1 and K_2 results (see Crawford and Ferguson 1970, or Clarkson and Jennrich 1988). Other values of ω yield other rotations. Common values for ω are as in Table 1.

FIMAG

Computes the image transformation matrix.

Required Arguments

T — NF by NF transformation matrix. (Input)

TI — NF by NF image transformation matrix. (Output)

Optional Arguments

NF — Number of factors. (Input)
Default: NF = size (T,2).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

LDTI — Leading dimension of TI exactly as specified in the dimension statement in the calling program. (Input)
Default: LDTI = size (TI,1).

FORTRAN 90 Interface

Generic: `CALL FIMAG (T, TI [,…])`

Specific: The specific interface names are S_FIMAG and D_FIMAG.

FORTRAN 77 Interface

Single: `CALL FIMAG (NF, T, LDT, TI, LDTI)`

Double: The double precision name is DFIMAG.

Example

This example is a continuation of the example contained in the manual document for routine FROTA (page 918). The image transformation matrix is obtained from the orthogonal rotation matrix. Some small differences between the matrix TI when compared with the matrix T computed via routine FROTA can be seen. These differences are because of roundoff error since for orthogonal rotations, the image transformation matrix is the same as the rotation matrix.

```
      USE FIMAG_INT
      USE WRRRN_INT
      INTEGER    LDT, LDTI, NF
      PARAMETER  (LDT=3, LDTI=3, NF=3)
!
      REAL       T(LDT,NF), TI(LDTI,NF)
!
      DATA T/.7307, .6816, -.0382, -.5939, .6623, .4569, .3367, &
       -.3112, .8887/
!
      CALL FIMAG (T, TI)
!
      CALL WRRRN ('TI', TI)
      END
```

Output

```
            TI
        1       2       3
1   0.7307  -0.5938   0.3367
2   0.6816   0.6622  -0.3112
3  -0.0382   0.4569   0.8887
```

Comments

1. Workspace may be explicitly provided, if desired, by use of F2MAG/DF2MAG. The reference is:

   ```
   CALL F2MAG (NF, T, LDT, TI, LDTI, RWK, IWK)
   ```

   The additional arguments are as follows:

   RWK — Real work vector of length NF + NF(NF – 1)/2.

   IWK — Integer work vector of length NF.

2. Informational Error

Type	Code	
3	1	T is ill-conditioned. The solution may not be accurate.

Description

Routine FIMAG computes the image transformation matrix TI from the factor rotation matrix (T). The image transformation matrix takes the unrotated factor loadings into the factor structure matrix when the unrotated loadings are computed from a correlation matrix. It is computed as the inverse of the transpose of the factor rotation matrix T. When orthogonal rotations are used, $(T^T)^{-1} = T$ so there is no reason to compute the image transformation matrix.

FRVAR

Computes the factor structure and the variance explained by each factor.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

T — NF by NF factor rotation matrix. (Input)

VAR — Vector of length NVAR containing the variances of the original variables. (Input)
If standardized variables were used (i.e., the loadings are from a correlation matrix), then set VAR(1) to any negative number. In this case, VAR may be dimensioned of length one.

S — NVAR by NF factor structure matrix. (Output)

FVAR — Vector of length NF containing the variance accounted for by each of the NF rotated factors. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
Default: Nf = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A, 1).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

LDS — Leading dimension of S exactly as specified in the dimension statement in the calling program. (Input)
Default: LDS = size (S,1).

FORTRAN 90 Interface

Generic: `CALL FRVAR (A, T, VAR, S, FVAR [,…])`

Specific: The specific interface names are `S_FRVAR` and `D_FRVAR`.

FORTRAN 77 Interface

Single: `CALL FRVAR (NVAR, NF, A, LDA, T, LDT, VAR, S, LDS, FVAR)`

Double: The double precision name is `DFRVAR`.

Example

The following example illustrates the use of routine `FRVAR` when the structure and and an index of factor importance for obliquely rotated loadings (obtained from routine `FDOBL`, page 924) are desired. Note in this example that the elements of `FVAR` are not variances since the rotation is oblique.

```
      USE FRVAR_INT
      USE WRRRN_INT
      INTEGER    LDA, LDS, LDT, NF, NVAR
      PARAMETER  (LDA=9, LDS=9, LDT=3, NF=3, NVAR=9)
!
      REAL       A(LDA,NF), FVAR(NF), S(LDS,NF), T(LDT,NF), VAR(NVAR)
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960, &
      -.2955, -.4274, .0735, -.1933, -.2224, -.0354, -.1528, &
      .1045, -.0778, .4914, -.0117/
!
      DATA T/0.611, 0.923, 0.042, -0.462, 0.813, 0.728, 0.203, -0.249, &
      1.050/
!
      DATA VAR/9*1.0/
!
      CALL FRVAR (A, T, VAR, S, FVAR)
!
      CALL WRRRN ('S', S)
      CALL WRRRN ('FVAR', FVAR, 1, NF, 1)
      END
```

Output

```
             S
         1        2        3
1   0.3958  -0.6825   0.5274
2   0.4662  -0.7385   0.3093
3   0.2715  -0.6171   0.2052
4   0.8673  -0.5328   0.3010
5   0.7712  -0.4473   0.1338
6   0.8897  -0.4348   0.3654
7   0.3606  -0.7618   0.4397
8   0.2160  -0.3860   0.7270
```

```
9   0.4303  -0.8437   0.5566

        FVAR
    1       2       3
2.170   2.559   0.915
```

Comments

Workspace may be explicitly provided, if desired, by use of F2VAR/DF2VAR. The reference is

```
CALL F2VAR (NVAR, NF, A, LDA, T, LDT, VAR, S, LDS, FVAR, TINV, WK,
IWK)
```

The additional arguments are as follows:

TINV — Work vector of length NF^2.

WK — Work vector of NF * (1 + NVAR).

IWK — Work vector of length NF.

Description

Routine FRVAR computes the factor structure matrix (the matrix of correlations between the observed variables and the hypothesized factors) and the variance explained by each of the factors (for orthogonal rotations). For oblique rotations, FRVAR computes a measure of the importance of the factors, the sum of the squared elements in each column.

Let Δ denote the diagonal matrix containing the elements of the vector VAR along its diagonal. The estimated factor structure matrix S is computed as

$$S = \Delta^{-\frac{1}{2}} A(T^{-1})^T$$

while the elements of FVAR are computed as the diagonal elements of

$$S^T \Delta^{\frac{1}{2}} AT$$

If the factors were obtained from a correlation matrix (or the factor variances for standardized variables are desired), then the elements of the vector VAR should either all be 1.0, or the first element of VAR should be set to any negative number. In either case, variances of 1.0 are used.

The user should be careful to input the unrotated loadings. When obliquely rotated loadings are input, the output vector FVAR contains a measure of each factors importance, but it does not contain the variance of each factor.

FCOEF

Computes a matrix of factor score coefficients for input to the routine FSCOR.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

COV — The variance-covariance or correlation matrix of order NVAR from which the factor loadings were obtained. (Input)
COV is not used and may be dimensioned of length 1 if IMTH = 2 or 5.

T — NF by NF factor rotation matrix or transformation matrix. (Input)
If the image method is being used, then routine FIMAG (page 940) needs to be called after the rotation routine to obtain the image transformation matrix. TI is then input for T in FCOEF. If factor score coefficients for the unrotated loadings are desired, T should be set to the identity matrix prior to calling FCOEF.

SCOEF — NVAR by NF factor score coefficient matrix. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
Default: NF = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A, 1).

IMTH — Method to be used to obtain the factor scores. (Input)
Default: IMTH = 1.

IMTH	Method
1	Regression method
2	Least squares method
3	Bartlett method
4	Anderson and Rubin method
5	Image score for image analysis

See the Comments for a table of the methods that are appropriate for a given type of factor extraction and rotation.

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

LDSCOE — Leading dimension of SCOEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSCOE = size (SCOEF,1).

FORTRAN 90 Interface

Generic: `CALL FCOEF (A, COV, T, SCOEF [,…])`

Specific: The specific interface names are S_FCOEF and D_FCOEF.

FORTRAN 77 Interface

Single:
```
CALL FCOEF (NVAR, NF, A, LDA, IMTH, COV, LDCOV, T,
LDT, SCOEF, LDSCOE)
```

Double: The double precision name is DFCOEF.

Example

In the following example, the regression method is used to obtain estimated factor score coefficients for a 9-variable problem with 3 factors. An oblique rotation method was used with the maximum likelihood common factor model to obtain the factor loadings. Routine FDOBL (page 924) was used to obtain the oblique factor loadings.

```
      USE FCOEF_INT
      USE WRRRN_INT
      INTEGER    IMTH, LDA, LDCOV, LDSCOE, LDT, NF, NVAR
      PARAMETER  (IMTH=1, LDA=9, LDCOV=9, LDSCOE=9, LDT=3, NF=3, NVAR=9)
!
      REAL A(LDA,NF), COV(LDCOV,NVAR), SCOEF(LDSCOE,NF), &
      T(LDT,NF)
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960, &
      -.2955, -.4274, .0735, -.1933, -.2224, -.0354, -.1528, &
      .1045, -.0778, .4914, -.0117/
!
      DATA T/0.611, 0.923, 0.042, -0.462, 0.813, 0.728, 0.203, -0.249, &
      1.050/
!
      DATA COV/1.000, 0.523, 0.395, 0.471, 0.346, 0.426, 0.576, 0.434, &
       0.639, 0.523, 1.000, 0.479, 0.506, 0.418, 0.462, 0.547, &
       0.283, 0.645, 0.395, 0.479, 1.000, 0.355, 0.270, 0.254, &
```

```
     0.452, 0.219, 0.504, 0.471, 0.506, 0.355, 1.000, 0.691, &
     0.791, 0.443, 0.285, 0.505, 0.346, 0.418, 0.270, 0.691, &
     1.000, 0.679, 0.383, 0.149, 0.409, 0.426, 0.462, 0.254, &
     0.791, 0.679, 1.000, 0.372, 0.314, 0.472, 0.576, 0.547, &
     0.452, 0.443, 0.383, 0.372, 1.000, 0.385, 0.680, 0.434, &
     0.283, 0.219, 0.285, 0.149, 0.314, 0.385, 1.000, 0.470, &
     0.639, 0.645, 0.504, 0.505, 0.409, 0.472, 0.680, 0.470, &
     1.000/
!
     CALL FCOEF (A, COV, T, SCOEF)
!
     CALL WRRRN ('SCOEF', SCOEF)
     END
```

Output

```
          SCOEF
        1       2       3
1  -0.0102 -0.1350  0.1781
2   0.0269 -0.2191 -0.0825
3  -0.0080 -0.1536 -0.0791
4   0.3788 -0.0597 -0.0596
5   0.2067 -0.0554 -0.1768
6   0.4885  0.1103  0.2084
7  -0.0258 -0.2317  0.0612
8  -0.0474  0.0345  0.5269
9  -0.0431 -0.3967  0.2507
```

Comments

1. Workspace may be explicitly provided, if desired, by use of F2OEF/DF2OEF. The reference is:

   ```
   CALL F2OEF (NVAR, NF, A, LDA, IMTH, COV, LDCOV, T, LDT, SCOEF,
   LDSCOE, B, RWK1, S, UNIQ, RWK2)
   ```

   The additional arguments are as follows:

   B — Real work vector of length 2 * NVAR * NF if IMTH = 4, and of length NVAR * NF otherwise.

   RWK1 — Real work vector of length $NVAR^2$ if IMTH = 1 or 4, and of length NF^2 if IMTH = 2 or 3. Otherwise, RWK1 is of length 1.

   S — Real work vector of length NF^2 if IMTH = 4. Otherwise, S is dimensioned of length 1.

   UNIQ — Real work vector of length NVAR if IMTH = 2, 3, or 4. Otherwise, UNIQ is dimensioned of length 1.

   RWK2 — Real work vector of length NF if IMTH is not 5. If IMTH = 5, then RWK2 is of length 1.

2. The method used for computing the factor score coefficients depends both upon the method used to extract the factor loadings in routine FACTR (page 909) and whether the factor loadings were orthogonally or obliquely rotated. In the following table, the numbers in parentheses refer to IMTH in routine FACTR and the numbers in the cells refer to IMTH in FCOEF.

FACTR Method (IMTH)	No Rotation	Orthogonal Rotation	Oblique Rotation
Component (0)	1, 2	1, 2	1, 2
Image (4)	5	5	5
Common Factor			
ULS (1)	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
GLS (2)	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
ML (3)	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
Alpha (5)	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4

Description

Routine FCOEF computes factor score coefficients that may subsequently be used in computing the factor scores in routine FSCOR (page 949). Five options for computing the coefficients are available according to the input parameter IMTH. The method that should be used depends upon the method used in extracting the factor loadings. See the Comments section above for values to use for IMTH when various methods of factor extraction are used.

Let S denote the covariance (or correlation) matrix from which the factors were obtained, let β denote the factor score coefficients, let $U^2 = \text{diag}(S - AA^T)$ denote the unique error variances, and let $B = AT$ denote the rotated factor loadings (if coefficients for the unrotated loadings are desired, then $B = A$). The various methods for computing the factor score coefficients are discussed in detail in Harman (1976, Chapter 16) and are given as follows:

1. The regression method may be used with any method of factor extraction and rotation (but not with image analysis). The coefficients are computed as follows:

$$\hat{\beta} = S^{-1}B(T^T T)^{-1}$$

2. The least-squares method may also be used with any method of factor extraction and rotation (but not in image analysis). The factor score coefficients are computed as

$$\hat{\beta} = B(B^T B)^{-1}$$

Note that estimated coefficients in the least-squares method yield different factor scores depending upon the scale of the observed variables. In particular, factor scores computed from standardized data (i.e., for the correlation matrix) will be different from factor scores computed from the raw data (i.e., from a covariance matrix). Generally, the differences will not be great. These differences are not observed in any of the other methods.

3. The Bartlett (1937) method may be used with common factor models only. The coefficients are computed as

$$\hat{\beta} = U^{-2}B(B^T U^{-2} B)^{-1}$$

4. The Anderson and Rubin (1956) method may also be used with common factor models only. It is a modification of the Bartlett method where the modification is used to insure that the factors obtained are orthogonal. The factor score coefficients are computed as

$$\hat{\beta} = U^{-2}B(B^T U^{-2} S U^{-2} B)^{-\frac{1}{2}}$$

5. The image method is appropriate for image analysis. In this method, the coefficients are computed as

$$\hat{\beta} = B_I T_I = A(T^T)^{-1}(T^T)^{-1}$$

where B_I is the image score coefficient matrix, and T_I is the image transformation matrix (the matrix TI in routine FIMAG, page 940).

Harman (1976, pages 385-387) discusses choosing a method for computing factor score coefficients. According to Harman, the most desirable properties of any of the methods can be summarized as follows.

- Validity—The estimated factor scores should have high correlation with the population factor scores.
- Orthogonality—The estimated factor scores should not correlate highly with one another.
- Univocal—The estimated factor scores should correlate only with the corresponding true factor scores.

With these criteria in mind, Harman states that:

1. The regression method yields factor scores which usually have the highest correlation with the true factor scores.
2. The Bartlett and least-squares methods are univocal but not orthogonal.
3. The Anderson and Rubin method is orthogonal but not univocal.
4. Univocality is of more significance than orthogonality.

FSCOR

Computes a set of factor scores given the factor score coefficient matrix.

Required Arguments

SCOEF — `NVAR` by `NF` matrix containing the factor score coefficients as output from routine `FCOEF` (page 944). (Input)

X — NOBS by NVAR data matrix for which factor scores are to be computed. (Input)

XBAR — Vector of length NVAR containing the means of the NVAR variables. (Input)

STD — Vector of length NVAR containing the standard deviations of the NVAR variables. (Input)
If STD(1) is not positive, then it is assumed that the factor score coefficients are from a covariance matrix and the observed variables are not standardized to unit variance.

SCOR — NOBS by NF matrix containing the factor scores. (Output)
If X is not needed, X and SCOR can share the same memory locations.

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (SCOEF,1).

NF — Number of factors. (Input)
Default: NF = size (SCOEF,2).

LDSCOE — Leading dimension of SCOEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSCOE = size (SCOEF, 1).

NOBS — Number of observations for which factor scores are to be computed. (Input)
Default: NOBS = size (X,1).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X, 1).

LDSCOR — Leading dimension of SCOR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSCOR = size (SCOR,1).

FORTRAN 90 Interface

Generic: `CALL FSCOR (SCOEF, X, XBAR, STD, SCOR [,…])`

Specific: The specific interface names are S_FSCOR and D_FSCOR.

FORTRAN 77 Interface

Single:
```
CALL FSCOR (NVAR, NF, SCOEF, LDSCOE, NOBS, X, LDX, XBAR,
STD, SCOR, LDSCOR)
```

Double: The double precision name is DFSCOR.

Example

The following example is a continuation of the example given in the manual document for routine FACTR (page 909). The rotated loadings are those obtained from the manual document for routine FROTA (page 918), and the factor score coefficients are as described in the manual document for routine FCOEF (page 944).

```
      USE FSCOR_INT
      USE WRRRN_INT
      INTEGER    LDSCOE, LDSCOR, LDX, NF, NOBS, NVAR
      PARAMETER  (LDSCOE=2, LDSCOR=5, LDX=5, NF=1, NOBS=5, NVAR=2)
!
      REAL       SCOEF(NVAR,NF), SCOR(LDSCOR,NF), STD(NVAR), &
      X(LDX,NVAR), XBAR(NVAR)
!
      DATA X/40.0, 60.0, 30.0, 15.0, 45.0, 3.0, 9.0, 2.0, 0.0, 4.0/
      DATA SCOEF/0.33563, 0.33562/
      DATA XBAR/38.0, 3.6/, STD/16.80774, 3.361547/
!
      CALL FSCOR (SCOEF, X, XBAR, STD, SCOR)
!
      CALL WRRRN ('Factor Scores', SCOR)
      END
```

Output

```
Factor Scores
1  -0.0200
2   0.9785
3  -0.3195
4  -0.8187
5   0.1797
```

Comments

Workspace may be explicitly provided, if desired, by use of F2COR/DF2COR. The reference is

```
CALL F2COR (NVAR, NF, SCOEF, LDSCOE, NOBS, X, LDX, XBAR, STD, SCOR,
LDSCOR, WK)
```

The additional argument is

WK — Work vector of length NVAR.

Description

Routine FSCOR computes the factor scores from the factor score coefficient matrix. In FSCOR, the data are input as originally observed, and standardization is performed as required according to the value of STD(1). When the factor loadings are computed from the correlation matrix, the observed data must be standardized to a mean of zero and a variance of one prior to computing the factor scores. This requires that STD contain the observed standard deviations of the observed data and that XBAR contain the means. On the other hand, if the factor loadings are computed from the covariance matrix, then the observed data must be standardized to a mean of

zero, but the variance must be left unchanged in computing the factor scores. In this case, STD(1) must be negative or zero.

After standardizing the observed data, the factor scores are computed as the product of the factor score coefficient matrix times the standardized data. If factor scores are computed from the same data from which the covariance matrix was computed, then the sample variance (using weights and frequencies as required) of the resulting factor scores will be 1.0.

FRESI

Computes communalities and the standardized factor residual correlation matrix.

Required Arguments

COV — NVAR by NVAR matrix containing the variance-covariance or correlation matrix. (Input)
Only the upper triangular part of COV is referenced.

A — NVAR by NF orthogonal factor-loading matrix. (Input)

Y — Vector of length NVAR containing the communalities. (Output)

RESID — NVAR by NVAR matrix containing the normalized residual variance-covariance or correlation matrix. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (COV,1).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV, 1).

NF — Number of factors. (Input)
Default: NF = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

LDRESI — Leading dimension of RESID exactly as specified in the dimension statement in the calling program. (Input)
Default: LDRESI = size (RESID,1).

FORTRAN 90 Interface

Generic: `CALL FRESI (COV, A, Y, RESID [,…])`

Specific: The specific interface names are `S_FRESI` and `D_FRESI`.

FORTRAN 77 Interface

Single: `CALL FRESI (NVAR, COV, LDCOV, NF, A, LDA, Y, RESID, LDRESI)`

Double: The double precision name is `DFRESI`.

Example

The following example computes the residual correlation matrix with communalities in a 9-factor problem. The resulting residual correlations do not seem to exhibit any pattern.

```
      USE FRESI_INT
      USE WRRRN_INT

      INTEGER    LDA, LDCOV, LDRESI, NF, NVAR
      PARAMETER  (LDA=9, LDCOV=9, LDRESI=9, NF=3, NVAR=9)
!
      REAL       A(9,3), COV(9,9), RESID(9,9), Y(9)
!
      DATA COV/1.000, 0.523, 0.395, 0.471, 0.346, 0.426, 0.576, 0.434, &
      0.639, 0.523, 1.000, 0.479, 0.506, 0.418, 0.462, 0.547, &
      0.283, 0.645, 0.395, 0.479, 1.000, 0.355, 0.270, 0.254, &
      0.452, 0.219, 0.504, 0.471, 0.506, 0.355, 1.000, 0.691, &
      0.791, 0.443, 0.285, 0.505, 0.346, 0.418, 0.270, 0.691, &
      1.000, 0.679, 0.383, 0.149, 0.409, 0.426, 0.462, 0.254, &
      0.791, 0.679, 1.000, 0.372, 0.314, 0.472, 0.576, 0.547, &
      0.452, 0.443, 0.383, 0.372, 1.000, 0.385, 0.680, 0.434, &
      0.283, 0.219, 0.285, 0.149, 0.314, 0.385, 1.000, 0.470, &
      0.639, 0.645, 0.504, 0.505, 0.409, 0.472, 0.680, 0.470, &
      1.000/
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960, &
      -.2955, -.4274, .0735, -.1933, -.2224, -.0354, -.1528, &
      .1045, -.0778, .4914, -.0117/
!
      CALL FRESI (COV, A, Y, RESID)
!
      CALL WRRRN ('Communalities', Y, 1, NVAR, 1)
      CALL WRRRN ('Residuals', RESID)
      END
```

Output

```
                         Communalities
     1        2        3        4        5        6        7        8
0.5495   0.5729   0.3834   0.7877   0.6195   0.8231   0.6005   0.5385

     9
0.7691

                              Residuals
        1       2       3       4       5       6       7       8       9
```

```
1   1.000   0.001  -0.024   0.037  -0.024  -0.016   0.036  -0.002  -0.018
2   0.001   1.000   0.043  -0.017  -0.048   0.041  -0.052  -0.023   0.031
3  -0.024   0.043   1.000   0.064  -0.033  -0.037  -0.022   0.025  -0.013
4   0.037  -0.017   0.064   1.000   0.012  -0.004   0.008   0.017  -0.052
5  -0.024  -0.048  -0.033   0.012   1.000  -0.003   0.075  -0.014   0.007
6  -0.016   0.041  -0.037  -0.004  -0.003   1.000  -0.046  -0.003   0.036
7   0.036  -0.052  -0.022   0.008   0.075  -0.046   1.000   0.008   0.011
8  -0.002  -0.023   0.025   0.017  -0.014  -0.003   0.008   1.000  -0.004
9  -0.018   0.031  -0.013  -0.052   0.007   0.036   0.011  -0.004   1.000
```

Description

Routine FRESI computes the communalities and a standardized residual covariance/correlation matrix for input covariance/correlation matrix COV. The user must also input the orthogonal (unrotated) factor loadings, A, obtained from the matrix COV. Let a_i denote the i-th row of matrix A. Then, the communalities are given as

$$y_i = a_i a_i^T$$

where y_i is the i-th communality. The residual covariance/correlation matrix is given by

$$r_{ij} = s_{ij} - a_i a_j^T$$

where s_{ij} denotes an element of the covariance/correlation matrix and $R = (r_{ij})$ denotes the residual matrix. Standardization is performed by dividing the r_{ij} by

$$\sqrt{u_i u_j}$$

where $u_i = s_{ii} - y_i$ is the unique error variance for the i-th variable. If u_i is zero (or slightly less than zero due to roundoff error), $u_i = 1.0$ is assumed and division by zero is avoided.

MVIND

Computes a test for the independence of k sets of multivariate normal variables.

Required Arguments

NDF — Number of degrees of freedom in COV. (Input)

COV — NVAR by NVAR variance-covariance matrix. (Input)

NVSET — Index vector of length NGROUP. (Input)
NVSET(i) gives the number of variables in the i-th set of variables. The first NVSET(1) variables in COV define the first set of covariates, the next NVSET(2) variables define the second set of covariates, etc.

STAT — Vector of length 4 containing the output statistics. (Output)

I	STAT(I)
1	Statistic *V* for testing the hypothesis of independence of the NGROUP sets of variables.
2	Chi-squared statistic associated with *V*.
3	Degrees of freedom for STAT(2).
4	Probability of exceeding STAT(2) under the null hypothesis of independence.

Optional Arguments

NVAR — Number of variables in the covariance matrix. (Input)
Default: NVAR = size (COV,2).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

NGROUP — Number of sets of variables to be tested for independence. (Input)
Default: NGROUP = size (NVSET,1).

FORTRAN 90 Interface

Generic: `CALL MVIND (NDF, COV, NVSET, STAT [,…])`

Specific: The specific interface names are S_MVIND and D_MVIND.

FORTRAN 77 Interface

Single: `CALL MVIND (NDF, NVAR, COV, LDCOV, NGROUP, NVSET, STAT)`

Double: The double precision name is DMVIND.

Example

The example is taken from Morrison (1976, page 258). It involves two sets of covariates, with each set having two covariates. The null hypothesis of no relationship is rejected.

```
      USE MVIND_INT
      USE UMACH_INT
      INTEGER    LDCOV, NDF, NGROUP, NVAR
      PARAMETER  (NDF=932, NGROUP=2, NVAR=4, LDCOV=NVAR)
!
      INTEGER    NOUT, NVSET(NGROUP)
      REAL       COV(NVAR,NVAR), STAT(4)
!
      DATA COV/1.00, 0.45, -0.19, 0.43, 0.45, 1.00, -0.02, 0.62, &
```

```
     -0.19, -0.02, 1.00, -0.29, 0.43, 0.62, -0.29, 1.00/
!
      DATA NVSET/2, 2/
!
      CALL MVIND (NDF, COV, NVSET, STAT)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STAT
99999 FORMAT (' Likelihood ratio ........... ', F12.4, /, ' ', &
       'Chi-squared ................ ', F9.1, /, ' Degrees of '&
      , 'freedom ......... ', F9.1, /, ' p-value ', &
        '..................... ', F12.4)
      END
```

Output

```
Likelihood ratio ...........       0.5497
Chi-squared ................     556.2
Degrees of freedom .........       4.0
p-value ....................       0.0000
```

Comments

1. Workspace may be explicitly provided, if desired, by use of M2IND/DM2IND. The reference is:

   ```
   CALL M2IND (NDF, NVAR, COV, LD COV, NGROUP, NVSET, STAT, FACT,
   WK, IPVT)
   ```

   The additional arguments are as follows:

   FACT — Work vector of length $\texttt{NVAR}^2$.

   WK — Work vector of length NVAR.

   IPVT — Work vector of length NVAR.

2. Informational errors

Type	Code	
4	1	A covariance matrix for a subset of the variables is singular.
4	2	The covariance matrix for all variables is singular.

Description

Routine MVIND computes a likelihood ratio test statistic proposed by Wilks (1935) for testing the independence of NGROUP sets of multivariate normal variates. The likelihood ratio statistic is computed as the ratio of the determinant $|S|$ of the sample covariance matrix to the product of the determinants $|S_1|\ldots|S_K|$ of the covariance matrices of each of the k = NGROUP sets of variates. An asymptotic chi-squared statistic obtained from the likelihood ratio, along with corresponding p-value, is computed according to formulas given by Morrison (1976, pages 258-259). The chi-squared statistic is computed as:

$$\chi^2 = -\frac{n}{C}\ln(V)$$

where n = NDF,

$$\begin{aligned} V &= \frac{|S|}{|S_{11}|\cdots|S_{kk}|} \\ C^{-1} &= 1 - \frac{2\sigma_2 + 3\sigma_3}{6n\sigma_2} \\ \sigma_2 &= \left(\sum_{i=1}^{k} p_i\right)^2 - \sum_{i=1}^{k} p_i^2 \\ \sigma_3 &= \left(\sum_{i=1}^{k} p_i\right)^3 - \sum_{i=1}^{k} p_i^3 \end{aligned}$$

where $|S_{ii}|$ is the determinant of the i-th covariance matrix, k = NGROUP, and p_i = NVSET(i), and $|S|$ is the determinant of COV.

Because determinants appear in both the numerator and denominator of the likelihood ratio, the test statistic is unchanged when correlation matrices are substituted for covariance matrices as input to MVIND.

In using MVIND, the covariance matrix must first be computed (possibly via routine CORVC see Chapter 3, Correlation). The covariance matrix may then need to be rearranged (possible via routine RORDM, page 1268) so that the NVSET(1) variables in the first set correspond to the first NVSET(1) columns (and rows) of the covariance matrix, with the next NVSET(2) columns and rows containing the variables for the second set of variables, etc. With this special arrangement of the covariance matrix, routine MVIND may then be called.

CANCR

Performs canonical correlation analysis from a data matrix.

Required Arguments

X — NOBS by NVAR1 + NVAR2 + m data matrix where m is 0, 1, or 2 depending on whether any columns of X correspond to frequencies or weights. (Input)
Each row of X contains an observation of the NVAR1 + NVAR2 variables for which canonical correlations are desired (plus a weight and/or a frequency variable if IFRQ and/or IWT(see below) are not zero). If both IWT and IFRQ are zero, m is 0; 1, if one of IFRQ or IWT is positive; and 2, otherwise. X may not have any missing values (NaN, not a number).

IND1 — Vector of length NVAR1 containing the column numbers in X of the group 1 variables. (Input)

IND2 — Vector of length NVAR2 containing the column numbers in X of the group 2 variables. (Input)

XX — NOBS by NVAR1 + NVAR2 + *m* matrix containing the canonical scores. (Output)
m is defined in the description for X. X and XX may occupy the same storage locations. Canonical scores are returned in the first NVAR1 + NVAR2 columns of XX. Scores for the NVAR1 variables come first. If one of IFRQ or IWT are not zero, then the last column of XX contains the weight or frequency. If both IFRQ and IWT are not zero, then the frequencies and weights are in the second to last and last column of XX, respectively.

CORR — NV by 6 matrix of output statistics. (Output)
NV is the minimum of NVAR1 and NVAR2. CORR has the following statistics.

Col.	Statistic
1	Canonical correlations sorted from the largest to the smallest.
2	Wilks' lambda for testing that the current and all smaller canonical correlations are zero.
3	Rao's *F* corresponding to Wilks' lambda. If the canonical correlation is greater than 0.99999, then *F* is set to 9999.99.
4	Numerator degrees of freedom for *F*.
5	Denominator degrees of freedom for *F*.
6	Probability of a larger *F* statistic.

If an *F* statistic is negative, then CORR(*i*, 6) is set to one. If either CORR(*i*, 4) or CORR(*i*, 5) is not positive, then CORR(*i*, 6) is set to the missing value code (NaN).

COEF1 — NVAR1 by NVAR1 matrix containing the group 1 canonical coefficients. (Output)
The columns of COEF1 contain the vectors of canonical coefficients for group 1.

COEF2 — NVAR2 by NVAR2 matrix containing the group 2 canonical coefficients. (Output)
The columns of COEF2 contain the vectors of canonical coefficients for group 2.

COEFR1 — NVAR1 by NV matrix containing the correlations between the group 1 variables and the group 1 canonical scores. (Output)
NV is the minimum of NVAR1 and NVAR2.

COEFR2 — NVAR2 by NV matrix containing the correlations between the group 2 variables and the group 2 canonical scores. (Output)
NV is the minimum of NVAR1 and NVAR2.

STAT — 15 by NVAR1 + NVAR2 matrix containing statistics on all of the variables. (Output)
The first NVAR1 columns of STAT correspond to the group one variables with the last NVAR2 columns corresponding to the group two variables.

Row	Statistic
1	Means
2	Variances
3	Standard deviations
4	Coefficients of skewness
5	Coefficients of excess (kurtosis)
6	Minima
7	Maxima
8	Ranges
9	Coefficients of variation, when defined, 0.0 otherwise
10	Numbers of nonmissing observations
11	Lower endpoints of 95% confidence interval for the means
12	Upper endpoints of 95% confidence interval for the means
13	Lower endpoints of 95% confidence interval for the variances
14	Upper endpoints of 95% confidence interval for the variances
15	Sums of the weights if `IWT` greater than zero, 0.0 otherwise

Optional Arguments

NOBS — Number of observations. (Input)
Default: `NOBS` = size (`X`,1).

NVAR1 — Number of variables in group 1. (Input)
Default: `NVAR1` = size (`IND1`,1).

NVAR2 — Number of variables in group 2. (Input)
Default: `NVAR2` = size (`IND2`,1).

NCOL — Number of columns in `X`. (Input)
Default: `NCOL` = size(`X`,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)
If IFRQ = 0, then all frequencies are 1. If IFRQ is positive, then column number IFRQ of X contains the nonnegative frequencies.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
If IWT = 0, then there is no weighting, i.e., all weights are 1. If IWT is positive, then column number IWTof X contains the nonnegative weights.
Default: IWT = 0.

TOL — Constant used for determining linear dependence. (Input)
If the squared multiple correlation coefficient of a variable with its predecessors in IND1 (or IND2) is greater than 1 − TOL, then the variable is considered to be linearly dependent upon the previous variables; it is excluded from the analysis. TOL = .001 is a typical value. TOL must be in the exclusive range of 0.0 to 1.0.
Default: TOL = .001.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing.
1	Print CORR, COEF1, COEF2, COEFR1, COEFR2, and STAT.
2	Print all output.

LDXX — Leading dimension of XX exactly as specified in the dimension statement in the calling program. (Input)
Default: LDXX = size (XX,1).

LDCORR — Leading dimension of CORR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCORR = size (CORR,1).

LDCOF1 — Leading dimension of COEF1 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOF1 = size (COEF1,1).

LDCOF2 — Leading dimension of COEF2 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOF2 = size (COEF2,1).

LDCFR1 — Leading dimension of COEFR1 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCFR1 = size (COEFR1,1).

LDCFR2 — Leading dimension of COEFR2 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCFR2 = size (COEFR2,1).

LDSTAT — Leading dimension of STAT exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSTAT = size(STAT,1).

FORTRAN 90 Interface

Generic: `CALL CANCR (X, IND1, IND2, XX, CORR, COEF1, COEF2, COEFR1, COEFR2, STAT [,…])`

Specific: The specific interface names are S_CANCR and D_CANCR.

FORTRAN 77 Interface

Single:
```
CALL CANCR (NOBS, NVAR1, NVAR2, NCOL, X, LDX, IFRQ, IWT,
IND1, IND2, TOL, IPRINT, XX, LDXX, CORR, LDCORR, COEF1,
LDCOF1, COEF2, LDCOF2, COEFR1, LDCFR1, COEFR2, LDCFR2,
STAT, LDSTAT)
```

Double: The double precision name is DCANCR.

Example 1

The following example is taken from Levin and Marascuilo (1983), pages 191–197. It is examining the relationship between the performance of individuals in a sociology course and predictor variables. The measures of performance in the sociology course are two midterms examinations, a final examination, and a course evaluation, the predictor variables are social class, sex, grade point average, college board test score, whether the student has previously taken a course in sociology, and the student's score on a pretest.

```
      USE WRRRL_INT
      USE CANCR_INT
      INTEGER    IPRINT, LDCFR1, LDCFR2, LDCOF1, LDCOF2, LDCORR, LDSTAT,&
                 LDX, LDXX, NCOL, NOBS, NV, NVAR1, NVAR2
      REAL       TOL
      PARAMETER  (IPRINT=1, LDSTAT=15, NCOL=10, NOBS=40, NVAR1=6, &
                  NVAR2=4, TOL=0.0001, LDCFR1=NVAR1, LDCFR2=NVAR2, &
                  LDCOF1=NVAR1, LDCOF2=NVAR2, LDX=NOBS, LDXX=NOBS, &
                  NV=NVAR2, LDCORR=NV)
!
      INTEGER    IND1(NVAR1), IND2(NVAR2)
      REAL       COEF1(LDCOF1,NVAR1), COEF2(LDCOF2,NVAR2), &
                 COEFR1(LDCFR1,NV), COEFR2(LDCFR2,NV), &
```

```
                CORR(LDCORR,6),STAT(LDSTAT,NVAR1+NVAR2), &
                X(LDX,NCOL), XX(LDXX,NCOL)
      CHARACTER  FMT*35, NUMBER(1)*6, XLAB(11)*25
!
      DATA IND1/1, 2, 3, 4, 5, 6/, IND2/7, 8, 9, 10/
      DATA (X(I,1),I=1,NOBS)/3*2.0, 3.0, 2.0, 3.0, 1.0, 2.0, 3.0, &
      2*2.0, 3.0, 1.0, 4*2.0, 3.0, 3*2.0, 1.0, 3*2.0, 1.0, 2.0, &
      1.0, 2.0, 3.0, 2*2.0, 2*1.0, 2.0, 3.0, 1.0, 2.0, 3.0, 1.0/
      DATA (X(I,2),I=1,NOBS)/6*1.0, 0.0, 2*1.0, 3*0.0, 3*1.0, 3*0.0, &
      1.0, 0.0, 3*1.0, 3*0.0, 4*1.0, 0.0, 8*1.0, 0.0/
      DATA (X(I,3),I=1,NOBS)/3.55, 2.70, 3.50, 2.91, 3.10, 3.49, 3.17, &
      3.57, 3.76, 3.81, 3.60, 3.10, 3.08, 3.50, 3.43, 3.39, 3.76, &
      3.71, 3.00, 3.47, 3.69, 3.24, 3.46, 3.39, 3.90, 2.76, 2.70, &
      3.77, 4.00, 3.40, 3.09, 3.80, 3.28, 3.70, 3.42, 3.09, 3.70, &
      2.69, 3.40, 2.95/
      DATA (X(I,4),I=1,NOBS)/410.0, 390.0, 510.0, 430.0, 600.0, &
      2*610.0, 560.0, 700.0, 460.0, 590.0, 500.0, 410.0, 470.0, &
      210.0, 610.0, 510.0, 600.0, 470.0, 460.0, 800.0, 610.0, &
      490.0, 470.0, 610.0, 580.0, 410.0, 630.0, 790.0, 490.0, &
      400.0, 2*610.0, 500.0, 430.0, 540.0, 610.0, 400.0, 390.0, &
      490.0/
      DATA (X(I,5),I=1,NOBS)/8*0.0, 4*1.0, 0.0, 2*1.0, 0.0, 1.0, 0.0, &
      1.0, 0.0, 1.0, 3*0.0, 1.0, 2*0.0, 2*1.0, 2*0.0, 4*1.0, &
      5*0.0/
      DATA (X(I,6),I=1,NOBS)/17.0, 20.0, 22.0, 13.0, 16.0, 28.0, 14.0, &
      10.0, 28.0, 30.0, 28.0, 15.0, 24.0, 15.0, 26.0, 16.0, 25.0, &
      3.0, 5.0, 16.0, 28.0, 13.0, 9.0, 13.0, 30.0, 10.0, 13.0, &
      8.0, 29.0, 17.0, 15.0, 16.0, 13.0, 30.0, 2*17.0, 25.0, &
      10.0, 23.0, 18.0/
      DATA (X(I,7),I=1,NOBS)/43.0, 50.0, 47.0, 24.0, 47.0, 57.0, &
      2*42.0, 69.0, 48.0, 59.0, 21.0, 52.0, 2*35.0, 59.0, 68.0, &
      38.0, 45.0, 37.0, 54.0, 45.0, 31.0, 39.0, 67.0, 30.0, 19.0, &
      71.0, 80.0, 47.0, 46.0, 59.0, 48.0, 68.0, 43.0, 31.0, 64.0, &
      19.0, 43.0, 20.0/
      DATA (X(I,8),I=1,NOBS)/61.0, 47.0, 79.0, 40.0, 60.0, 59.0, 61.0, &
      79.0, 83.0, 67.0, 74.0, 40.0, 71.0, 40.0, 57.0, 58.0, 66.0, &
      58.0, 24.0, 48.0, 100.0, 83.0, 70.0, 48.0, 85.0, 14.0, &
      55.0, 100.0, 94.0, 45.0, 58.0, 90.0, 84.0, 81.0, 49.0, &
      54.0, 87.0, 36.0, 51.0, 59.0/
       DATA (X(I,9),I=1,NOBS)/129.0, 60.0, 119.0, 100.0, 79.0, 99.0, &
       92.0, 107.0, 156.0, 110.0, 116.0, 49.0, 107.0, 125.0, 64.0, &
       100.0, 138.0, 63.0, 82.0, 73.0, 132.0, 87.0, 89.0, 99.0, &
       119.0, 100.0, 84.0, 166.0, 111.0, 110.0, 93.0, 141.0, 99.0, &
       114.0, 96.0, 39.0, 149.0, 53.0, 39.0, 91.0/
      DATA (X(I,10),I=1,NOBS)/3.0, 3*1.0, 2.0, 1.0, 3.0, 2.0, 4*1.0, &
      5.0, 1.0, 5.0, 1.0, 2.0, 1.0, 2*3.0, 3*2.0, 1.0, 2.0, 1.0, &
      2.0, 3.0, 2.0, 2*1.0, 2*2.0, 5.0, 2*1.0, 4.0, 3.0, 2*1.0/
!
      DATA XLAB/' ', 'Social%/Class', '%/Sex', '%/GPA', &
      'College%/Boards', 'H.S.%/Soc.', 'Pretest%/Score', &
      '%/Exam 1', '%/Exam 2', 'Final%/Exam', 'Course%/Eval.'/
      DATA NUMBER/'NUMBER'/, FMT/'(2W3.1,W5.3,W4.1,W3.1,4W5.1,W3.1)'/
!
      CALL WRRRL ('First 10 Observations', X, NUMBER, XLAB, &
                  10, NCOL, LDX, FMT=FMT)
!
```

```
      CALL CANCR (X, IND1, IND2, XX, CORR, COEF1, &
      COEF2, COEFR1, COEFR2, STAT, TOL=TOL, IPRINT=IPRINT)
!
      END
```

Output

```
                                First 10 Observations
    Social                  College  H.S.  Pretest                  Final Course
     Class   Sex    GPA     Boards  Soc.    Score  Exam 1  Exam 2  Exam   Eval
 1       2    1   3.55        410     0       17      43      61    129      3
 2       2    1   2.70        390     0       20      50      47     60      1
 3       2    1   3.50        510     0       22      47      79    119      1
 4       3    1   2.91        430     0       13      24      40    100      1
 5       2    1   3.10        600     0       16      47      60     79      2
 6       3    1   3.49        610     0       28      57      59     99      1
 7       1    0   3.17        610     0       14      42      61     92      3
 8       2    1   3.57        560     0       10      42      79    107      2
 9       3    1   3.76        700     1       28      69      83    156      1
10       2    0   3.81        460     1       30      48      67    110      1

              *** Canonical Correlations Statistics ***
      Canonical                                                 Prob. of
   Correlations  Wilks Lambda      Raos F  Num. df  Denom. df  Larger F
1        0.9242        0.0612       5.412       24      105.9    0.0000
2        0.7184        0.4201       2.116       15       86.0    0.0162
3        0.2893        0.8683       0.586        8       64.0    0.7861
4        0.2290        0.9476       0.609        3       33.0    0.6142

        Group One Canonical Coefficients
         1       2       3       4       5       6
1  -0.622   1.158  -0.285  -0.179   0.601  -0.423
2   0.558  -0.739   0.231  -1.278   1.391  -0.024
3   1.796  -0.432   0.765   0.185  -0.643  -3.314
4   0.002   0.006   0.004  -0.002   0.000   0.006
5  -0.059  -0.043  -0.456   1.671   1.463   0.774
6   0.031   0.018  -0.121  -0.058  -0.042   0.056

  Group Two Canonical Coefficients
         1        2        3        4
1   0.0233  -0.0365   0.0845  -0.0176
2   0.0257  -0.0057  -0.0352   0.0555
3   0.0073   0.0110  -0.0259  -0.0341
4   0.1034   0.8089   0.2828   0.0260

Correlations Between the Group One Variables
and the Group One Canonical Scores
         1        2        3        4
1  -0.3685   0.6795  -0.2291  -0.1854
2   0.2157  -0.3252   0.0521  -0.5985
3   0.8153   0.2770  -0.0692   0.2123
4   0.6144   0.5681   0.4151  -0.0050
5   0.4661   0.0603  -0.3034   0.6530
6   0.5461   0.1768  -0.7915  -0.1375

Correlations Between the Group Two Variables
```

and the Group Two Canonical Scores

	1	2	3	4
1	0.8713	-0.2406	0.3864	-0.1835
2	0.9174	-0.0557	-0.2068	0.3355
3	0.7707	0.0293	-0.3146	-0.5533
4	0.3490	0.8765	0.3077	0.1240

*** Statistics for Group One Variables ***
Univariate Statistics from UVSTA

Variable	Mean	Variance	Std. Dev.	Skewness	Kurtosis
1	1.9750	0.4353	0.6597	0.02476	-0.6452
2	0.6750	0.2250	0.4743	-0.74726	-1.4416
3	3.3758	0.1247	0.3532	-0.37911	-0.7521
4	524.2499	13148.1377	114.6653	0.09897	0.6494
5	0.4000	0.2462	0.4961	0.40825	-1.8333
6	18.1250	55.1378	7.4255	0.10633	-0.9358

Variable	Minimum	Maximum	Range	Coef. Var.	Count
1	1.0000	3.0000	2.0000	0.3340	40.0000
2	0.0000	1.0000	1.0000	0.7027	40.0000
3	2.6900	4.0000	1.3100	0.1046	40.0000
4	210.0000	800.0000	590.0000	0.2187	40.0000
5	0.0000	1.0000	1.0000	1.2403	40.0000
6	3.0000	30.0000	27.0000	0.4097	40.0000

Variable	Lower CLM	Upper CLM	Lower CLV	Upper CLV
1	1.7640	2.1860	0.29207	0.7176
2	0.5233	0.8267	0.15098	0.3710
3	3.2628	3.4887	0.08369	0.2056
4	487.5782	560.9217	8822.72168	21677.9590
5	0.2413	0.5587	0.16518	0.4058
6	15.7502	20.4998	36.99883	90.9083

*** Statistics for Group Two Variables ***
Univariate Statistics from UVSTA

Variable	Mean	Variance	Std. Dev.	Skewness	Kurtosis
1	46.0500	237.0231	15.3956	0.08762	-0.5505
2	62.8750	403.4967	20.0872	-0.10762	-0.3642
3	99.4750	919.4864	30.3230	-0.03483	-0.2533
4	1.9500	1.4333	1.1972	1.27704	0.8407

Variable	Minimum	Maximum	Range	Coef. Var.	Count
1	19.0000	80.0000	61.0000	0.3343	40.0000
2	14.0000	100.0000	86.0000	0.3195	40.0000
3	39.0000	166.0000	127.0000	0.3048	40.0000
4	1.0000	5.0000	4.0000	0.6140	40.0000

Variable	Lower CLM	Upper CLM	Lower CLV	Upper CLV
1	41.1263	50.9737	159.0483	390.7912
2	56.4508	69.2992	270.7562	665.2642
3	89.7772	109.1728	616.9979	1516.0009
4	1.5671	2.3329	0.9618	2.3632

Example 2

Correspondence analysis is an interesting application of canonical correlation in the analysis of contingency tables. The example is taken from Kendall and Stuart (1979, pages 595–599) and involves finding the optimal scores for the values of two categorical variables to maximize the correlation between the two variables. The contingency table is given below, along with the more traditional matrix X of "observations" for which canonical correlations are desired.

$$\begin{pmatrix} 821 & 112 & 85 & 35 \\ 116 & 494 & 145 & 27 \\ 72 & 151 & 583 & 87 \\ 43 & 34 & 106 & 331 \end{pmatrix}$$

The data matrix X is given as:

Group 1 Var.				Group 2 Var.				Frequencies
1	0	0	0	1	0	0	0	821
1	0	0	0	0	1	0	0	112
1	0	0	0	0	0	1	0	85
1	0	0	0	0	0	0	1	35
0	1	0	0	1	0	0	0	116
0	1	0	0	0	1	0	0	494
0	1	0	0	0	0	1	0	145
0	1	0	0	0	0	0	1	27
0	0	1	0	1	0	0	0	72
0	0	1	0	0	1	0	0	151
0	0	1	0	0	0	1	0	583
0	0	1	0	0	0	0	1	87
0	0	0	1	1	0	0	0	43
0	0	0	1	0	1	0	0	34
0	0	0	1	0	0	1	0	106
0	0	0	1	0	0	0	1	331

For this table, the optimal correlation turns out to be 0.70 when scores of 2.67, 1.34, 0.62, and 0.00 (see Column 1 of COEF1) are assigned to the variable 1 categories, and scores of 2.72, 1.37, 0.68, and 0.00 are assigned to the variable 2 categories. These scores are obtained as the canonical scores when canonical correlations are computed between the the row and column variable indicator variables (variables 1-4 and variables 5-8 in X, respectively). The warning error appears in the output because the covariance matrix is not of full rank (indeed, neither the group 1 or the group 2 covariance matrices are of full rank).

```
USE CANCR_INT
INTEGER    IFRQ, IPRINT, LDCFR1, LDCFR2, LDCOF1, LDCOF2, &
           LDCORR, LDSTAT, LDX, LDXX, NCOL, NOBS, NV, NVAR1, &
```

```
                NVAR2
      REAL      TOL
      PARAMETER (IFRQ=9, IPRINT=2, LDCFR1=4, LDCFR2=4, &
                LDCOF1=4, LDCOF2=4, LDCORR=4, LDSTAT=15, LDX=16, &
                LDXX=16, NCOL=9, NOBS=16, NV=4, NVAR1=4, NVAR2=4, &
                TOL=0.0001)
!
      INTEGER   IND1(NVAR1), IND2(NVAR2)
      REAL      COEF1(LDCOF1,NVAR1), COEF2(LDCOF2,NVAR2), &
                COEFR1(LDCFR1,NV), COEFR2(LDCFR2,NV), CORR(LDCORR,6), &
                STAT(LDSTAT,8), X(LDX,NCOL), XX(LDXX,NCOL)
!
      DATA IND1/1, 2, 3, 4/, IND2/5, 6, 7, 8/
      DATA X/4*1.0, 16*0.0, 4*1.0, 16*0.0, 4*1.0, 16*0.0, 5*1.0, &
          3*0.0, 1.0, 3*0.0, 1.0, 3*0.0, 1.0, 4*0.0, 1.0, 3*0.0, 1.0, &
          3*0.0, 1.0, 3*0.0, 1.0, 4*0.0, 1.0, 3*0.0, 1.0, 3*0.0, 1.0, &
          3*0.0, 1.0, 4*0.0, 1.0, 3*0.0, 1.0, 3*0.0, 1.0, 3*0.0, 1.0, &
          821.0, 112.0, 85.0, 35.0, 116.0, 494.0, 145.0, 27.0, 72.0, &
          151.0, 583.0, 87.0, 43.0, 34.0, 106.0, 331.0/
!
      CALL CANCR (X, IND1, IND2, XX, CORR, COEF1, &
                COEF2, COEFR1, COEFR2, STAT, IFRQ=IFRQ, &
                TOL=TOL, IPRINT=IPRINT)
!
      END
```

Output

```
*** WARNING  ERROR 2 from C2NCR.  One or more Group 1 variables is linearly
***          dependent on the proceeding variables in Group 1.
Here is a traceback of subprogram calls in reverse order:
Routine name                    Error type  Error code
------------                    ----------  ----------
C2NCR                               6           2    (Called internally)
CANCR                               0           0
USER                                0           0

*** WARNING  ERROR 3 from C2NCR.  One or more Group 2 variables is linearly
***          dependent on the proceeding variables in Group 2.
Here is a traceback of subprogram calls in reverse order:
Routine name                    Error type  Error code
------------                    ----------  ----------
C2NCR                               6           3    (Called internally)
CANCR                               0           0
USER                                0           0

          *** Canonical Correlations Statistics ***
      Canonical                                                 Prob. of
   Correlations  Wilks Lambda      Raos F  Num. df  Denom. df  Larger F
1        0.6965        0.2734     615.925        9     7875.7    0.0000
2        0.5883        0.5310     602.598        4     6474.0    0.0000
3        0.4336        0.8120     749.823        1     3238.0    0.0000
4        0.0000        0.0000       0.000        0        0.0    0.0000

Group One Canonical Coefficients
        1       2       3       4
```

```
1   2.670   1.100   1.023   0.000
2   1.341   2.905  -0.460   0.000
3   0.624   2.222   2.147   0.000
4   0.000   0.000   0.000   0.000

Group Two Canonical Coefficients
        1       2       3       4
1   2.715   1.164   1.053   0.000
2   1.366   2.972  -0.393   0.000
3   0.676   2.250   2.182   0.000
4   0.000   0.000   0.000   0.000

Correlations Between the Group One Variables
     and the Group One Canonical Scores
         1       2       3       4
1   0.9068  -0.3954   0.1459   0.0000
2  -0.0121   0.6965  -0.7175   0.0000
3  -0.4555   0.3404   0.8226   0.0000
4   0.0000   0.0000   0.0000   0.0000

Correlations Between the Group Two Variables
     and the Group Two Canonical Scores
         1       2       3       4
1   0.9072  -0.3997   0.1310   0.0000
2  -0.0227   0.6995  -0.7143   0.0000
3  -0.4590   0.3205   0.8287   0.0000
4   0.0000   0.0000   0.0000   0.0000

                 *** Statistics for Group One Variables ***
                     Univariate Statistics from UVSTA

Variable          Mean      Variance    Std. Dev.     Skewness     Kurtosis
    1           0.3248        0.2194       0.4684       0.7482      -1.4401
    2           0.2412        0.1831       0.4279       1.2098      -0.5363
    3           0.2754        0.1996       0.4468       1.0053      -0.9894
    4           0.1585        0.0000       0.0000       1.8697       1.4958
Variable       Minimum       Maximum        Range   Coef. Var.        Count
    1           0.0000        1.0000       1.0000       1.4420    3242.0000
    2           0.0000        1.0000       1.0000       1.7739    3242.0000
    3           0.0000        1.0000       1.0000       1.6221    3242.0000
    4           0.0000        1.0000       1.0000       2.3041    3242.0000

Variable     Lower CLM     Upper CLM     Lower CLV     Upper CLV
    1           0.3087        0.3409        0.2091        0.2305
    2           0.2265        0.2559        0.1745        0.1923
    3           0.2601        0.2908        0.1903        0.2097
    4           0.1460        0.1711        0.1272        0.1402

Canonical Scores for Group One
         1       2       3       4
 1   1.307  -0.570   0.210   0.000
 2   1.307  -0.570   0.210   0.000
 3   1.307  -0.570   0.210   0.000
 4   1.307  -0.570   0.210   0.000
 5  -0.021   1.235  -1.272   0.000
 6  -0.021   1.235  -1.272   0.000
```

7	-0.021	1.235	-1.272	0.000
8	-0.021	1.235	-1.272	0.000
9	-0.739	0.552	1.334	0.000
10	-0.739	0.552	1.334	0.000
11	-0.739	0.552	1.334	0.000
12	-0.739	0.552	1.334	0.000
13	-1.362	-1.670	-0.813	0.000
14	-1.362	-1.670	-0.813	0.000
15	-1.362	-1.670	-0.813	0.000
16	-1.362	-1.670	-0.813	0.000

*** Statistics for Group Two Variables ***
Univariate Statistics from UVSTA

Variable	Mean	Variance	Std. Dev.	Skewness	Kurtosis
1	0.3245	0.2193	0.4683	0.7497	-1.4379
2	0.2440	0.1845	0.4296	1.1922	-0.5787
3	0.2835	0.2032	0.4508	0.9609	-1.0766
4	0.1481	0.0000	0.0000	1.9819	1.9280

Variable	Minimum	Maximum	Range	Coef. Var.	Count
1	0.0000	1.0000	1.0000	1.4430	3242.0000
2	0.0000	1.0000	1.0000	1.7606	3242.0000
3	0.0000	1.0000	1.0000	1.5901	3242.0000
4	0.0000	1.0000	1.0000	2.3992	3242.0000

Variable	Lower CLM	Upper CLM	Lower CLV	Upper CLV
1	0.3084	0.3406	0.2090	0.2303
2	0.2292	0.2588	0.1758	0.1938
3	0.2679	0.2990	0.1936	0.2134
4	0.1358	0.1603	0.1203	0.1326

Canonical Scores for Group Two

	1	2	3	4
1	1.309	-0.577	0.189	0.000
2	-0.040	1.231	-1.257	0.000
3	-0.730	0.509	1.317	0.000
4	-1.406	-1.740	-0.864	0.000
5	1.309	-0.577	0.189	0.000
6	-0.040	1.231	-1.257	0.000
7	-0.730	0.509	1.317	0.000
8	-1.406	-1.740	-0.864	0.000
9	1.309	-0.577	0.189	0.000
10	-0.040	1.231	-1.257	0.000
11	-0.730	0.509	1.317	0.000
12	-1.406	-1.740	-0.864	0.000
13	1.309	-0.577	0.189	0.000
14	-0.040	1.231	-1.257	0.000
15	-0.730	0.509	1.317	0.000
16	-1.406	-1.740	-0.864	0.000

*** WARNING ERROR 1 from CANCR. The standardized cross covariance matrix
*** is not of full rank or is very ill-conditioned. Small
*** canonical correlations may not be accurate.

Comments

1. Workspace may be explicitly provided, if desired, by use of C2NCR/DC2NCR. The reference is:

```
CALL C2NCR (NOBS, NVAR1, NVAR2, NCOL, X, LDX, IFRQ, IWT, IND1,
IND2, TOL, IPRINT, XX, LDXX, CORR,LDCORR, COEF1, LDCOF1, COEF2,
LDCOF2, COEFR1, LDCFR1, COEFR2, LDCFR2, STAT, LDSTAT, R, S, IND,
WORK, WKA, WK)
```

   The additional arguments are as follows:

   R — Work vector of length $\texttt{NVAR1}^2$.

   S — Work vector of length $\texttt{NVAR2}^2$.

   IND — Work vector of length NVAR1 + NVAR2 + 2.

   WORK — Work vector of length max(NOBS, 2 * (NVAR1 + NVAR2))

   WKA — Work vector of length $(\max(\texttt{NVAR1}, \texttt{NVAR2}))^2$.

   WK — Work vector of length 3 * max(NVAR1, NVAR2) − 1.

2. Informational errors

Type	Code	
3	1	The standardized cross covariance matrix is not of full rank or is very ill-conditioned. Small canonical correlations may not be accurate.
3	2	One or more variables is linearly dependent upon the proceeding variables in its group.
4	3	The sum of the frequencies is equal to zero. The sum of the frequencies must be positive.
4	4	The sum of the weights is equal to zero. The sum of the weights must be positive.

Description

Routine CANCR computes the canonical correlations, the canonical coefficients, the canonical scores, Wilks' lambda for testing the independence of two sets of variates, and a series of Bartlett's tests of the hypothesis that the *k*-th largest and all larger canonical correlations are simultaneously zero. A matrix of observations is used in these computations.

Let x_{ij} denote the *j*-th variable on the *i*-th observation, w_i denote the observation weight, f_i denote the observation frequency, Γ_{11} denote the upper triangular Cholesky ($R^T R$) factorization of the sample covariance matrix of the group 1 variables, Γ_{22} denote the upper triangular Cholesky ($R^T R$) factorization of the group 2 variables sample covariance matrix, and

$$\Gamma_{12} = \left(\Gamma_{11}\right)^{-1} \hat{\Sigma}_{12} \left(\Gamma_{22}\right)^{-T}$$

where

$$\hat{\Sigma}_{12}$$

is the sample estimate of the matrix of covariances between the group 1 and the group 2 variables. Then, the computational procedure in obtaining the canonical correlations is as follows:

1. The weighted mean of each variable is computed via the standard formula (see UVSTA, see Chapter 1, Basic Statistics). The means are then subtracted from the observations.

2. Each element in the *i*-th row of X is multiplied by

$$\sqrt{(w_i f_i)}$$

3. Gram-Schmidt orthogonalization is used on X to obtain Y_1 and Y_2, where Y_1 and Y_2 are the results of the Gram-Schmidt orthogonalization of the group 1 and the group 2 variables, respectively. The matrices Γ_{11} and Γ_{22} are obtained as a by-product of the orthogonalization. Compute

$$\Gamma_{12} = Y_1^T Y_2$$

4. The canonical correlations are obtained as the singular values of the matrix Γ_{12}. Denote the left and right orthogonal matrices obtained as a by-product of this decomposition by *L* and *R*, respectively.

5. The canonical coefficients are obtained from *L* and *R* by multiplying *L* and *R* by the inverses of Γ_{11} and Γ_{22}, respectively (see Golub 1969).

6. The correlations of the original variables with the canonical variables are obtained by multiplying *L* and *R* by Γ_{11} and Γ_{22}, respectively.

7. The canonical scores are obtained by multiplying the matrices Y_1 and Y_2 by the matrices *L* and *R*, respectively, and then dividing each row of Y_1 and Y_2 by

$$\sqrt{(w_i f_i)}$$

8. Wilks' lambda, the Bartlett's tests, Rao's *F* corresponding to these tests, the numerator and denominator degrees of freedom of *F*, and the significance level of *F* are computed as in Rao (1973, page 556). Bartlett's tests are computed as

$$\Lambda_i = \prod_{j=1}^{q} (1 - \rho_j^2)$$

where q = NVAR2 is the number of canonical correlations, the canonical correlations are ordered from largest to smallest, and ρ_j denotes the *j*-th largest canonical correlation. Wilks' lambda is given as Λ_1. The degrees of freedom in the numerator of the corresponding Rao's *F* statistic is given as

$$d_1 = pu$$

where $p = v_1 - i + 1$, $u = v_2 - i + 1$, v_1 = NVAR2, and v_2 = NVAR1. Let

$$m = t - \frac{p + u + 1}{2}$$

where t is the degrees of freedom in COV($\sum_i f_i - 1$), and let

$$s = \sqrt{\frac{p^2u^2 - 4}{p^2 + u^2 - 5}}$$

if $p^2 + u^2 - 5 \neq 0$, and let $s = 2$ otherwise. Then, Rao's F corresponding to Bartlett's test is computed as

$$F_i = \frac{1 - \Lambda_i^{\frac{1}{s}}}{\Lambda_i^{\frac{1}{s}}}(ms - pu/2 + 1)/pu$$

Rao's F has numerator degrees of freedom $d_2 = ms - pu/2 + 1$. The significance level of F is obtained from the standard F distribution.

CANVC

Performs canonical correlation analysis from a variance-covariance matrix or a correlation matrix.

Required Arguments

NDF — Number of degrees of freedom in the covariance or correlation matrix. (Input)
If NDF is unknown, an estimate of NDF = 100 is suggested in which case the last four columns of CORR are meaningless.

COV — NVAR1 + NVAR2 by NVAR1 + NVAR2 matrix containing the covariance or correlation matrix. (Input)
Routines COVPL, RBCOV, or CORVC (see Chapter 3, Correlation) may be used to calculate COV from a data matrix. COV must be nonnegative definite within a tolerance of 100.0 * AMACH(4). Only the upper triangle of COV is referenced.

IND1 — Vector of length NVAR1 containing the column and row numbers in COVfor the group 1 variables. (Input)

IND2 — Vector of length NVAR2 containing the column and row numbers in COV for the group 2 variables. (Input)

CORR — NV by 6 matrix containing the output statistics. (Output)
NV is the minimum of NVAR1 and NVAR2.

Col.	Statistic
1	Canonical correlations sorted from the largest to the smallest.
2	Wilks' lambda for testing that the current and all smaller canonical correlations are zero.
3	Rao's *F* corresponding to Wilks' lambda. If the canonical correlation is greater than 0.99999, *F* is set to 9999.99.
4	Numerator degrees of freedom for the *F*.
5	Denominator degrees of freedom for the *F*.
6	Probability of a larger *F* statistic.

If an *F* statistic is negative, then `CORR`(*i*, 6) is set to one. If either `CORR`(*i*, 4) or `CORR`(*i*, 5) is not positive, then `CORR`(*i*, 6) is set to the missing value code (NaN).

COEF1 — `NVAR1` by `NVAR1` matrix containing the group 1 canonical coefficients. (Output)
The columns of `COEF1` contain the vectors of canonical coefficients for group 1.

COEF2 — `NVAR2` by `NVAR2` matrix containing the group 2 canonical coefficients. (Output)
The columns of `COEF2` contain the vectors of canonical coefficients for group 2.

COEFR1 — `NVAR1` by `NV` matrix containing the correlations between the group 1 variables and the group 1 canonical scores. (Output)
`NV` is the minimum of `NVAR1` and `NVAR2`.

COEFR2 — `NVAR2` by `NV` matrix containing the correlations between the group 2 variables and the group 2 canonical scores. (Output)
`NV` is the minimum of `NVAR1` and `NVAR2`.

Optional Arguments

NVAR1 — Number of variables in group 1. (Input)
Default: `NVAR1` = size (`IND1`,1).

NVAR2 — Number of variables in group 2. (Input)
Default: `NVAR2` = size (`IND2`,1).

LDCOV — Leading dimension of `COV` exactly as specified in the dimension statement in the calling program. (Input)
Default: `LDCOV` = size (`COV`,1).

IPRINT — Printing option. (Input)
Default: `IPRINT` = 0.

IPRINT	Action
0	No printing.
1	Printing of CORR, COEF1, COEF2, COEFR1, and COEFR2 is performed.

LDCORR — Leading dimension of CORR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCORR = size (CORR,1).

LDCOF1 — Leading dimension of COEF1 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOF1 = size (COEF1,1).

LDCOF2 — Leading dimension of COEF2 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOF2 = size (COEF2,1).

LDCFR1 — Leading dimension of COEFR1 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCFR1 = size (COEFR1,1).

LDCFR2 — Leading dimension of COEFR2 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCFR2 = size (COEFR2,1).

FORTRAN 90 Interface

Generic: `CALL CANVC (NDF, COV, IND1, IND2, CORR, COEF1, COEF2, COEFR1, COEFR2 [,…])`

Specific: The specific interface names are S_CANVC and D_CANVC.

FORTRAN 77 Interface

Single:
```
CALL CANVC (NDF, NVAR1, NVAR2, COV, LDCOV, IND1,
IND2,IPRINT, CORR, LDCORR, COEF1, LDCOF1, COEF2, LDCOF2,
COEFR1, LDCFR1, COEFR2, LDCFR2)
```

Double: The double precision name is DCANVC.

Example

The following example is taken from Van de Geer (1971). There are six group 1 variables and two group 2 variables. The maximum correlation turns out to be 0.609.

```
USE CANVC_INT
INTEGER    IPRINT, LDCFR1, LDCFR2, LDCOF1, LDCOF2, LDCORR, &
           LDCOV, NDF, NV, NVAR1, NVAR2
```

```
      PARAMETER  (IPRINT=1, LDCFR1=6, LDCFR2=2, LDCOF1=6, LDCOF2=2, &
                 LDCORR=2, LDCOV=8, NDF=100, NV=2, NVAR1=6, NVAR2=2)
!
      INTEGER    IND1(NVAR1), IND2(NVAR2)
      REAL       COEF1(NVAR1,NVAR1), COEF2(NVAR2,NVAR2), &
                 COEFR1(NVAR1,NVAR2), COEFR2(NVAR2,NVAR2), &
                 CORR(NVAR2,NVAR1), COV(LDCOV,NVAR1+NVAR2)
!
      DATA COV/1.0000, 0.1839, 0.0489, 0.0186, 0.0782, 0.1147, 0.2137, &
          0.2742, 0.1839, 1.0000, 0.2220, 0.1861, 0.3355, 0.1021, &
          0.4105, 0.4043, 0.0489, 0.2220, 1.0000, 0.2707, 0.2302, &
          0.0931, 0.3240, 0.4047, 0.0186, 0.1861, 0.2707, 1.0000, &
          0.2950, -0.0438, 0.2930, 0.2407, 0.0782, 0.3355, 0.2302, &
          0.2950, 1.0000, 0.2087, 0.2995, 0.2863, 0.1147, 0.1021, &
          0.0931, -0.0438, 0.2087, 1.0000, 0.0760, 0.0702, 0.2137, &
          0.4105, 0.3240, 0.2930, 0.2995, 0.0760, 1.0000, 0.6247, &
          0.2742, 0.4043, 0.4047, 0.2407, 0.2863, 0.0702, 0.6247, &
          1.0000/
!
      DATA IND1/1, 2, 3, 4, 5, 6/, IND2/7, 8/
!
      CALL CANVC (NDF, COV, IND1, IND2, CORR, &
                 COEF1, COEF2, COEFR1, COEFR2, IPRINT=IPRINT)
!
!
      END
```

Output

```
               *** Canonical Correlations Statistics ***
     Canonical                                                 Prob. of
   Correlations  Wilks Lambda      Raos F  Num. df  Denom. df  Larger F
1        0.6093        0.6159       4.250       12        186    0.0000
2        0.1431        0.9795       0.393        5         94    0.8524

        Group One Canonical Coefficients
         1       2       3       4       5       6
1   0.326   0.411  -0.799   0.358  -0.032   0.053
2   0.481  -0.340  -0.083  -0.766  -0.484  -0.139
3   0.456   0.718   0.625   0.134  -0.056   0.038
4   0.202  -0.689   0.060   0.732  -0.335   0.080
5   0.184  -0.125  -0.064  -0.045   1.079  -0.225
6  -0.027  -0.174   0.054  -0.086  -0.021   1.017

Group Two Canonical Coefficients
         1       2
1   0.464   1.194
2   0.642  -1.108

Correlations Between the Group One Variables
     and the Group One Canonical Scores
                     1       2
             1   0.4517   0.3408
             2   0.7388  -0.2932
             3   0.6733   0.4313
             4   0.4769  -0.5799
```

```
             5   0.5299  -0.2811
             6   0.1319  -0.0903

Correlations Between the Group Two Variables
     and the Group Two Canonical Scores
                      1       2
             1   0.8653   0.5013
             2   0.9320  -0.3625
```

Comments

1. Workspace may be explicitly provided, if desired, by use of C2NVC/DC2NVC. The reference is:

   ```
   CALL C2NVC (NDF, NVAR1, NVAR2, COV, LDCOV, IND1, IND2, IPRINT,
   CORR, LDCORR, COEF1, LDCOF1, COEF2, LDCOF2, COEFR1, LDCFR1,
   COEFR2, LDCFR2, R, S, STD1, STD2, WKA, WK)
   ```

   The additional arguments are as follows:

   R — Work vector of length NVAR1^2.

   S — Work vector of length NVAR2^2.

   STD1 — Work vector of length NVAR1.

   STD2 — Work vector of length NVAR2.

   WKA — Work vector of length (NVAR1 + NVAR2)^2.

   WK — Work vector of length 3 * max(NVAR1, NVAR2).

2. Informational errors

Type	Code	
3	1	The standardized cross covariance matrix is not of full rank or is very ill-conditioned. Small canonical correlations may not be accurate.
4	2	COV is not nonnegative definite.

Description

Routine CANVC computes the canonical correlations, the canonical coefficients, Wilks' lambda (for testing the independence of two sets of variates), and a series of tests due to Bartlett for testing that all canonical correlations greater than or equal to the *k*-th largest are simultaneously zero. The covariance matrix is used in these computations.

The group 1 variables covariance matrix is first extracted from COV and placed in the matrix S_{11}. Similarly, the group 2 variables covariance matrix is placed in S_{22}. The "standardized" cross covariance matrix is then computed as:

$$C = \left(S_{11}^{-\frac{1}{2}}\right)^T S_{12} \left(S_{22}^{-\frac{1}{2}}\right)$$

where S_{12} is the NVAR1 × NVAR2 matrix of covariances between the group 1 and group 2 variables, and $S^{1/2}$denotes the upper triangular Cholesky ($R^T R$) factorization of S. In the computation of C and in the following, it is assumed that NVAR1 is greater than NVAR2. The group 1 and group 2 variables should be interchanged in the following if this is not the case.

The canonical correlations are computed as the singular values of the matrix C. The canonical coefficients are obtained from the left and right orthogonal matrices resulting from the singular value decomposition of C. In particular, for Γ_1 = COEF1.

$$\Gamma_1 = \left(S_{11}^{-\frac{1}{2}}\right) L$$

where L is the left orthogonal matrix from the singular value decomposition.

Similarly, the correlations between the original variables and the canonical variables, R_1 = COEFR1, are obtained for the group 1 variables as:

$$R_1 = \Delta_{11}^{-\frac{1}{2}} \left(S_{11}^{\frac{1}{2}}\right)^T L$$

where Δ_{11} is a diagonal matrix containing the diagonal of S_{11} along its diagonal.

Wilks' lambda, the Bartlett's tests, Rao's F corresponding to these tests, the numerator and denominator degrees of freedom of F , and the significance level of F are computed as in Rao (1973, page 556). Bartlett's tests are computed as

$$\Lambda_i = \prod_{j=i}^{q} \left(1 - \rho_j^2\right)$$

where q = NVAR2 is the number of canonical correlations, the canonical correlations are ordered from largest to smallest, and ρ_j denotes the j-th largest canonical correlation. Wilks' lambda is given as Λ_1. The degrees of freedom in the numerator of the corresponding Rao's F statistic is given as

$$d_1 = pu$$

where $p = v_1 - i + 1$, $u = v_2 - i + 1$, v_1 = NVAR2, and v_2 = NVAR1. Let

$$m = t - \frac{p + u + 1}{2}$$

where t is the degrees of freedom in COV, and let

$$s = \sqrt{\frac{p^2 u^2 - 4}{p^2 + u^2 - 5}}$$

if $p^2 + u^2 - 5 \neq 0$, and let $s = 2$ otherwise. Then, Rao's F corresponding to Bartlett's test is computed as

$$F_i = \frac{1 - \Lambda_i^{\frac{1}{s}}}{\Lambda_i^{\frac{1}{s}}} \frac{ms - pu/2 + 1}{pu}$$

Rao's F has numerator degrees of freedom $d_2 = ms - pu/2 + 1$. The significance level of F is obtained from the standard F distribution.

Chapter 10: Discriminant Analysis

Routines

Usage Notes

The routine DSCRM (page 979) allows linear or quadratic discrimination and the use of either reclassification, split sample, or the leaving-out-one methods in order to evaluate the rule. Moreover, DSCRM can be executed in an online mode, that is, one or more observations can be added to the rule during each invocation of DSCRM.

The mean vectors for each group of observations and an estimate of the common covariance matrix for all groups are input to DMSCR (page 993). These estimates can be computed via routine DSCRM. Output from DMSCR are linear combinations of the observations, which at most separate the groups. These linear combinations may subsequently be used for discriminating between the groups. Their use in graphically displaying differences between the groups is possibly more important, however.

Nearest neighbor discrimination is performed in routine NNBRD (page 997). In this routine, the user can set the number of nearest neighbors to be used in the discrimination and the threshold for classification. Split samples can also be used.

DSCRM

Performs a linear or a quadratic discriminant function analysis among several known groups.

Required Arguments

NROW — The absolute value of NROW is the number of rows of X that contain an observation. (Input)

If NROW is negative, the observations are deleted from the discriminant statistics. If NROW is positive, they are added.

NVAR — Number of variables to be used in the discrimination. (Input)

X — |NROW| by NVAR + *m* matrix containing the data to be used on this call. (Input, if |NROW| > 0; X is not referenced otherwise)
m is 1, 2, or 3 depending upon whether any columns in X contain frequencies or weights. One column in X must contain the group number for each observation. Group numbers must be 1.0, 2, 0, …, NGROUP. If present, IFRQ gives the column containing the frequencies, while IWT gives the column in X containing the weights.

NGROUP — Number of groups in the data. (Input)

COV — NVAR by NVAR by *g* matrix of covariances. (Output, for IDO = 0 or 1; input/output, for IDO = 2, 3, or 5; input, for IDO = 4; not referenced if IDO = 6)
g = NGROUP + 1 when IMTH = 1, 2, 4, or 5, and *g* = 1 otherwise. When IMTH = 3 or 6, the within-group covariance matrices are not computed. Regardless of the value of IMTH, the pooled covariance matrix is always computed and saved as the *g*-th covariance matrix in COV.

COEF — NGROUP by NVAR + 1 matrix containing the linear discriminant function coefficients. (Output, if IDO = 0 or 3; input, if IDO = 4; not referenced if IDO = 1, 2, 5, or 6)
The first column of COEF contains the constant term, and the remaining columns contain the variable coefficients. Row *i* of COEF corresponds to group *i*. COEF is always computed as the linear discriminant function coefficients even when quadratic discrimination is specified.

ICLASS — Vector of length |NROW| containing the group to which the observation was classified. (Output, if IDO = 0 or 4; not referenced otherwise)
If an observation has an invalid group number, frequency, or weight when the leaving-out-one method has been specified, then the observation is not classified and the corresponding elements of ICLASS and PROB are set to zero.

PROB — |NROW| by NGROUP matrix containing the posterior probabilities for each observation. (Output, if IDO = 0 or 4; not referenced otherwise)

CLASS — NGROUP by NGROUP matrix containing the classification table. (Output, if IDO = 0 or 1, input/output, if IDO = 4; not referenced otherwise)
Each observation that is classified and has a group number equal to 1.0, 2.0, …, NGROUP is entered into the table. The rows of the table correspond to the known group membership. The columns refer to the group to which the observation was classified. Classification results accumulate with each call to DSCRM with IDO = 4. For example, if 2 calls with IDO = 4 are made, then the elements in CLASS sum to the total number of valid observations in the 2 calls.

D2 — NGROUP by NGROUP matrix containing the Mahalanobis distances

$$D_{ij}^2$$

between the group means. (Output, when IDO= 0 or 3; not referenced otherwise)
For linear discrimination, the Mahalanobis distance is computed using the pooled covariance matrix. Otherwise, the Mahalanobis distance

$$D_{ij}^2$$

between group means *i* and *j* is computed using the within covariance matrix for group *i* in place of the pooled covariance matrix.

STAT — Vector of length 4 + 2 * (NGROUP + 1) containing statistics of interest. (Input/Output, if IDO = 3 or 5; output, if IDO = 0 or 1; not referenced otherwise)
The first element of STAT is the sum of the degrees of freedom for the within-covariance matrices. The second, third and fourth elements of STAT correspond to the chi-squared statistic, its degrees of freedom, and the probability of a greater chi-squared, respectively, of a test of the homogeneity of the within-covariance matrices (not computed if IMTH = 3 or 6). The 5-th through 5 + NGROUP elements of STAT contain the log of the determinants of each group's covariance matrix (not computed if IMTH = 3 or 6) and of the pooled covariance matrix (element 5 + NGROUP). Finally, the last NGROUP + 1 elements of STAT contain the sum of the weights within each group and, in the last position, the sum of the weights in all groups.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of DSCRM; all the data are input at once.
1	This is the first invocation of DSCRM with this data, additional calls will be made. Initialization and updating for the NROW observations are performed.
2	This is an intermediate invocation of DSCRM; updating for the NROW observations is performed.
3	All statistics are updated for the NROW observations. The discriminant functions and other statistics are computed.
4	The discriminant functions are used to classify each of the NROW observations in X.

5 The covariance matrices are computed, and workspace is released. No further calls to DSCRM with IDO greater than 1 should be made without first calling DSCRM with IDO = 1.

6 Workspace is released. No further calls to DSCRM with IDO greater than 1 should be made without first calling DSCRM with IDO = 1. This option is not required if a call has been made with IDO = 5 or if workspace is explicitly provided by use of D2CRM.

See Comments 5 and 6 for further information.

NCOL — Number of columns in matrix X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IND — Vector of length NVAR containing the column numbers in X to be used in the discrimination. (Input)
By default, IND(I)=I.

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. Positive IFRQ indicates that column number IFRQ of X contains the frequencies. All frequencies should be integer values. If this is not the case, the NINT (nearest integer) function is used to obtain integer frequencies.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. Positive IWT means that column IWT of X contains the weights. Negative weights are not allowed.
Default: IWT = 0.

IGRP — Column number in X containing the group numbers. (Input)
The group numbers must be 1.0, 2,0, …, NGROUP for an observation to be used in the discriminant functions. An observation will be classified regardless of its group number when the reclassification method is specified.
Default: IGRP = NVAR + 1.

IMTH — Option parameter giving the method of discrimination. (Input)
IMTH determines whether linear or quadratic discrimination is used whether the group covariance matrices are computed (the pooled covariance matrix is always computed) and whether the leaving-out-one or the reclassification method is used to classify each observation.
Default: IMTH = 1.

IMTH	Discrim.	Covariance	Classification

1	Linear	All	Reclassification
2	Quadratic	All	Reclassification
3	Linear	Pooled only	Reclassification
4	Linear	All	Leaving-out-one
5	Quadratic	All	Leaving-out-one
6	Linear	Pooled only	Leaving-out-one

In the leaving-out-one method of classification, the posterior probabilities are adjusted so as to eliminate the effect of the observation from the sample statistics prior to its classification. In the reclassification method, the effect of the observation is not eliminated from the classification function. Calls to DSCRM with IMTH = 1, 2, 4, or 5 can be intermixed, as can calls to DSCRM with IMTH = 3 or 6. Calls to DSCRM with IMTH = 1, 2, 4, or 5 cannot be intermixed with calls to DSCRM with IMTH = 3 or 6 without first calling DSCRM with IDO = 1 (or 0).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.
For the given combination of IDO and IPRINT, the following arrays are printed.

IPRINT	IDO	Printing
0	Any	None
1 or 2	0	PRIOR, NI, XMEAN, COV, COEF, ICLASS, PROB, CLASS, D2, STAT, NRMISS
1 or 2	1 or 2	None
1 or 2	3	PRIOR, NI, XMEAN, COEF, D2, STAT, NRMISS
1	4	None
2	4	ICLASS, PROB
1 or 2	5	COV, CLASS
1 or 2	6	None

Note that the only change from IPRINT = 1 to IPRINT = 2 is the printing when IDO = 4. Also, note that PRIOR is printed even though it may be input only.

PRIOR — Vector of length NGROUP containing the prior probabilities for each group. (Input, if PRIOR(1) is not −1.0 and IDO is 0 or 3; input/output, if PRIOR(1) is −1.0 and IDO is 0 or 3; input, if IDO is 4; not referenced if IDO is 1, 2, 5, or 6)
If PRIOR(1) is not −1.0, then the elements of PRIOR should sum to 1.0. Proportional priors can be selected by setting PRIOR(1) = −1.0. In this case, the prior probabilities will be proportional to the sample size in each group, and the elements of PRIOR will contain the proportional prior probabilities after the first call with IDO = 0 or 3.
Default: PRIOR(1) = −1.0.

NI — Vector of length NGROUP. (Input, for IDO = 3, 4, or 5; input/output, for IDO = 2; output, for IDO = 0 or 1; not referenced if IDO = 6)
The *i*-th element of NI contains the number of observations in group *i*.

XMEAN — NGROUP by NVAR matrix. (Input, for IDO = 3, 4, or 5; input/output, for IDO = 2; output, for IDO = 0 or 1; not referenced if IDO = 6)
The *i*-th row of XMEAN contains the group *i* variable means.

LDXMEA — Leading dimension of XMEAN exactly as specified in the dimension statement in the calling program. (Input)
Default: LDXMEA = size(XMEAN,1).

LDCOV — Leading and second dimensions of COV exactly as specified in the dimension statement of the calling program. (Input)
The first two dimensions of COV must be equal.
Default: LDCOV = size (COV,1).

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOEF = size (COEF,1).

LDPROB — Leading dimension of PROB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDPROB = size (PROB,1).

LDCLAS — Leading dimension of CLASS exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCLAS = size (CLASS,1).

LDD2 — Leading dimension of D2 exactly as specified in the dimension statement of the calling program. (Input)
Default: LDD2 = size (D2,1).

NRMISS — Number of rows of data encountered in calls to DSCRM containing missing values (NaN) for the classification, group, weight, and/or frequency variables. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3, not referenced otherwise)
If a row of data contains a missing value (NaN) for any of these variables, that row is excluded from the computations.

FORTRAN 90 Interface

Generic: `CALL DSCRM (NROW, NVAR, X, NGROUP, COV, COEF, ICLASS, PROB, CLASS, D2, STAT [,…])`

Specific: The specific interface names are S_DSCRM and D_DSCRM.

FORTRAN 77 Interface

Single: `CALL DSCRM (IDO, NROW, NVAR, NCOL, X, LDX, IND, IFRQ, IWT, IGRP, NGROUP, IMTH, IPRINT, PRIOR, NI, XMEAN, LDXMEA, COV, LDCOV, COEF, LDCOEF, ICLASS, PROB, LDPROB, CLASS, LDCLAS, D2, LDD2, STAT, NRMISS)`

Double: The double precision name is `DDSCRM`.

Example 1

The following example uses linear discrimination with equal prior probabilities on Fisher's (1936) iris data. This example illustrates the execution of `DSCRM` when one call is made.

```
      USE GDATA_INT
      USE DSCRM_INT
      INTEGER   IGRP, IMTH, IPRINT, LDCLAS, LDCOEF, LDCOV, LDD2, &
                LDPROB, LDX, LDXMEA, NCOL, NGROUP, NROW, NVAR
      PARAMETER  (IGRP=1, IMTH=3, IPRINT=1, LDCOV=4, NCOL=5, &
                  NGROUP=3, NROW=150, NVAR=4, LDCLAS=NGROUP, &
                  LDCOEF=NGROUP, LDD2=NGROUP, LDPROB=NROW, &
                  LDX=NROW, LDXMEA=NGROUP)
!
      INTEGER    ICLASS(NROW), IND(4), NI(NGROUP), NOBS, NRMISS, NV
      REAL       CLASS(LDCLAS,NGROUP), COEF(LDCOEF,NVAR+1), &
                 COV(LDCOV,LDCOV,1), D2(LDD2,NGROUP), PRIOR(3), &
                 PROB(LDPROB,NGROUP), STAT(6+2*NGROUP), X(LDX,5), &
                 XMEAN(LDXMEA,NVAR)
!
      DATA IND/2, 3, 4, 5/, PRIOR/0.3333333, 0.3333333, 0.3333333/
!
      CALL GDATA (3, X, NOBS, NV)
!
      CALL DSCRM (NROW, NVAR, X, NGROUP, COV, COEF, ICLASS, PROB, &
                  CLASS, D2, STAT, IND=IND, IGRP=IGRP, IMTH=IMTH, &
                  IPRINT=IPRINT, PRIOR=PRIOR, NI=NI, XMEAN=XMEAN, &
                  NRMISS=NRMISS)
!
      END
```

Output

```
PRIOR, the prior probabilities
     1        2        3
0.3333   0.3333   0.3333

NI, the number in each group
 1    2    3
50   50   50

     XMEAN, the group means
        1       2       3       4
1   5.006   3.428   1.462   0.246
2   5.936   2.770   4.260   1.326
3   6.588   2.974   5.552   2.026
```

```
The pooled within-groups covariance matrix
          1         2         3         4
1    0.2650    0.0927    0.1675    0.0384
2    0.0927    0.1154    0.0552    0.0327
3    0.1675    0.0552    0.1852    0.0427
4    0.0384    0.0327    0.0427    0.0419

COEF, the discriminant function coefficients
         1       2       3       4       5
1    -86.3    23.5    23.6   -16.4   -17.4
2    -72.9    15.7     7.1     5.2     6.4
3   -104.4    12.4     3.7    12.8    21.1

ICLASS, the classifications

 Obs.  Class
  1     1
  2     1
  3     1
  4     1
  5     1
  6     1


     .
     .
     .
145     3
146     3
147     3
148     3
149     3
150     3

PROB, the posterior probabilities
          1       2       3
  1   1.000   0.000   0.000
  2   1.000   0.000   0.000
  3   1.000   0.000   0.000
  4   1.000   0.000   0.000
  5   1.000   0.000   0.000
  6   1.000   0.000   0.000
          .
          .
          .
145   0.000   0.000   1.000
146   0.000   0.000   1.000
147   0.000   0.006   0.994
148   0.000   0.003   0.997
149   0.000   0.000   1.000
150   0.000   0.018   0.982

CLASS, the classification table
          1       2       3
```

```
  1   50.00    0.00    0.00
  2    0.00   48.00    2.00
  3    0.00    1.00   49.00

D2, the distances between group means
          1       2       3
  1     0.0    89.9   179.4
  2    89.9     0.0    17.2
  3   179.4    17.2     0.0

                                        STAT
    1      2      3      4      5      6      7      8      9      10
147.0    NaN    NaN    NaN    NaN    NaN    NaN  -10.0   50.0   50.0

11      12
50.0   150.0

NRMISS, number of missing observations =   0
```

Comments

1. Workspace may be explicitly provided, if desired, by use of D2CRM/DD2CRM. The reference is:

   ```
   CALL D2CRM (IDO, NROW, NVAR, NCOL, X, LDX, IND, IFRQ, IWT, IGRP,
   NGROUP, IMTH, IPRINT, RIOR, NI, XMEAN, LDXMEA, COV, LDCOV, COEF,
   LDCOEF, ICLASS, PROB, LDPROB, CLASS, LDCLAS, D2, LDD2, STAT,
   NRMISS, D, OB, OB1)
   ```

   The additional arguments are as follows:

   D — Work vector of length equal to (NGROUP + 1) * NVAR if IMTH is not 3 or 6, and of length NVAR otherwise.

   OB — Work vector of length equal to NVAR.

   OB1 — Work vector of length equal to NVAR.

2. Informational errors

Type	Code	
3	1	A row of the data matrix X has an invalid group number.
4	2	The variance-covariance matrix for a group is singular.
4	3	The pooled variance-covariance matrix is singular.
3	4	The variance-covariance matrix for a group is singular. STAT(2) cannot be computed. STAT(2) and STAT(4) are set to the missing value code (NaN).
3	5	An element of PRIOR is less than or equal to 10^{-20}.
3	6	The leaving-out-one method is specified, but this observation does not have a valid weight, or it does not have a valid frequency. This observation is ignored.

3	7	The leaving-out-one method is specified, but this observation does not have a valid group number. This observation is ignored.

3. Common choices for the Bayesian prior probabilities are given by:

   PRIOR(i) = 1.0/NGROUP (equal prior probabilities)
   PRIOR(i) = NI(i)/NOBS (proportional prior probabilities)
   PRIOR(i) = Past history or subjective judgement
   In all cases, the prior probabilities should sum to 1.0.

4. Two passes of the data are made. In the first pass, the statistics required to compute the discriminant functions are obtained (IDO = 1, 2, and 3). In the second pass, the discriminant functions are used to classify the observations. When IDO = 0, all of the data are memory resident, and both passes are made in one call to DSCRM. When IDO > 0 and workspace is not explicitly provided by use of D2CRM, a third call to DSCRM involving no data is required with IDO = 5 or 6.

5. Here are a few rules and guidelines for the correct value of IDO in a series of calls.

   (1) Calls with IDO = 0 or 1 may be made at any time. These calls destroy all statistics from previous calls.
   (2) IDO may not be 2, 3, 4, 5, or 6
   (a) immediately after a call where IDO was 0,
   (b) before a call with IDO = 1 has been made, or
   (c) immediately after a call with IDO = 5 or 6 has been made.
   (3) IDO may not be 4 or 5 before a call with IDO = 3 has been made.
   (4) Each series of calls to DSCRM which begins with IDO = 1 should end with IDO = 5 or 6 to ensure the proper release of workspace.
   This is a valid sequence of IDOs:
   0, 1, 2, 3, 4, 5, 1, 3, 4, 3, 5, 1, 6, 1, 2, 0, 0, 1, 3, 5.

6. Unlike many routines using the parameter IDO, because of the workspace allocation and saved variables, neither DSCRM or D2CRM can be called with IDO greater than 1 in consecutive invocations with more than one dataset.

Description

Routine DSCRM performs discriminant function analysis using either linear or quadratic discrimination. The output from DSCRM includes a measure of distance between the groups, a table summarizing the classification results, a matrix containing the posterior probabilities of group membership for each observation, and the within-sample means and covariance matrices. The linear discriminant function coefficients are also computed.

All observations can be input during one call to DSCRM, a method of operation that has the advantage of simplicity. Alternatively, one or more rows of observations can be input during separate calls. This method does not require that all observations be memory resident, a significant advantage with large data sets. Note, however, that DSCRM requires two passes of the data. During the first pass the discriminant functions are computed while in the second pass, the observations are classified. Thus, with the second method of operation, the data will usually need to be input into DSCRM twice.

Because both methods result in the same operations being performed, the algorithm for DSCRM is discussed as if only a few observations are input during each call. The operations performed during each call to DSCRM depend upon the IDO parameter. IDO = 0 should be used if all observations are to be input at one time.

The IDO = 1 step is the initialization step. The variables XMEAN, CLASS, and COV are initialized to zero, and other program parameters are set. After this call, all subroutine arguments except IDO, NROW, X, LDX and IMTH should not be changed by the user except via another call to DSCRM with IDO = 0 or IDO = 1. IMTH can be changed from one call to the next within the two sets {1, 2, 4, 5} or {3, 6} but not between these sets when IDO > 1. That is, do not call DSCRM with IMTH = 1 in one call and IMTH = 3 in another call without first calling DSCRM with IDO = 1.

After initialization has been performed in the IDO = 1 step, the within-group means are updated for all valid observations in X. Observations with invalid group numbers are ignored, as are observations with missing values. The *LU* factorization of the covariance matrices are updated by adding (or deleting) observations via Givens rotations.

The IDO = 2 step is used solely for adding or deleting observations from the model as in the above paragraph.

The IDO = 3 step begins by adding all observations in X to the means and the factorizations of the covariance matrices. It continues by computing some statistics of interest: the linear discriminant functions, the prior probabilities (if PRIOR(1) = –1.0), the log of the determinant of each of the covariance matrices, a test statistic for testing that all of the within-group covariance matrices are equal, and a matrix of Mahalanobis distances between the groups. The matrix of Mahalanobis distances is computed via the pooled covariance matrix when linear discrimination is specified, the row covariance matrix is used when the discrimination is quadratic.

Covariance matrices are defined as follows. Let N_i denote the sum of the frequencies of the observations in group i, and let M_i denote the number of observations in group i. Then, if S_i denotes the within-group i covariance matrix,

$$S_i = \frac{1}{N_i - 1} \sum_{j=1}^{M_i} w_j f_j \left(x_j - \bar{x}\right)\left(x_j - \bar{x}\right)^T$$

where w_j is the weight of the j-th observation in group i, f_j is its frequency, x_j is the j-th observation column vector (in group i), and $\bar{x}$ denotes the mean vector of the observations in group i. The mean vectors are computed as

$$\bar{x} = \frac{1}{W_i} \sum_{j=1}^{M_i} w_j f_j x_j$$

where

$$W_i = \sum_{j=1}^{M_i} w_j f_j$$

Given the means and the covariance matrices, the linear discriminant function for group i is computed as:

$$z_i = \ln\left(p_i\right) - 0.5 \bar{x}_i^T S_p^{-1} \bar{x}_i + x^T S_p^{-1} \bar{x}_i$$

where $\ln(p_i)$ is the natural log of the prior probability for the i-th group, x is the observation to be classified, and S_p denotes the pooled covariance matrix.

Let S denote either the pooled covariance matrix or one of the within-group covariance matrices S_i. (S will be the pooled covariance matrix in linear discrimination, and S_i otherwise.) The Mahalanobis distance between group i and group j is computed as:

$$D_{ij}^2 = \left(\bar{x}_i - \bar{x}_j\right)^T S^{-1}\left(\bar{x}_i - \bar{x}_j\right)$$

Finally, the asymptotic chi-squared test for the equality of covariance matrices is computed as follows (Morrison 1976, page 252):

$$\gamma = C^{-1}\sum_{i=1}^{k} n_i \left\{\ln\left(\left|S_p\right|\right) - \ln\left(\left|S_i\right|\right)\right\}$$

where n_i is the number of degrees of freedom in the i-th sample covariance matrix, k is the number of groups, and

$$C^{-1} = 1 - \frac{2p^2 + 3p - 1}{6(p+1)(k-1)}\left(\sum_{i=1}^{k}\frac{1}{n_i} - \frac{1}{\Sigma_j n_j}\right)$$

where p is the number of variables.

When IDO = 4, the estimated posterior probability of each observation x belonging to group i is computed using the prior probabilities and the sample mean vectors and estimated covariance matrices under a multivariate normal assumption. Under quadratic discrimination, the within-group covariance matrices are used to compute the estimated posterior probabilities. The estimated posterior probability of an observation x belonging to group i is

$$\hat{q}_i(x) = \frac{e^{-\frac{1}{2}D_i^2(x)}}{\sum_{j=1}^{k} e^{-\frac{1}{2}D_j^2(x)}}$$

where

$$D_i^2(x) = \begin{cases} \left(x - \bar{x}_i\right)^T S_i^{-1}\left(x - \bar{x}_i\right) + \ln\left|S_i\right| - 2\ln\left(p_i\right) & \text{IMTH} = 1 \text{ or } 2 \\ \left(x - \bar{x}_i\right)^T S_p^{-1}\left(x - \bar{x}_i\right) - 2\ln\left(p_i\right) & \text{IMTH} = 3 \end{cases}$$

For the leaving-out-one method of classification (IMTH = 4, 5, and 6), the sample mean vector and sample covariance matrices in the formula for

$$D_i^2(x)$$

are adjusted so as to remove the observation x from their computation. For linear discrimination (IMTH = 1, 2, 4, and 6), the linear discriminant function coefficients are actually used to compute the same posterior probabilities.

Using the posterior probabilities, each observations in X is classified into a group; the result is tabulated in the matrix CLASS and saved in the vector ICLASS. CLASS is not altered at this stage if X(i, IGRP) contains a group number that is out of range. If the reclassification method is specified, then all observations with no missing values in the NVAR classification variables are

classified. When the leaving-out-one method is used, observations with invalid group numbers, weights, frequencies or classification variables are not classified. Regardless of the frequency, a 1 is added (or subtracted) from CLASS for each row of X that is classified and contains a valid group number.

When IMTH > 3, adjustment is made to the posterior probabilities to remove the effect of the observation in the classification rule. In this adjustment, each observation is presumed to have a weight of X(*i*, IWT), if IWT > 0 and a frequency of 1.0. See Lachenbruch (1975, page 36) for the required adjustment.

Finally, when IDO = 5, the covariance matrices are computed from their *LU* factorizations.

Additional Example

Example 2

Continuing with Fisher's iris data, the following example computes the quadratic discriminant functions using values of IDO > 0. In the first loop, all observations are added to the functions, two observations at a time. In the second loop, each of three observations is classified, one by one, using the leaving-out-one method. Output for statistics that are identical to those reported in the first example are not printed here.

```
      USE GDATA_INT
      USE DSCRM_INT
      INTEGER    IGRP, IMTH, LDCLAS, LDCOEF, LDCOV, LDD2, LDPROB, &
                 LDX, LDXMEA, NCOL, NGROUP, NROW, NVAR
      PARAMETER  (IGRP=1, IMTH=2, LDPROB=10, LDX=150, &
                 NCOL=5, NGROUP=3, NROW=1, NVAR=4, LDCLAS=NGROUP, &
                 LDCOEF=NGROUP, LDCOV=NVAR, LDD2=NGROUP, LDXMEA=NGROUP)
!
      INTEGER    I, ICLASS(LDPROB), IDO, IND(4), IPRINT, NI(NGROUP), &
                 NOBS, NRMISS, NV
      REAL       CLASS(LDCLAS,NGROUP), COEF(LDCOEF,NVAR+1), &
                 COV(LDCOV,LDCOV,NGROUP+1), D2(LDD2,NGROUP), PRIOR(3), &
                 PROB(LDPROB,NGROUP), STAT(6+2*NGROUP), X(LDX,5), &
                 XMEAN(LDXMEA,NVAR)
!
      DATA IND/2, 3, 4, 5/, PRIOR/0.3333333, 0.3333333, 0.3333333/
!
      CALL GDATA (3, X, NOBS, NV)
!
      IPRINT = 0
      IDO    = 1
      CALL DSCRM (0, NVAR, X, NGROUP, COV, COEF, ICLASS, &
                  PROB, CLASS, D2, STAT, IDO=IDO, IND=IND, IGRP=IGRP, &
                  IMTH=IMTH, IPRINT=IPRINT, PRIOR=PRIOR, NI=NI, &
                  XMEAN=XMEAN, NRMISS=NRMISS)
!                                 Add the observations
      IDO = 2
      DO 10  I=1, NOBS
      CALL DSCRM (NROW, NVAR, X(I:,1:), NGROUP, COV, COEF, ICLASS, &
                   PROB, CLASS, D2, STAT, IDO=IDO, IND=IND, IGRP=IGRP, &
                   IMTH=IMTH, IPRINT=IPRINT, PRIOR=PRIOR, NI=NI, &
                   XMEAN=XMEAN, NRMISS=NRMISS)
```

```
   10 CONTINUE
!                                  Summarize the statistics
      IDO = 3
      CALL DSCRM (0, NVAR, X, NGROUP, COV, COEF, ICLASS, &
                  PROB, CLASS, D2, STAT, IDO=IDO, IND=IND, IGRP=IGRP, &
                  IMTH=IMTH, IPRINT=IPRINT, PRIOR=PRIOR, NI=NI, &
                  XMEAN=XMEAN, NRMISS=NRMISS)
!                                  Classify the first three observations
      IPRINT = 2
      IDO    = 4
      DO 20  I=1, 3
         CALL DSCRM (NROW, NVAR, X(I:,1:), NGROUP, COV, COEF, ICLASS(I:), &
                     PROB(I:,1:), CLASS, D2, STAT, IDO=IDO, IND=IND, &
                     IGRP=IGRP, IMTH=IMTH, IPRINT=IPRINT, PRIOR=PRIOR, &
                     NI=NI, XMEAN=XMEAN, NRMISS=NRMISS)
   20 CONTINUE
!                                  Release Workspace
      IDO = 6
      CALL DSCRM (0, NVAR, X, NGROUP, COV, &
                  COEF, ICLASS, PROB, CLASS, D2, STAT, IDO=IDO, IND=IND, &
                  IGRP=IGRP, IMTH=IMTH, IPRINT=IPRINT, PRIOR=PRIOR, NI=NI, &
                  XMEAN=XMEAN, NRMISS=NRMISS)
!
      END
```

Output

```
ICLASS, the classifications

Obs.  Class
1     1

PROB, the posterior probabilities
    1       2       3
1.000   0.000   0.000

ICLASS, the classifications

Obs.  Class
1     1

PROB, the posterior probabilities
    1       2       3
1.000   0.000   0.000

ICLASS, the classifications
Obs.  Class
1     1

PROB, the posterior probabilities
    1       2       3
1.000   0.000   0.000
```

DMSCR

Uses Fisher's linear discriminant analysis method to reduce the number of variables.

Required Arguments

XMEAN — NGROUP by NVAR matrix containing the means of the variables in each group. (Input)

SUMWT — Vector of length NGROUP containing the sum of the weights of the observations in each group. (Input)

COV — NVAR by NVAR matrix containing the pooled within-groups variance-covariance matrix S_p. (Input)

NNV — Number of eigenvectors extracted from

$$S_p^{-1} S_b$$

the standardized between-groups variance-covariance matrix. (Output)
S_p is the pooled within-groups variance-covariance matrix, and S_b is the between-groups variance-covariance matrix. NNV is usually the minimum of NVAR and NGROUP − 1, but it may be smaller if any row of XMEAN or COV is a linear combination of the other rows.

EVAL — Vector of length NNV containing the eigenvalues extracted from the standardized between-means variancecovariance matrix, in descending order. (Output)
NNV is less than or equal to the minimum of NVAR and (NGROUP − 1).

COEF — NVAR by NNV matrix of eigenvectors from the standardized between-means variance-covariance matrix. (Output)
The eigenvector coefficients have been standardized such that the canonical scores can be obtained directly by multiplication of the original data by COEF.

CMEAN — NGROUP by NNV matrix of group means of the canonical variables. (Output)

Optional Arguments

NGROUP — Number of groups. (Input)
Default: NGROUP = size (XMEAN,1).

NVAR — Number of variables. (Input)
Default: NVAR = size (XMEAN,2).

LDXMEA — Leading dimension of XMEAN exactly as specified in the dimension statement in the calling program. (Input)
Default: LDXMEA = size (XMEAN,1).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOEF = size (COEF,1).

LDCMEA — Leading dimension of CMEAN exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCMEA = size (CMEAN,1).

FORTRAN 90 Interface

Generic: `CALL DMSCR (XMEAN, SUMWT, COV, NNV, EVAL, COEF, CMEAN [,…])`

Specific: The specific interface names are S_DMSCR and D_DMSCR.

FORTRAN 77 Interface

Single:
```
CALL DMSCR (NGROUP, NVAR, XMEAN, LDXMEA, SUMWT, COV, LDCOV,
NNV, EVAL, COEF, LDCOEF, CMEAN, LDCMEA)
```

Double: The double precision name is DDMSCR.

Example

The following example illustrates a typical sequence. Fisher's iris data is used. (See routine GDATA, Chapter 19, Utilities). Routine DSCRM is first used to perform a discriminant analysis based on all the variables. COV, XMEAN, and NI are obtained from DSCRM. Function DMSCR, which uses these arrays, is then called.

```
      USE IMSL_LIBRARIES
      INTEGER    IGRP, IMTH, IPRINT, LDCLAS, LDCMEA, LDCO, LDCOEF, &
                 LDCOV, LDD2, LDPROB, LDX, LDXMEA, NCOL, &
                 NGROUP, NROW, NVAR
      PARAMETER  (IGRP=1, IMTH=3, IPRINT=0, LDCOV=4, NCOL=5, NGROUP=3, &
                  NROW=150, NVAR=4, LDCLAS=NGROUP, LDCMEA=NGROUP, &
                  LDCO=NGROUP, LDCOEF=NVAR, LDD2=NGROUP, LDPROB=NROW, &
                  LDX=NROW, LDXMEA=NGROUP)
!
      INTEGER    ICLASS(NROW), IND(4), NI(NGROUP), NNV, NOBS, NOUT, &
                NRMISS, NV
      REAL       CLASS(LDCLAS,NGROUP), CMEAN(LDCMEA,NGROUP-1), &
                 CO(LDCO,NVAR+1), COEF(LDCOEF,NGROUP-1), &
                 COV(LDCOV,LDCOV,1), D2(LDD2,NGROUP), EVAL(NGROUP-1), &
                 PRIOR(3), PROB(LDPROB,NGROUP), REAL, &
                 STAT(6+2*NGROUP), SUMWT(NGROUP), X(LDX,5), &
                 XMEAN(LDXMEA,NVAR)
      INTRINSIC  REAL
!
```

```
      DATA IND/2, 3, 4, 5/, PRIOR/0.3333333, 0.3333333, 0.3333333/
!
      CALL GDATA (3, X, NOBS, NV)
!
      CALL DSCRM (NROW, NVAR, X, NGROUP, COV(1:,1:,1), CO, ICLASS, &
                  PROB, CLASS, D2, STAT, IND=IND, IGRP=IGRP, IMTH=IMTH, &
                  PRIOR=PRIOR, XMEAN=XMEAN)
!
      SUMWT(1) = STAT(6+NGROUP)
      SUMWT(2) = STAT(7+NGROUP)
      SUMWT(3) = STAT(8+NGROUP)
!
      CALL DMSCR (XMEAN, SUMWT, COV(1:,1:,1), NNV, EVAL, COEF, CMEAN)

      CALL UMACH (2, NOUT)
      WRITE (NOUT,'('' NNV = '',I1)') NNV
      CALL WRRRN ('EVAL', EVAL, 1, NNV, 1)
      CALL WRRRN ('COEF', COEF)
      CALL WRRRN ('CMEAN', CMEAN)
      END
```

Output

```
NNV = 2

     EVAL
        1       2
32.19    0.29

        COEF
           1       2
1  -0.829   0.024
2  -1.534   2.165
3   2.201  -0.932
4   2.810   2.839

       CMEAN
           1       2
1  -5.502   6.877
2   3.930   5.934
3   7.888   7.174
```

Comments

1. Workspace may be explicitly provided, if desired, by use of D2SCR/DD2SCR. The reference is:

   ```
   CALL D2SCR (NGROUP, NVAR, XMEAN, LDXMEA, SUMWT, COV, LDCOV, NNV,
   EVAL, COEF, LDCOEF, CMEAN, LDCMEA, BCOV, EVAL2, EVEC, WKR, WK)
   ```

   The additional arguments are as follows:

   BCOV — Work array of length NVAR * NVAR.

   EVAL2 — Work array of length NVAR.

EVEC — Work array of length NVAR * NVAR.

WKR — Work array of length NVAR * NVAR.

WK — Work array of length 2 * NVAR.

2. IMSL routine DSCRM (page 979) may be used to calculate the input arrays for this routine from the original data.

Description

Routine DMSCR is a natural generalization of R.A. Fisher's linear discrimination procedure for two groups. This method of discrimination obtains those linear combinations of the observed random variables that maximize the between-groups variation relative to the withingroups variation. Denote the first of these linear combinations by

$$z_1 = \beta_1^T x$$

where β_1 is a column vector of coefficients of length NVAR and x is an observation to be classified. On the basis of one linear combination, the discriminant rule assigns the observation, z, to a group (characterized by the group mean) by minimizing the Euclidean distance between z and the group mean.

To obtain β_1 (see, e.g., Tatsuoka 1971, page 158), let S_p denote the pooled within-groups covariance matrix (S_p is defined and can be computed via routine DSCRM, page 979) and let S_b denote the between-groups covariance matrix defined by

$$S_b = \sum_{i=1}^{g} w_i \left(\bar{x}_i - \bar{x}\right)\left(\bar{x}_i - \bar{x}\right)^T / \left(N - g\right)$$

where g is the number of groups,

$$\bar{x}_i$$

is the mean vector for the i-th group of observations, $\bar{x}$ denotes the vector of means over all observations, w_i is the sum of the weights times the frequencies as input in SUMWT and as used in the computation of

$$\bar{x}_i$$

and N is the total number of observations used in computing COV. Then, β_1, such that

$$\beta_1^T S_p \beta_1 = 1$$

can be computed as the maximum of

$$\psi = \beta_1^T S_b \beta_1$$

This yields β_1 as the eigenvector associated with the largest eigenvalue from

$$S_p^{-1} S_b$$

Generally,

$$S_p^{-1} S_b$$

has rank m, where $m = \min(g - 1, p)$ and $p =$ NVAR.

$$S_p^{-1} S_b$$

has m such eigenvectors, and the matrix COEF is obtained as $(\beta_1, \beta_2, \ldots, \beta_m)$, where each β_i is an eigenvector.

The matrix CMEAN is taken as the within-group means vector of the linear combinations z_i defined by the β's. For each observation x, scores

$$z_i = \beta_i^T x$$

can be computed, because of the restriction on β_i, the sample variance of the z_i is 1.0. The observation is classified into the group (as specified by the group mean of the z_i's) to which, on the basis of the z_i, the Euclidean distance is the least.

Note that the linear combinations z_i have meaning even when discrimination is not desired. The linear combination of the observed variables that most separates the g groups is z_1; z_2, giving the second highest such separation orthogonal to the first, and so on. Thus, a plot of the mean vectors of the first two variables gives a good two-dimensional summarization of the relationships between the groups.

NNBRD

Performs k nearest neighbor discrimination.

Required Arguments

X — NROW by NVAR + 1 matrix containing the data to be used on this call. (Input/Output)
One column in X must contain the group number for each observation. On output, X is sorted into a *k-d* tree. The first NRULE + NCLASS rows of X must not contain missing values in the columns specified by IND and IGRP.

K — Number of nearest neighbors to be used in the discriminant rule. (Input)

NGROUP — Number of groups in the data. (Input)

NRULE — Number of observations in X to be used in the discriminant rule. (Input)
The first |NRULE| observations in X are used as the set defining the rule. If NRULE is positive, then the NRULE observations defining the rule are classified. If NRULE is negative, the NRULE observations defining the rule are not classified.

NCLASS — Number of observations in X to classify. (Input)
NCLASS is the number of observations in a second sample that may be used to test the

rule formed from the first NRULE observations. If present, this sample is in rows NRULE + 1 through NRULE + NCLASS of X.

THRESH — Threshold for the posterior probabilities. (Input)
If the maximum posterior probability is less than THRESH, the observation is classified into group NGROUP + 1 (the group "other").

PART — Vector of length NRULE containing the values to be used in the partition of X for the *k-d* tree. (Output)

IDISCR — Vector of length NRULE containing the element number in IND that points to the column of X to be used as the discriminator in the *k-d* tree. (Output)
IDISCR(*i*) = 0 if the observation is a terminal node. IND(IDISCR(*i*)) is the column number in X to be used as the discriminator.

NI — Vector of length NGROUP containing the number of observations in each group. (Output)

ICLASS — Vector of length *m* containing the group to which the observation was classified. (Output)
If NRULE > 0, *m* = NRULE + NCLASS; otherwise, *m* = NCLASS. The *i*-th element in ICLASS corresponds to to *i*-th row in the sorted matrix X.

PROB — *m* by NGROUP matrix containing the posterior probabilities for each observation. (Output)
The *i*-th row in PROB corresponds to the *i*-th row in the in the sorted matrix X.

CLASS — NGROUP by NGROUP + 1 matrix containing the classification table. (Output)
Each observation that is classified and has a group number equal to 1.0, 2.0, …, NGROUP is entered into the table. The rows of the table correspond to the known group membership. The columns refer to the group to which the observation was classified. Column NGROUP + 1 refers to the column "other" (see THRESH).

Optional Arguments

NROW — Number of rows of X that contain an observation. (Input)
Default: NROW = size (X,1).

NVAR — Number of variables to be used in the discrimination. (Input)
Default: NVAR = size (X,2) – 1.

NCOL — Number of columns in matrix X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IND — Vector of length NVAR containing the column numbers in X to be used in the discrimination. (Input)
By default, IND(I)=1.

IGRP — Column number in X containing the group numbers. (Input)
The group numbers must be 1.0, 2.0, …, NGROUP for an observation to be used in the discriminant functions. (Note, however, that the nearest integer (NINT) function is used to obtain the group numbers.)
Default: IGRP = NVAR + 1.

METRIC — Metric to be used in computing the *k* nearest neighbors. (Input)
Default: METRIC = 0.

METRIC	Metric used
0	Euclidean distance
1	L_1 norm
2	L_∞ norm

PRIOR — Vector of length NGROUP containing the prior probabilities for each group. (Input, if PRIOR(1) is not −1.0; input/output, if PRIOR(1) is −1.0)
If PRIOR(1) is not −1.0, then the elements of PRIOR should sum to 1.0. Proportional priors can be selected by setting PRIOR(1) = −1.0. In this case, the prior probabilities will be proportional to the sample size in each group based upon the first NRULE observations, and the elements of PRIOR will contain the proportional prior probabilities on return from NNBRD.
Default: PRIOR(1) = −1.0.

LDPROB — Leading dimension of PROB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDPROB = size (PROB,1).

LDCLAS — Leading dimension of CLASS exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCLAS = size (CLASS,1).

FORTRAN 90 Interface

Generic: `CALL NNBRD (X, K, NGROUP, NRULE, NCLASS, THRESH, PART, IDISCR, NI, ICLASS, PROB, CLASS [,…])`

Specific: The specific interface names are S_NNBRD and D_NNBRD.

FORTRAN 77 Interface

Single: `CALL NNBRD (NROW, NVAR, NCOL, X, LDX, K, IND, IGRP, NGROUP, NRULE, NCLASS, METRIC, PRIOR, THRESH, PART, IDISCR, NI, ICLASS, PROB, LDPROB, CLASS, LDCLAS)`

Double: The double precision name is `DNNBRD`.

Example

Fisher's iris data are used to illustrate routine `NNBRD`. The data consist of three types of iris. `NNBRD` is called with $k = 5$ and Euclidean distance as the metric. The results show a clear separation of the groups.

```
      USE GDATA_INT
      USE NNBRD_INT
      USE WRRRN_INT
      USE WRIRN_INT
      INTEGER    IGRP, K, LDCLAS, LDPROB, LDX, NCLASS, NCOL, &
                 NGROUP, NROW, NRULE, NVAR
      REAL       THRESH
      PARAMETER  (IGRP=1, K=5, LDCLAS=3, LDPROB=150, LDX=150, &
                 NCLASS=0, NCOL=5, NGROUP=3, NROW=150, &
                 NRULE=150, NVAR=4, THRESH=0.10)
!
      INTEGER    ICLASS(NROW), IDISCR(NROW), IND(NVAR), NI(NGROUP), &
                 NRA, NRB
      REAL       CLASS(LDCLAS,NGROUP+1), PART(NRULE), PRIOR(NGROUP), &
                 PROB(LDPROB,NGROUP), X(LDX,NCOL)
!
      DATA IND/2, 3, 4, 5/
!
      CALL GDATA (3, X, NRA, NRB)
!
      PRIOR(1) = -1.0
      CALL NNBRD (X, K, NGROUP, NRULE, NCLASS, THRESH, PART, IDISCR, &
                 NI, ICLASS, PROB, CLASS, IND=IND, IGRP=IGRP, PRIOR=PRIOR)
      CALL WRRRN ('The first 10 rows of X', X, 10, NCOL, LDX)
      CALL WRRRN ('PRIOR', PRIOR, 1, NGROUP, 1)
      CALL WRRRN ('The first 10 elements of PART', PART, 1, 10, 1)
      CALL WRIRN ('The first 10 elements of IDISCR', IDISCR, 1, 10, 1)
      CALL WRIRN ('NI', NI, 1, NGROUP, 1)
      CALL WRIRN ('The first 10 elements of ICLASS', ICLASS, 1, 10, 1)
      CALL WRRRN ('The first 10 rows of PROB', PROB, 10, NGROUP, LDPROB)
      CALL WRRRN ('CLASS', CLASS, NGROUP, NGROUP, LDCLAS)
!
      END
```

Output

```
          The first 10 rows of X
          1       2       3       4       5
 1    1.000   4.500   2.300   1.300   0.300
 2    1.000   4.400   2.900   1.400   0.200
```

```
 3   1.000   4.800   3.000   1.400   0.300
 4   1.000   4.400   3.000   1.300   0.200
 5   1.000   4.800   3.000   1.400   0.100
 6   1.000   4.300   3.000   1.100   0.100
 7   1.000   4.600   3.100   1.500   0.200
 8   1.000   4.900   3.100   1.500   0.100
 9   1.000   4.900   3.000   1.400   0.200
10   1.000   4.900   3.100   1.500   0.200


          PRIOR
     1        2        3
0.3333   0.3333   0.3333

                    The first 10 elements of PART
    1      2      3      4      5      6      7      8      9     10
0.000  0.000  3.000  0.000  3.000  0.000  0.000  4.900  0.000  3.100

    The first 10 elements of IDISCR
1   2   3   4   5   6   7   8   9  10
0   0   2   0   2   0   0   1   0   2

     NI
 1    2    3
50   50   50

    The first 10 elements of ICLASS
1   2   3   4   5   6   7   8   9  10
1   1   1   1   1   1   1   1   1   1

The first 10 rows of PROB
         1       2       3
 1   1.000   0.000   0.000
 2   1.000   0.000   0.000
 3   1.000   0.000   0.000
 4   1.000   0.000   0.000
 5   1.000   0.000   0.000
 6   1.000   0.000   0.000
 7   1.000   0.000   0.000
 8   1.000   0.000   0.000
 9   1.000   0.000   0.000
10   1.000   0.000   0.000

           CLASS
        1       2       3
1   50.00    0.00    0.00
2    0.00   47.00    3.00
3    0.00    2.00   48.00
```

Comments

Workspace may be explicitly provided, if desired, by use of N2BRD/DN2BRD. The reference is:

```
CALL N2BRD (NROW, NVAR, NCOL, X, LDX, K, IND, IGRP, NGROUP, NRULE,
NCLASS, METRIC, PRIOR, THRESH, PART, IDISCR, NI, ICLASS, PROB,
LDPROB, CLASS, LDCLAS, WK, IWK, ILOW, IHIGH, ISIDE, BNDL, BNDH, XKEY,
IPQR, PQD)
```

The additional arguments are as follows:

WK — Work vector of length NROW.

IWK — Work vector of length NROW.

ILOW — Work vector of length $\log_2(\text{NROW}) + 3$.

IHIGH — Work vector of length $\log_2(\text{NROW}) + 3$.

ISIDE — Work vector of length $\log_2(\text{NROW}) + 3$.

BNDL — Work vector of length NVAR * ($\log_2(\text{NROW}) + 3$).

BNDH — Work vector of length NVAR * ($\log_2(\text{NROW}) + 3$).

XKEY — Work vector of length NVAR.

IPQR — Work vector of length K + 1.

PQD — Work vector of length K + 1.

Description

Routine NNBRD performs *k*-th nearest neighbor discriminant function analysis. The *k-d* tree algorithm of Friedman, Bentley, and Finkel (1977) is used to find the nearest neighbors. Consult this reference for a discussion of *k-d* trees and how one goes about finding nearest neighbors in them.

In NNBRD, the *k* nearest neighbors of any observation used in forming the rule (i.e., one of the first NRULE observations in X), do not include the observation. Let $k_i (i = 1, \ldots, \text{NGROUP})$ denote the number of nearest neighbors found from each of the groups for a given observation ($\sum_i k_i = k$); let $p_i = \text{PRIOR}(i)$($\sum_i p_i = 1$); and let

$$\hat{\theta}_i$$

denote the estimated posterior probability of membership in group *i*. Compute

$$\hat{\theta}_i \text{ as } \hat{\theta}_i = \frac{k_i p_i / n_i}{\sum_{j=1}^{g} k_j p_j / n_j}$$

where $g = \text{NGROUP}$. (If $n_j = 0$ for some *j*, the associated term in the denominator is excluded and

$$\hat{\theta}_j$$

is set to 0.0.)

Let m denote the index of the maximum

$$\hat{\theta}_i$$

and ϕ = THRESH. Then if

$$\hat{\theta}_m > \phi$$

the observation is classified into group m. If

$$\hat{\theta}_m \leq \phi$$

or if the maximum $\hat{\theta}$ is not unique, then the observation is not classified into any group and ICLASS is set to zero.

Three metrics are available in NNBRD for finding the nearest neighbors. These are Euclidean (L_2) distance, L_1 norm, and L_∞ norm. In order to use Mahalanobis distance, $x^T\Sigma^{-1}x$, a transformation $y = \Sigma^{-1/2}x$ is first needed so that Var(y) = I. These transformations can be accomplished by use of the mathematical routines. The L_2 norm would then be used with y as input to obtain the Mahalanobis metric.

Chapter 11: Cluster Analysis

Routines

Usage Notes

The routines described in this chapter perform various forms of hierarchical or *K*-means cluster analysis. By appropriate manipulation of the input data, either variables or cases may be clustered. Additionally, for hierarchical clustering, similarity or dissimilarity (distance) matrices created by routines not included in this chapter can be clustered. Hartigan (1975) and Anderberg (1973) are general references that may be used in this chapter.

The first step in agglomerative hierarchical cluster analysis is to compute the distance between each observation (or variable). Initially, each observation (variable) is treated as a cluster. The two clusters that are closest to one another in distance are merged, and the distance of the new cluster from all other clusters is computed. This process continues until only one cluster remains. No attempt at finding an optimal clustering (in the sense of minimizing some criterion) is made.

The usual steps in a hierarchical cluster analysis might proceed as follows:

1. Routine `CDIST` (page 1006) is used to compute a distance (or possibly a similarity) matrix from the input data matrix. A scaled matrix of Euclidean distances is a common choice for a distance matrix, while a correlation matrix is a common choice for a similarity matrix. If a correlation matrix is to be used, many of the routines described in Chapter 3, "Correlation", may also be used to compute the correlation measures for the matrix. In particular, routine `CORVC` (see Chapter 3, Correlation) from this chapter may be used.

2. Once the distance matrix has been computed, routine `CLINK` (page 1011) is used to perform the agglomerative hierarchical cluster analysis using either single, complete, average, or Ward's linkage.

3. The results obtained from CLINK are examined, and if desired, the number of clusters is selected. Routine TREEP in Chapter 16, "Line Printer Graphics," may be used to print the cluster tree. This tree may aid in selecting the number of clusters, assuming that such a number is desired. Based upon the number of clusters selected, routine CNUMB (page 1015) is used to obtain the cluster number of each of the clustered observations (or variables).

4. Routines described in Chapter 1, "Basic Statistics," and other chapters in the IMSL STAT/LIBRARY are used to obtain descriptive and other statistics to evaluate the clustering.

Because routine CDIST produces similarity and distance matrices for either rows or columns, it is easy to cluster either observations or variables. Optionally, the user may wish to cluster a correlation matrix obtained from one of the routines in the correlation chapter or to input a matrix of similarities (or dissimilarities) obtained via experimentation. The objects within such matrices may be clustered directly in routine CLINK.

Basic *K*-means clustering attempts to find a clustering that minimizes the within-cluster sums of squares. In this method of clustering the data, matrix X is grouped so that each observation (row in X) is assigned to one of a fixed number, *K*, of clusters. The sum of the squared difference of each observation about its assigned clusters mean is used as the criterion for assignment. In the basic algorithm, observations are transferred from one cluster to another when doing so decreases the within-cluster sums of squared differences. When, in a pass through the entire data set, no transfer occurs, the algorithm stops. Routine KMEAN (page 1019) is one implementation of the basic algorithm.

The usual course of events in *K*-means cluster analysis might be to use routine KMEAN to obtain the optimal clustering. The clustering is then evaluated via routines described in Chapter 1, "Basic Statistics," and/or other chapters in the IMSL STAT/LIBRARY. Often, *K*-means clustering with more than one value for *K* is performed, and the value of *K* that best fits the data is used.

Clustering can be performed either on observations or on variables. The discussion of the routine KMEAN assumes the clustering is to be performed on the observations, which correspond to the rows of the input data matrix. If variables, rather than observations, are to be clustered using KMEAN, the data matrix should first be transposed (possibly using routine TRNRR (IMSL MATH/LIBRARY)). In the documentation for KMEAN, the words "observation" and "variable" would then be exchanged.

CDIST

Computes a matrix of dissimilarities (or similarities) between the columns (or rows) of a matrix.

Required Arguments

X — NROW by NCOL matrix containing the data. (Input)

DIST — *m* by *m* matrix containing the computed dissimilarities or similarities, where *m* = NROW if IROW = 1 and *m* = NCOL otherwise. (Output)

Optional Arguments

NROW — Number of rows in the matrix. (Input)
Default: NROW = size (X,1).

NCOL — Number of columns in the matrix. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

NDSTM — Number of rows (columns, if IROW = 1) to be used in computing the distance measure between the columns (rows). (Input)
Default: NDSTM = size (IND,1) if IND is present. Otherwise, a default value of 2 is used.

IND — Vector of length NDSTM containing the indices of the rows (columns, if IROW = 1) to be used in computing the distance measure. (Input)
If IND(1) = 0; the first NDSTM rows (columns) are used.
By default, the first NDSTM rows(columns) are used.

IMETH — Method to be used in computing the dissimilarities or similarities. (Input)
Default: IMETH = 0.

IMETH	Method
0	Euclidean distance (L_2 norm)
1	Sum of the absolute differences (L_1 norm)
2	Maximum difference (L_∞ norm)
3	Mahalanobis distance
4	Absolute value of the cosine of the angle between the vectors
5	Angle in radians $(0, \pi)$ between the lines through the origin defined by the vectors
6	Correlation coefficient
7	Absolute value of the correlation coefficient
8	Number of exact matches

The algorithm section of the manual document has a more detailed description of each measure.

IROW — Row or columns option. (Input)
If IROW = 1, distances are computed between the NROW rows of X. Otherwise, distances between the NCOL columns of X are computed.
Default: IROW = 1.

ISCALE — Scaling option. (Input)
ISCALE is not used for methods 3 through 8.
Default: ISCALE = 0.

ISCALE	Scaling Performed
0	No scaling is performed.
1	Scale each column (row, if IROW = 1) by the standard deviation of the column (row).
2	Scale each column (row, if IROW = 1) by the range of the column (row).

LDDIST — Leading dimension of DIST exactly as specified in the dimension statement in the calling program. (Input)
Default: LDDIST = size (DIST,1).

FORTRAN 90 Interface

Generic: `CALL CDIST (X, DIST [,…])`

Specific: The specific interface names are S_CDIST and D_CDIST.

FORTRAN 77 Interface

Single:
```
CALL CDIST (NROW, NCOL, X, LDX, NDSTM, IND, IMETH, IROW,
ISCALE, DIST, LDDIST)
```

Double: The double precision name is DCDIST.

Example

The following example illustrates the use of CDIST for computing the Euclidean distance between the rows of a matrix.

```
      USE WRRRN_INT
      USE CDIST_INT
      INTEGER    IROW, LDDIST, LDX, NCOL, NDSTM, NROW
      PARAMETER  (IMETH=0, IROW=1, NCOL=2, NROW=4, LDDIST=NROW, LDX=NROW)
!
      REAL       DIST(LDDIST,NROW), X(NROW,NCOL)
!
      DATA IND/0/
      DATA X/1, 1, 1, 1, 1, 0, -1, 2/
```

```
      DATA DIST/16*0.0/
!                                     Print input matrix
      CALL WRRRN ('X', X)
!
      CALL CDIST (X, DIST)
!                                     Print distance matrix
      CALL WRRRN ('DIST', DIST)
!
      END
```

Output

```
          X
          1       2
1    1.000   1.000
2    1.000   0.000
3    1.000  -1.000
4    1.000   2.000

               DIST
          1       2       3       4
1    0.000   1.000   2.000   1.000
2    0.000   0.000   1.000   2.000
3    0.000   0.000   0.000   3.000
4    0.000   0.000   0.000   0.000
```

Comments

1. Workspace may be explicitly provided, if desired, by use of C2IST/DC2IST. The reference is:

   ```
   CALL C2IST (NROW, NCOL, X, LDX, NDSTM, IND, IMETH, IROW, ISCALE,
   DIST, LDDIST, X1, X2, SCALE, WK, IND1)
   ```

   The additional arguments are as follows:

   X1 — Work vector of length NDSTM. Not used if IMETH = 8.

   X2 — Work vector of length NDSTM. Not used if IMETH = 8.

   SCALE — Work vector of length NDSTM if IMETH is less than 4; of length NCOL or NROW when IROW is 0 or 1, respectively, and IMETH is 4 or 5; and of length 2 * NCOL or 2 * NROW when IROW is 0 or 1 and IMETH is 6 or 7. SCALE is not used when IMETH is 8.

   WK — Work vector of length NDSTM * NDSTM when IMETH is 3, or of length NDSTM when IMETH = 6 or 7. Not used otherwise.

   IND1 — Integer work vector of length NDSTM.

2. Informational error
   Type Code

3	3	A variable is numerically linearly dependent on the previous variables when IMETH is 3. The variable detected as being linearly dependent is omitted from the distance measure.

Description

Routine CDIST computes an upper triangular matrix (excluding the diagonal) of dissimilarities (or similarities) between the columns or rows of a matrix. Nine different distance measures can be computed. For the first three measures, three different scaling options can be employed. Output from CDIST is generally used as input to clustering or multidimensional scaling routines.

The following discussion assumes that the distance measure is being computed between the columns of the matrix, i.e., that IROW is not 1. If distances between the rows of the matrix are desired, set IROW to 1.

For IMETH = 0 to 2, each row of X is first scaled according to the value of ISCALE. The scaling parameters are obtained from the values in the row scaled as either the standard deviation of the row or the row range; the standard deviation is computed from the unbiased estimate of the variance. If ISCALE is 0, no scaling is performed, and the parameters in the following discussion are all 1.0. Once the scaling value (if any) has been computed, the distance between column i and column j is computed via the difference vector $z_k = (x_k - y_k)/s_k$, $i = 1, \ldots,$ NDSTM, where x_k denotes the k-th element in the i-th column, and y_k denotes the corresponding element in the j-th column. For given z_i, the metrics 0 to 2 are defined as:

IMETH	***Metric***	
0	$\sqrt{\left(\sum_{i=1}^{\text{NDSTM}} z_i^2\right)}$	Euclidean distance
1	$\sum_{i=1}^{\text{NDSTM}} \lvert z_i \rvert$	L_1 norm
2	$\max_i \lvert z_i \rvert$	L_∞ norm

Distance measures corresponding to IMETH = 3 to 8 do not allow for scaling. These measures are defined via the column vectors $X = (x_i)$, $Y = (y_i)$, and $Z = (x_i - y_i)$ as follows:

IMETH	***Metric***
3	$Z'\hat{\Sigma}^{-1}Z =$ Mahalanobis distance, where $\hat{\Sigma}$ is the usual unbiased sample estimate of the covariance matrix of the rows.
4	$\cos(\theta) = X^T Y / \left(\sqrt{X^T X}\sqrt{Y^T Y}\right) =$ the dot product of X and Y divided by the length of X times the length of Y.
5	θ, where θ is defined in 4.
6	ρ = the usual (centered) estimate of the correlation between X and Y.
7	The absolute value of ρ (where ρ is defined in 6).

8 The number of times $x_i = y_i$, where x_i and y_i are elements of X and Y.

For the Mahalanobis distance, any variable used in computing the distance measure that is (numerically) linearly dependent upon the previous variables in the IND vector is omitted from the distance measure.

CLINK

Performs a hierarchical cluster analysis given a distance matrix.

Required Arguments

DIST — NPT by NPT matrix containing the distance (or similarity) matrix.(Input/Output)
DIST is a symmetric matrix. On input, only the upper triangular part needs to be present. The routine CLINK saves the upper triangular part of DIST in the lower triangle. On return from CLINK, the upper triangular part of DIST is restored, and the matrix has been made symmetric.

CLEVEL — Vector of length NPT − 1 containing the level at which the clusters are joined. (Output)
CLEVEL(k) contains the distance (or similarity) level at which cluster NPT + k was formed. If the original data in DIST was transformed via the option parameter IDIST, the inverse transformation is applied to the values in CLEVEL prior to exit from CLINK.

ICLSON — Vector of length NPT − 1 containing the left sons of each merged cluster. (Output)
Cluster NPT + k is formed by merging clusters ICLSON(k) and ICRSON(k).

ICRSON — Vector of length NPT − 1 containing the right sons of each merged cluster. (Output)
Cluster NPT + k is formed by merging clusters ICLSON(k) and ICRSON(k).

Optional Arguments

NPT — Number of data points to be clustered. (Input)
Default: NPT = size (DIST,2).

IMETH — Option giving the method to be used for clustering. (Input)
Default: IMETH = 0.

IMETH	Method
0	Single linkage (minimum distance)
1	Complete linkage (maximum distance)

2 Average distance within (average distance between objects within the merged cluster)

3 Average distance between (average distance between objects in the two clusters)

4 Ward's method (minimize the within-cluster sums of squares). For Ward's method, the elements of DIST are assumed to be Euclidean distances.

IDIST — Option giving the type of transformation to be applied to the measures in DIST. (Input)
Default: IDIST = 0.

IDISTTransformation

0 No transformation is required. The elements of DIST are distances.

1 Convert similarities to distances by multiplication by –1.0.

2 Convert similarities (usually correlations) to distances by taking the reciprocal of the absolute value.

LDDIST — Leading dimension of DIST exactly as specified in the dimension statement in the calling program. (Input)
Default: LDDIST = size (DIST,1).

FORTRAN 90 Interface

Generic: `CALL CLINK (DIST, CLEVEL, ICLSON, ICRSON [,…])`

Specific: The specific interface names are S_CLINK and D_CLINK.

FORTRAN 77 Interface

Single: `CALL CLINK (NPT, IMETH, IDIST, DIST, LDDIST, CLEVEL, ICLSON, ICRSON)`

Double: The double precision name is DCLINK.

Example

In the following example, the average distance within clusters method is used to perform a hierarchical cluster analysis of the Fisher iris data. Routine GDATA (see Chapter 19, Utilities) is first used to obtain the Fisher iris data. The example is typical in that after the program obtains the data, routine CDIST (page 1006) computes the distance matrix (DIST) prior to calling CLINK.

```
      USE GDATA_INT
      USE CDIST_INT
      USE CLINK_INT
      USE UMACH_INT
      INTEGER    IDATA, IMETH, IPRINT, IROW, ISCALE, LDDIST, LDX,&
                 NCOL, NPT, NROW, NVAR
      PARAMETER  (IDATA=3, IMETH=2, IPRINT=0, IROW=1, ISCALE=1, &
                 NCOL=5, NROW=150, NVAR=4, LDX=NROW, &
                 NPT=NROW, LDDIST=LDX)
!
      INTEGER    I, ICLSON(NROW-1), ICRSON(NROW-1), IND(4), NOUT, &
                 NXCOL, NXROW
      REAL       CLEVEL(NROW-1), DIST(LDDIST,LDDIST), X(LDX,NCOL)
!
      DATA IND/2, 3, 4, 5/
!
      CALL GDATA (IDATA, X, NXROW, NXCOL)
!                                Compute the distances
      CALL CDIST (X, DIST, IND=IND, ISCALE=ISCALE)
!                                Clustering
      CALL CLINK (DIST, CLEVEL, ICLSON, ICRSON, IMETH=IMETH)
!                                Print some results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99996) (I,I=1,149,15)
      WRITE (NOUT,99997) (CLEVEL(I),I=1,149,15)
      WRITE (NOUT,99998) (ICLSON(I),I=1,149,15)
      WRITE (NOUT,99999) (ICRSON(I),I=1,149,15)
!
99996 FORMAT (' OBS # ', 10I6)
99997 FORMAT (' CLEVEL', 10F6.2)
99998 FORMAT (' ICLSON', 10I6)
99999 FORMAT (' ICRSON', 10I6)
!
      END
```

Output

```
OBS #      1    16    31    46    61    76    91   106   121   136
CLEVEL  0.00  0.17  0.23  0.27  0.31  0.37  0.41  0.48  0.60  0.78
ICLSON   143   153    17   140    53   198   186   218   261   249
ICRSON   102    29     6   113    51    91   212   243   266   262
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `C2INK/DC2INK`. The reference is:

   ```
   CALL C2INK (NPT, IMETH, IDIST, DIST, LDDIST, CLEVEL, ICLSON,
   ICRSON, IPTR, ICLUS, CWT, CSUM)
   ```

   The additional arguments are as follows:

   IPTR — Integer work vector of length `NPT`.

ICLUS — Integer work vector of length NPT.

CWT — Work vector of length NPT. Not used if IMETH = 0 or 1.

CSUM — Work vector of length NPT. Not used if IMETH = 0 or 1.

2. The clusters corresponding to the original data points are numbered from 1 to NPT. The NPT − 1 clusters formed by merging clusters are numbered NPT + 1 to NPT + (NPT − 1).

3. Raw correlations, if used as similarities, should be made positive and transformed to a distance measure. One such transformation can be performed by specifying IDIST = 2 in CLINK.

4. The user may cluster either variables or observations in CLINK since a dissimilarity matrix, not the original data, is used. Routine CDIST (page 1006) may be used to compute the matrix DIST.

5. Routine TREEP (see Chapter 16, Line Printer Graphics) in the graphics chapter can be used to obtain a line printer plot of the clustering tree. Routine CNUMB (page 1015) can be used to obtain the cluster number assigned to each of the original clusters when a specified number of clusters is desired.

Description

Routine CLINK performs hierarchical cluster analysis based upon a distance matrix, or by appropriate use of the IDIST option, based upon a similarity matrix. Only the upper triangular part of the matrix needs to be input to CLINK.

Hierarchical clustering in CLINK proceeds as follows. Initially, each data point is considered to be a cluster, numbered 1 to n = NPT.

1. If the data matrix contains similarities, they are converted to distances by the method specified in IDIST. Set $k = 1$.

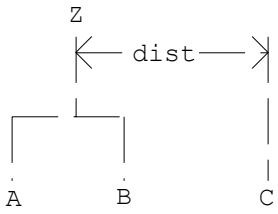
2. A search is made of the distance matrix to find the two closest clusters. These clusters are merged to form a new cluster, numbered $n + k$. The cluster numbers of the two clusters joined at this stage are saved in ICRSON and ICLSON, and the distance measure between the two clusters is stored in CLEVEL.

3. Based upon the method of clustering, updating of the distance measure in the row and column of DIST corresponding to the new cluster is performed.

4. Set $k = k + 1$. If $k < n$, go to Step 2.

The five methods differ primarily in how the distance matrix is updated after two clusters have been joined. The IMETH option parameter specifies how the distance of the cluster just merged with each of the remaining clusters will be updated. Routine CLINK allows five methods of computing the distances. To understand these measures, suppose in the following discussion

that clusters "*A*" and "*B*" have just been joined to form cluster "*Z*", and interest is in computing the distance of *Z* with another cluster called "*C*".



IMETH **Method**

0 This is the single linkage method. The distance from *Z* to *C* is the minimum of the distances (*A* to *C*, *B* to *C*).

1 This is the complete linkage method. The distance from *Z* to *C* is the maximum of the distances (*A* to *C*, *B* to *C*).

2 This is the average-distance-within-clusters method. The distance from *Z* to *C* is the average distance of all objects that would be within the cluster formed by merging clusters *Z* and *C*. This average may be computed according to formulas given by Anderberg (1973, page 139).

3 This is the average-distance-between-clusters method. The distance from *Z* to *C* is the average distance of objects within cluster *Z* to objects within cluster *C*. This average may be computed according to methods given by Anderberg (1973, page 140).

4 This is Ward's method. Clusters are formed so as to minimize the increase in the within-cluster sums of squares. The distance between two clusters is the increase in these sums of squares if the two clusters were merged. A method for computing this distance from a squared Euclidean distance matrix is given by Anderberg (1973, pages 142–145).

In general, single linkage will yield long thin clusters while complete linkage will yield clusters that are more spherical. Average linkage and Ward's linkage tend to yield clusters that are similar to those obtained with complete linkage.

Routine CLINK produces a unique representation of the binary cluster tree via the following three conventions; the fact that the tree is unique should aid in interpreting the clusters. First, when two clusters are joined and each cluster contains two or more data points, the cluster that was initially formed with the smallest level (in CLEVEL) becomes the left son. Second, when a cluster containing more than one data point is joined with a cluster containing a single data point, the cluster with the single data point becomes the right son. Finally, when two clusters containing only one object are joined, the cluster with the smallest cluster number becomes the right son.

CNUMB

Computes cluster membership for a hierarchical cluster tree.

Required Arguments

NODE — Number of data points clustered. (Input)

ICLSON — Vector of length NODE − 1 containing the left son cluster numbers. (Input)
Cluster NODE + I is formed by merging clusters ICLSON(I) and ICRSON(I).

ICRSON — Vector of length NODE − 1 containing the right son cluster numbers. (Input)
Cluster NODE + I is formed by merging clusters ICLSON(I) and ICRSON(I).

K — Desired number of clusters. (Input)

ICLUS — Vector of length NODE containing the cluster membership of each observation. (Output)
Observation I is in cluster ICLUS(I) when K clusters are specified.

NCLUS — Vector of length K containing the number of observations in each cluster. (Output)

FORTRAN 90 Interface

Generic: `CALL CNUMB (NODE, ICLSON, ICRSON, K, ICLUS, NCLUS)`

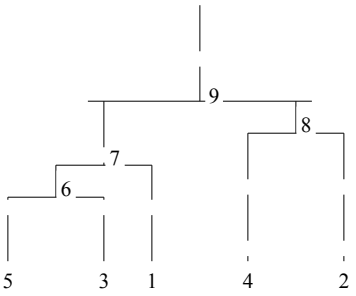
Specific: The specific interface name is CNUMB.

FORTRAN 77 Interface

Single: `CALL CNUMB (NODE, ICLSON, ICRSON, K, ICLUS, NCLUS)`

Example 1

In the following example, cluster membership for $K = 2$ clusters is found for the displayed cluster tree. The output vector ICLUS contains the cluster numbers for each observation.




```
      USE CNUMB_INT
      USE WRIRN_INT
      INTEGER    K, NODE
      PARAMETER  (K=2, NODE=5)
!
```

```
      INTEGER    ICLSON(NODE-1), ICLUS(NODE), ICRSON(NODE-1), NCLUS(K)
!
      DATA ICLSON/5, 6, 4, 7/
      DATA ICRSON/3, 1, 2, 8/
!                                  Compute cluster membership
      CALL CNUMB (NODE, ICLSON, ICRSON, K, ICLUS, NCLUS)
!                                  Print output
      CALL WRIRN ('ICLUS', ICLUS, 1, NODE, 1)
      CALL WRIRN ('NCLUS', NCLUS, 1, K, 1)
!
      END
```

Output

```
      ICLUS
1   2   3   4   5
1   2   1   2   1

      NCLUS
1   2
3   2
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `C2UMB`. The reference is:

   ```
   CALL C2UMB (NODE, ICLSON, ICRSON, K, ICLUS, NCLUS, IPT)
   ```

   The additional argument is:

   IPT — Work vector of length 2 * `NODE`.

2. Informational errors

Type	Code	
4	1	The tree structure specified by `ICLSON` and `ICRSON` is invalid because an attempt to assign an observation to more than one cluster is being made.
4	2	The tree structure specified by `ICLSON` and `ICRSON` is incorrect because an observation is not assigned to a cluster.

Description

Given a fixed number of clusters (*K*) and the cluster tree (vectors `ICRSON` and `ICLSON`) produced by the hierarchical clustering algorithm (see routine `CLINK`, page 1011), routine `CNUMB` determines the cluster membership of each observation. The routine `CNUMB` first determines the root nodes for the *K* distinct subtrees forming the *K* clusters and then traverses each subtree to determine the cluster membership of each observation. The routine `CNUMB` also returns the number of observations found in each cluster.

Example 2

This example illustrates the typical usage of CNUMB. The Fisher iris data (see routine GDATA, see Chapter 19, Utilities) is clustered. First the distance between the irises are computed using routine CDIST (page 1006). The resulting distance matrix is then clustered using routine CLINK (page 1011). The cluster membership for 5 clusters is then obtained via routine CNUMB using the output from CLINK. The need for 5 clusters can be obtained either by theoretical means or by examining a cluster tree. Because the cluster tree is too large to be included in this example, the call to routine TREEP (see Chapter 16, Line Printer Graphics) that would ordinarily print the cluster tree has been commented in the example code. The cluster membership for each of the iris observations is printed.

```
      USE IMSL_LIBRARIES
      INTEGER    IDATA, IPRINT, IROW, K, &
                 LDDIST, LDX, NCOL, NODE, NODEX, NROW, NVAR
      PARAMETER  (IDATA=3, IPRINT=0, IROW=1, K=5, LDDIST=150, &
                  LDX=150, NCOL=5, NODE=150, NODEX=5, NROW=150, NVAR=4)
!
      INTEGER    I, ICLSON(NROW-1), ICLUS(NODE), ICRSON(NROW-1), &
                 IND(4), J, NCLUS(K), NSCALE, NXCOL, NXROW
      REAL       AMAX1, CLEVEL(NROW-1), DIST(LDDIST,LDDIST), RN, &
                SCALE(2), X(LDX,NCOL)
      CHARACTER  NODENM(NODE)*7
      INTRINSIC  AMAX1
!
      DATA IND/2, 3, 4, 5/
      DATA NSCALE/1/
      DATA SCALE/0.0, 3.5/
!                                Get IRIS data.
      CALL GDATA (IDATA, X, NXROW, NXCOL)
!                                Compute the dissimilarities.
      CALL CDIST (X, DIST, IND=IND)
!                                Make sure each distance is unique,
!                                then copy the upper triangle matrix
!                                to the lower triangle matrix.
      CALL RNSET (4)
      DO 20  I=1, NODE
         DO 10  J=I + 1, NODE
      RN = RNUNF()
            DIST(I,J) = AMAX1(0.0,DIST(I,J)+(0.001*RN))
   10    CONTINUE
         DIST(I,I) = 0.0
         CALL SCOPY (I-1, DIST(1:,I), 1, DIST(I:,1), LDDIST)
   20 CONTINUE
!                                The initial clustering
      CALL CLINK (DIST, CLEVEL, ICLSON, ICRSON)
!                                Print the tree.
      NODENM(1) = 'DEFAULT'
!     CALL TREEP (ICLSON, ICRSON, CLEVEL, NSCALE, SCALE, NODENM)
!                                Compute membership for 5 clusters
      CALL CNUMB (NODE, ICLSON, ICRSON, K, ICLUS, NCLUS)
```

```
!                                    Print output
      CALL WRIRN ('ICLUS', ICLUS, 1, NODE, 1)
      CALL WRIRN ('NCLUS', NCLUS, 1, K, 1)
!
      END
```

Output

```
                                        ICLUS
1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
5  5  5  5  5  5  5  5  5   5   5   5   5   5   5   5   5   5   5   5

21 22 23 24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
 5  5  5  5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5

41 42 43 44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
 5  5  5  5   5   5   5   5   5   5   2   2   2   2   2   2   2   1   2   2

61 62 63 64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
 1  2  2  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2

81 82 83 84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99
 2  2  2  2   2   2   2   2   2   2   2   2   2   1   2   2   2   2   1

100 101 102 103  104  105  106  107  108  109  110  111  112  113  114  115
  2   2   2   2    2    2    2    3    2    2    2    2    2    2    2    2

116 117 118 119  120  121  122  123  124  125  126  127  128  129  130  131
  2   2   4   2    2    2    2    2    2    2    2    2    2    2    2    2

132 133 134 135  136  137  138  139  140  141  142  143  144  145  146  147
  4   2   2   2    2    2    2    2    2    2    2    2    2    2    2    2

148  149  150
  2    2    2

         NCLUS
1    2    3    4    5
4   93    1    2   50
```

KMEAN

Performs a *K*-means (centroid) cluster analysis.

Required Arguments

X — NOBS by NCOL matrix containing the observations to be clustered. (Input)
The only columns of X used are those indicated by IND and possibly IFRQ and/or IWT.

CM — K by NVAR matrix containing, on input, the cluster seeds, i.e., estimates for the cluster centers, and the cluster means on output. (Input/Output)
The cluster seeds must be unique.

SWT — K by NVAR matrix containing the sum of the weights used to compute each cluster mean. (Output)
Missing observations are excluded from SWT.

IC — Vector of length NOBS containing the cluster membership for each observation. (Output)

NC — Vector of length K containing the number of observations in each cluster. (Output)

WSS — Vector of length K containing the within sum of squares for each cluster. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

NVAR — Number of variables to be used in computing the metric. (Input)
Default: NVAR = size (CM,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)
IFRQ = 0 means all frequencies are 1. For positive IFRQ, column number IFRQ of X contains the nonnegative frequencies.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
IWT = 0 means all weights are 1. For positive IWT, column number IWT contains the nonnegative weights.
Default: IWT = 0.

IND — Vector of length NVAR containing the columns of X to be used in computing the metric. (Input)
In the usual case in which X is the data matrix, no observation has multiple frequency, and unequal weighting is not desired, IND = (1, 2, 3, …, NVAR).
By default, IND(I) = (I)

K — Number of clusters. (Input)
Default: K = size (CM,1).

MAXIT — Maximum number of iterations. (Input)
MAXIT = 30 is usually sufficient.
Default: MAXIT = 30.

LDCM — Leading dimension of CM exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCM = size (CM,1).

LDSWT — Leading dimension of SWT exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSWT = size (SWT,1).

FORTRAN 90 Interface

Generic: `CALL KMEAN (X, CM, SWT, IC, NC, WSS [,…])`

Specific: The specific interface names are `S_KMEAN` and `D_KMEAN`.

FORTRAN 77 Interface

Single:
```
CALL KMEAN (NOBS, NCOL, NVAR, X, LDX, IFRQ, IWT, IND, K,
MAXIT, CM, LDCM, SWT, LDSWT, IC, NC, WSS)
```

Double: The double precision name is `DKMEAN`.

Example

This example performs *K*-means cluster analysis on Fisher's iris data, which is first obtained via routine GDATA (see Chapter 19, Utilities). The initial cluster seed for each iris type is an observation known to be in the iris type.

```
      USE IMSL_LIBRARIES
      INTEGER    IPRINT, K, LDCM, LDSWT, LDX, NCOL, NOBS, NV, NVAR
      PARAMETER  (IPRINT=0, K=3, NCOL=5, NOBS=150, NV=5, NVAR=4, &
                  LDCM=K, LDSWT=K, LDX=NOBS)
!
      INTEGER    IC(NOBS), IND(NVAR), NC(K), NXCOL, NXROW
      REAL       CM(K,NVAR), SWT(K,NVAR), WSS(K), X(NOBS,NV)
!
      DATA IND/2, 3, 4, 5/
!
      CALL GDATA (3, X, NXROW, NXCOL)
!                                Copy the cluster seeds into CM
      CALL SCOPY (NVAR, X(1:,2), LDX, CM(1:,1), LDCM)
      CALL SCOPY (NVAR, X(51:,2), LDX, CM(2:,1), LDCM)
      CALL SCOPY (NVAR, X(101:,2), LDX, CM(3:,1), LDCM)
!
      CALL KMEAN (X, CM, SWT, IC, NC, WSS, IND=IND)
!
      CALL WRRRN ('CM', CM)
      CALL WRRRN ('SWT', SWT)
```

```
      CALL WRIRN ('IC', IC, 1, NOBS, 1)
      CALL WRIRN ('NC', NC, 1, K, 1)
      CALL WRRRN ('WSS', WSS, 1, K, 1)
      END
```

Output

```
                CM
         1       2       3       4
1    5.006   3.428   1.462   0.246
2    5.902   2.748   4.394   1.434
3    6.850   3.074   5.742   2.071

                SWT
         1       2       3       4
1    50.00   50.00   50.00   50.00
2    62.00   62.00   62.00   62.00
3    38.00   38.00   38.00   38.00

                                        IC
1  2  3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
1  1  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1

21 22 23 24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
 1  1  1  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1

41 42 43 44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
 1  1  1  1   1   1   1   1   1   1   2   2   3   2   2   2   2   2   2   2

61 62 63 64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
 2  2  2  2   2   2   2   2   2   2   2   2   2   2   2   2   2   3   2   2

81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99
 2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2

100 101 102 103  104  105  106  107  108  109  110  111  112  113  114  115
  2   3   2   3    3    3    3    2    3    3    3    3    3    3    2    2

116 117 118 119  120  121  122  123  124  125  126  127  128  129  130  131
  3   3   3   3    2    3    2    3    2    3    3    2    2    3    3    3

132 133 134 135  136  137  138  139  140  141  142  143  144  145  146  147
  3   3   2   3    3    3    3    2    3    3    3    2    3    3    3    2

148  149  150
  3    3    2

     NC
 1    2    3
50   62   38

          WSS
    1       2       3
15.15   39.82   23.88
```

Comments

1. Workspace may be explicitly provided, if desired, by use of K2EAN/DK2EAN. The reference is:

   ```
   CALL K2EAN (NOBS, NCOL, NVAR, X, LDX, IFRQ, IWT, IND, K, MAXIT,
   CM, LDCM, SWT, LDSWT, IC, NC, WSS, IC2, NCP, D, ITRAN, LIVE)
   ```

   The additional arguments are as follows:

   IC2 — Work vector of length NOBS.

   NCP — Work vector of length K.

   D — Work vector of length NOBS.

   ITRAN — Work vector of length K.

   LIVE — Work vector of length K.

2. Informational Error

Type	Code	
3	1	Convergence did not occur within MAXIT iterations.

Description

Routine KMEAN is an implementation of Algorithm AS 136 by Hartigan and Wong (1979). It computes *K*-means (centroid) Euclidean metric clusters for an input matrix starting with initial estimates of the *K* cluster means. Routine KMEAN allows for missing values (coded as NaN, "not a number") and for weights and frequencies.

Let p = NVAR denote the number of variables to be used in computing the Euclidean distance between observations. The idea in *K*-means cluster analysis is to find a clustering (or grouping) of the observations so as to minimize the total within-cluster sums of squares. In this case, the total sums of squares within each cluster is computed as the sum of the centered sum of squares over all nonmissing values of each variable. That is,

$$\phi = \sum_{i=1}^{K} \sum_{j=1}^{p} \sum_{m=1}^{n_i} f_{v_{im}} w_{v_{im}} \delta_{v_{im},j} \left(x_{v_{im},j} - \overline{x}_{ij} \right)^2$$

where v_{im} denotes the row index of the m-th observation in the i-th cluster in the matrix X; n_i is the number of rows of X assigned to group i; f denotes the frequency of the observation; w denotes its weight; δ is zero if the j-th variable on observation v_{im} is missing, otherwise δ is one; and

$$\overline{x}_{ij}$$

is the average of the nonmissing observations for variable j in group i. This method sequentially processes each observation and reassigns it to another cluster if doing so results in a decrease in

the total within-cluster sums of squares. The user in referred to Hartigan and Wong (1979) or Hartigan (1975) for the details.

Chapter 12: Sampling

Routines

Usage Notes

The routines for inferences regarding proportions require only counts as the input data. The other routines described in this chapter require the actual data. Since the amount of data may be quite large, these routines allow for the data to be input in small quantities (or even to be deleted after it has already been passed to the subroutine). This is accomplished by means of the processing option parameter, IDO, and an indicator of the number of observations being passed in, NROW. IDO has the following meaning:

IDO	Action
0	This is the only invocation of the subroutine for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to the subroutine will be made. Initialization and updating for the data are performed.
2	This is an intermediate invocation of the subroutine, and updating for the data is performed.
3	This is the final invocation of the routine. Updating for the data and any wrap-up computations are performed.

NROW can be positive or negative or zero. Its absolute value is the number of sample values being input. If NROW is negative, it is assumed that the observations being input have already been input once and now it is desired to delete them from the analysis. When IDO is 3, NROW can be set to 0.

In this case, only postprocessing is performed; no accumulation of statistics is done. This allows input of summary statistics rather than the actual data. See Example 2 in the documentation for the routine SMPSR (page 1048).

There are other variables used by several routines in this chapter that have a common meaning in all routines:

Y — The variable of interest.

X — The auxiliary variable.

NSAMP — The sample size.

NPOP — The population size.

CONPER — Confidence level.

STAT — Output statistics.

For stratified sampling, the following variables are often used:

NSTRAT — Number of strata.

NROWS — Vector with elements like NROW for strata.

NSAMPS — The strata sample sizes.

NPOPS — The population sizes for strata.

YBARS — The strata sample means.

YVARS — The strata sample variances.

SMPPR

Computes statistics for inferences regarding the population proportion and total given proportion data from a simple random sample.

Required Arguments

NINT — Number of sample units in the class of interest, for the population (or subpopulation) of interest. (Input)

NSAMP — Number of units in the entire random sample. (Input)

NPOP — Number of units in the population. (Input)

CONPER — Confidence level for two-sided interval estimates, in percent. (Input)
A CONPER percent confidence interval is computed; hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 – 2.0 * (100.0 – ONECL).

STAT — Vector of length 10 containing the resulting statistics. (Output)
These are:

I	STAT(I)
1	Estimate of the proportion.
2	Estimate of the total.
3	Variance estimate of the proportion estimate.
4	Variance estimate of the total estimate.
5	Lower confidence limit for the proportion.
6	Upper confidence limit for the proportion.
7	Lower confidence limit for the total.
8	Upper confidence limit for the total.
9	Estimate (expressed as a percentage) of the coefficient or variation of the total estimate. Not defined if NINT = 0.
10	Indicator of the distribution used to approximate the hypergeometric distribution for the confidence interval calculations. If STAT(10) = 0, then the normal is used. If STAT(10) = 1, then the Poisson is used. If STAT(10) = 2, then the binomial is used.

FORTRAN 90 Interface

Generic: `CALL SMPPR (NINT, NSAMP, NPOP, CONPER, STAT)`

Specific: The specific interface names are `S_SMPPR` and `D_SMPPR`.

FORTRAN 77 Interface

Single: `CALL SMPPR (NINT, NSAMP, NPOP, CONPER, STAT)`

Double: The double precision name is `DSMPPR`.

Example 1

The first example is from Cochran (1977, page 52). A simple random sample of size 200 was drawn from a list of 3042 names and addresses. Verification of the addresses in the sample showed 38 to be wrong. The objective is to estimate the total number of incorrect addresses.

```
USE UMACH_INT
USE SMPPR_INT

INTEGER    NINT, NOUT, NPOP, NSAMP
REAL       CONPER, SQRT, STAT(10), STDP, STDT
INTRINSIC  SQRT
```

```
!
      CALL UMACH (2, NOUT)
      NINT   = 38
      NSAMP  = 200
      NPOP   = 3042
      CONPER = 0.0
      CALL SMPPR (NINT, NSAMP, NPOP, CONPER, STAT)
      STDP = SQRT(STAT(3))
      STDT = SQRT(STAT(4))
      WRITE (NOUT,99999) STAT(1), STAT(2), STDP, STDT, STAT(9)
99999 FORMAT (' Estimate of proportion bad:              ', F5.3, /,&
            ' Estimate of total bad:                   ', F5.0, /, &
            ' Standard deviation estimate, proportion: ', F5.3, /, &
            ' Standard deviation estimate, total:      ', F5.1, /, &
            ' Coefficient of variation:                ', F5.1, &
            '%')
      END
```

Output

```
Estimate of proportion bad:              0.190
Estimate of total bad:                    578.
Standard deviation estimate, proportion: 0.027
Standard deviation estimate, total:       81.8
Coefficient of variation:                 14.1%
```

Description

The routine SMPPR computes point and interval estimates for the population proportion and total from a simple random sample. The simplest and most common case for which this routine is appropriate is one in which the population sampled contains two or more classes, and it is desired to estimate the proportion of the population falling into a particular class ("class of interest"). The data required by SMPPR consist of counts of the number of sample items in the class of interest, the sample size, and the population size. If there are more than two classes in the population, some of the classes may not be of interest.

Since the hypergeometric distribution is the appropriate probability model for the sampling for proportions in a finite population without replacement, exact confidence limits could be computed using that distribution. For populations with sizes that occur in practice (more than a hundred, often in the thousands or even millions), the confidence limits can be approximated very well by use of the normal, the binomial, or the Poisson distribution. Routine SMPPR uses one of these distributions in setting confidence limits, following the guidelines in the table on page 58 of Cochran (1977).

Additional Example

Example 2

The next example is also from Cochran (1977, page 68). A simple random sample of size 200 from 2000 colleges showed 120 colleges to be in favor of a certain proposal, 57 to be opposed, and 23 to have no opinion. We wish to estimate the number of colleges, out of the 2000, that favor the proposal.

```
      USE UMACH_INT
      USE SMPPR_INT
      INTEGER    NINT, NOUT, NPOP, NSAMP
      REAL       CONPER, STAT(10)
!
      CALL UMACH (2, NOUT)
      NINT   = 120
      NSAMP  = 200
      NPOP   = 2000
      CONPER = 95.0
      CALL SMPPR (NINT, NSAMP, NPOP, CONPER, STAT)
      WRITE (NOUT,99999) STAT(2), STAT(7), STAT(8)
99999 FORMAT (' Estimate of number in favor:   ', F5.0, /, ' 95% ', &
            'confidence interval: (', F5.0, ',', F5.0, ')')
      END
```

Output

```
Estimate of number in favor:   1200.
95% confidence interval: (1066.,1334.)
```

SMPPS

Computes statistics for inferences regarding the population proportion and total given proportion data from a stratified random sample.

Required Arguments

NINTS — Vector of length NSTRAT containing the observed number of units in each stratum from the class of interest. (Input)

NSAMPS — Vector of length NSTRAT containing the sample size in each stratum. (Input)

NPOPS — Vector of length NSTRAT containing the population in the strata. (Input)
If the population strata sizes are not known, estimates must be entered in their place.

PROPOR — Vector of length NSTRAT containing the within-strata proportion estimates. (Output)

STAT — Vector of length 10 containing the resulting statistics. (Output)
These are:

I	STAT(I)
1	Estimate of the proportion.
2	Estimate of the total.
3	Variance estimate of the proportion estimate.

I	STAT(I)
4	Variance estimate of the total estimate.
5	Lower confidence limit for the proportion.
6	Upper confidence limit for the proportion.
7	Lower confidence limit for the total.
8	Upper confidence limit for the total.
9	Estimate (expressed as a percentage) of the coefficient of variation of the total estimate.
10	Variance estimate of the proportion estimate assuming that sampling was simple random instead of stratified random.

Optional Arguments

NSTRAT — Number of strata into which the sample is divided. (Input)
In the vectors of length NSTRAT, the elements are all ordered in the same way.
Default: NSTRAT = size (NINTS,1).

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)
A CONPER percent confidence interval is computed; hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 – 2.0 * (100.0 – ONECL).
Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: `CALL SMPPS (NINTS, NSAMPS, NPOPS, PROPOR, STAT [,…])`

Specific: The specific interface names are `S_SMPPS` and `D_SMPPS`.

FORTRAN 77 Interface

Single:
```
CALL SMPPS (NSTRAT, NINTS, NSAMPS, NPOPS, CONPER, PROPOR,
STAT)
```

Double: The double precision name is `DSMPPS`.

Example

This example is an artificial modification of an example used in routine SMPPR (page 1024), which is from Cochran (1977, page 52). A list of 3042 names and addresses was built by an experienced secretary and a part-time student worker. The secretary entered 1838 names and addresses, and the student entered the remainder. Samples of size 100 were taken from the names entered by each. Verification of the addresses in the sample from the secretary's work showed 12 to be wrong, and verification of the student's sample showed 26 to be wrong. The objective is to estimate the total number of incorrect addresses.

```
      USE UMACH_INT
      USE SMPPS_INT
      INTEGER    NSTRAT
      PARAMETER  (NSTRAT=2)
!
      INTEGER    NINTS(NSTRAT), NOUT, NPOPS(NSTRAT), NSAMPS(NSTRAT)
      REAL       CONPER, PROPOR(NSTRAT), SQRT, STAT(10), STDP, STDSRS, &
                 STDT
      INTRINSIC  SQRT
!
      CALL UMACH (2, NOUT)
      NINTS(1)  = 12
      NINTS(2)  = 26
      NSAMPS(1) = 100
      NSAMPS(2) = 100
      NPOPS(1)  = 1838
      NPOPS(2)  = 1204
      CONPER    = 0.0
!
      CALL SMPPS (NINTS, NSAMPS, NPOPS, PROPOR, STAT, CONPER=CONPER)
!
      STDP   = SQRT(STAT(3))
      STDT   = SQRT(STAT(4))
      STDSRS = SQRT(STAT(10))
!
      WRITE (NOUT,99999) STAT(1), STAT(2), STDP, STDT, STAT(9), STDSRS
99999 FORMAT (' Estimate of proportion bad:              ', F7.3, /, &
            ' Estimate of total bad:                   ', F4.0, /, &
            ' Standard deviation estimate, proportion: ', F7.3, /, &
            ' Standard deviation estimate, total:      ', F5.1, /, &
            ' Coefficient of variation:                ', F5.1, &
            '%', /, ' Std. dev. under simple random sampling:  ', &
            F7.3)
      END
```

Output

```
Estimate of proportion bad:               0.175
Estimate of total bad:                  534.
Standard deviation estimate, proportion:  0.025
Standard deviation estimate, total:       77.4
Coefficient of variation:                 14.5%
Std. dev. under simple random sampling:   0.027
```

Description

Routine SMPPS computes point and interval estimates for the population proportion and total from a stratified random sample. If the strata are formed so that the proportions differ greatly from one stratum to the next, considerable gain in statistical efficiency can be realized by use of stratified sampling (see Cochran 1977, page 107).

Let N_h be the number in the population in the h-th stratum, let n_h be the number in the sample from the h-th stratum, let a_h be the number of the class of interest in the sample from the h-th stratum, let N be the population size ($\sum N_h$), let p_h be the proportion in the h-th stratum, a_h/n_h, and let L be the number of strata. Then, the estimate of the proportion is

$$p_{st} = \sum_{h=1}^{L} \frac{N_h a_h}{N n_h}$$

and the estimate of the variance is

$$v(p_{st}) = \frac{1}{N^2} \sum_{h=1}^{L} N_h (N_h - n_h) \frac{p_h (1 - p_h)}{n_h - 1}$$

The confidence intervals are computed using a normal approximation.

SMPRR

Computes statistics for inferences regarding the population mean and total using ratio or regression estimation, or inferences regarding the population ratio given a simple random sample.

Required Arguments

NROW — The absolute value of NROW is the number of observations currently input in X and Y. (Input)
NROW may be positive, zero, or negative. Negative –NROW means delete the NROW rows of data from the analysis.

X — Vector of length |NROW| containing the data for the auxiliary variable in the random sample. (Input)

Y — Vector of length |NROW| containing the data for the variable of interest in the random sample. (Input)
The value of Y(I) corresponds to that of X(I).

NPOP — Size of the population (number of pairs of elements in the sampled population). (Input)

XMEAN — Population mean of the auxiliary variable. (Input)
XMEAN is not used if IOPT = 1.

STAT — Vector of length 20 containing the resulting statistics. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)

I	STAT(I)
1	Estimate of the mean.
2	Estimate of the total.
3	Variance estimate of the mean estimate.
4	Variance estimate of the total estimate.
5	Lower confidence limit for the mean.
6	Upper confidence limit for the mean.
7	Lower confidence limit for the total.
8	Upper confidence limit for the total.
9	Estimate of the ratio.
10	Variance estimate of the estimate of the ratio. The population mean of the auxiliary variable is used in STAT(10) if the mean is known; otherwise, the sample estimate of the population mean is used.
11	Lower confidence limit for the ratio.
12	Upper confidence limit for the ratio.
13	Estimate (expressed as a percentage) of the coefficient of variation of the mean, total, and ratio and regression coefficient estimates that are defined, as controlled by IOPT. The standard deviation in the numerator of this quantity has been divided by the square root of the sample size. The coefficients of variation in STAT(14) and STAT(15) use the sample standard deviations without that divisor.
14	Estimate (expressed as a percentage) of the coefficient of variation of the auxiliary variable.
15	Estimate (expressed as a percentage) of the coefficient of variation of the variable of interest.
16	Sample mean of the auxiliary variable.
17	Sample mean of the variable of interest.
18	Estimate of the regression coefficient.
19	Sample size.

I	STAT(I)
20	Number of pairs in the sample with one or both values missing.

STAT(1) through STAT(8) and STAT(13) are undefined when IOPT = 1. STAT(9) through STAT(12) are undefined when IOPT = 2 or 3. STAT(18) is defined only when IOPT = 3. The elements of STAT that are undefined due to IOPT or an error are set to NaN (not a number).

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of SMPRR for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to SMPRR will be made. Initialization and updating for the data in X and Y are performed.
2	This is an intermediate invocation of SMPRR and updating for the data in X and Y is performed.
3	This is the final invocation of this routine. Updating for the data in X and Y, and wrap-up computations are performed.

IOPT — Estimation option. (Input)
Default: IOPT = 0.

IOPT	Action
0	Ratio estimation is used for inference about the population mean, total, and ratio.
1	The population mean of the auxiliary variable is not used, and only inference about the population ratio is desired.
2	Regression estimation with preassigned regression coefficient (in COEF) is used for inference about the population mean and total.
3	Regression estimation with estimated regression coefficient (returned in STAT(18)) is used for inference about the population mean and total.

COEF — Reassigned regression coefficient. (Input)
COEF is used only when IOPT = 2.
Default: COEF = 1.0.

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)
A CONPER percent confidence interval is computed, hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 – 2.0 * (100.0 – ONECL).
Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: `CALL SMPRR (NROW, X, Y, NPOP, XMEAN, STAT [,…])`

Specific: The specific interface names are S_SMPRR and D_SMPRR.

FORTRAN 77 Interface

Single:
```
CALL SMPRR (IDO, NROW, X, Y, NPOP, IOPT, XMEAN, COEF,
CONPER, STAT)
```

Double: The double precision name is DSMPRR.

Examples

The data for these examples come from Cochran (1977, Table 6.1, page 152). The variable of interest is the population of large U.S. cities in 1930; the auxiliary variable is the 1920 population of the same cities. There are 196 (NPOP) cities that are sampled (that is, that are in the population of interest). (Note that the word "population" is being used in two ways in this discussion.) The total 1920 population of these cities is 22,919 (XMEAN = 116.934). There are 49 cities in the sample. The data can be seen in the DATA statements in the programs below (actual values are 1000 times greater). There are no "missing data"; therefore, the sample size, STAT(19), is 49. Because the coefficient of variation is larger than 10%, SMPRR produces an informational "warning error" message in each example. When the coefficient of variation is larger than 10% (generally speaking), the confidence limits computed using the normal approximation are likely to be shorter than the actual limits at the same confidence level.

Example 1

In this example, ratio estimation is used, as on page 151 of Cochran (1977).

```
      USE SMPRR_INT
      USE UMACH_INT
      INTEGER    NROW
      PARAMETER  (NROW=49)
!
      INTEGER    I, NOUT, NPOP
      REAL       COEF, CONPER, STAT(20), X(NROW), XMEAN, Y(NROW)
!
      DATA X/76., 138., 67., 29., 381., 23., 37., 120., 61., 387., &
          93., 172., 78., 66., 60., 46., 2., 507., 179., 121., 50., &
          44., 77., 64., 64., 56., 40., 40., 38., 136., 116., 46., &
          243., 87., 30., 71., 256., 43., 25., 94., 43., 298., 36., &
```

```
          161., 74., 45., 36., 50., 48./
      DATA Y/80., 143., 67., 50., 464., 48., 63., 115., 69., 459., &
          104., 183., 106., 86., 57., 65., 50., 634., 260., 113., &
          64., 58., 89., 63., 77., 142., 60., 64., 52., 139., 130., &
          53., 291., 105., 111., 79., 288., 61., 57., 85., 50., 317., &
          46., 232., 93., 53., 54., 58., 75./
      DATA NPOP/196/, XMEAN/116.934/
!                                 All data are input at once.
!                                 Ratio estimation.
      CALL SMPRR (NROW, X, Y, NPOP, XMEAN, STAT)
!                                 Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (STAT(I),I=1,17), STAT(19), STAT(20)
99999 FORMAT (/, '                      RATIO ESTIMATION', /, &
            ' Mean estimate = ', F8.1, '      Total estimate = ', &
            F8.1, /, ' Vhat of mean  = ', F8.1, '      Vhat of total ' &
            , ' = ', F8.1, /, ' Confidence limits for mean  ', F8.1, &
            ',', F8.1, /, ' Confidence limits for total ', F8.1, &
            ',', F8.1, /, ' Ratio estimate = ', F8.3, '      Vhat of ' &
            , 'ratio = ', F8.4, /, ' Confidence limits for ratio ', &
            F8.3, ',', F8.3, /, ' Coefficient of variation of mean ', &
            'estimate = ', F8.1, /, ' CV of X =   ', F8.1, &
            '               CV of Y = ', F8.1, /, ' Mean of X = ', &
            F8.1, '            Mean of Y = ', F8.1, /, ' Sample size ' &
            , '= ', F8.1, '      Number missing = ', F8.1)
      END
```

Output

```
*** WARNING  ERROR 7 from SMPRR.  The coefficient of variation of one or
***          both of the variables exceeds 10%.  The confidence limits,
***          which are computed using a normal approximation, may not be
***          very accurate.

                   RATIO ESTIMATION
Mean estimate =    144.9      Total estimate =  28397.1
Vhat of mean  =      9.5      Vhat of total  = 364860.1
Confidence limits for mean     138.8,   150.9
Confidence limits for total  27213.3, 29581.0
Ratio estimate =    1.239      Vhat of ratio =   0.0007
Confidence limits for ratio    1.187,   1.291
Coefficient of variation of mean estimate =      2.1
CV of X =       89.3              CV of Y =     96.3
Mean of X =    103.1            Mean of Y =    127.8
Sample size =     49.0      Number missing =      0.0
```

Description

Routine SMPRR computes point and interval estimates for the population mean, total, and (optionally) ratio or regression coefficient, using a simple random sample of a variable of interest and an auxiliary variable. Routine SMPRR allows various options for the estimation techniques, which are discussed in Chapters 3, 6, and 7 of Cochran (1977). Let

$$\bar{x} \text{ and } \bar{y}$$

be the sample means of the auxiliary variable and the variable of interest, respectively. Let

$$\overline{X}$$

be the population mean of the auxiliary variable. Then, the ratio estimate of the population mean is

$$\overline{y}_R = \frac{\overline{y}}{\overline{x}}\overline{X}$$

The linear regression estimate of the population mean is

$$\overline{y}_{lr} = \overline{y} + b\left(\overline{X} - \overline{x}\right)$$

where b is the regression coefficient, which can be either preassigned, based on previous knowledge, or estimated from the data using least squares. The least-squares estimate of b is

$$\frac{\sum_{i=1}^{n}\left(y_i - \overline{y}\right)\left(x_i - \overline{x}\right)}{\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^2}$$

The confidence limits for the mean and for the total are computed using the normal approximation. If the coefficient of variation of either variable exceeds 10%, then this approximation may not be very accurate.

The parameters `IDO` and `NROW` allow either all or part of the data to be brought in.

Additional Examples

Example 2

In this example, regression estimation with an estimated coefficient is used, as in Exercise 7.3 of Cochran (1977).

```
      USE SMPRR_INT
      USE UMACH_INT
      INTEGER    NROW
      PARAMETER  (NROW=49)
!
      INTEGER    I, IOPT, NOUT, NPOP
      REAL       CONPER, STAT(20), X(NROW), XMEAN, Y(NROW)
!
      DATA X/76., 138., 67., 29., 381., 23., 37., 120., 61., 387., &
          93., 172., 78., 66., 60., 46., 2., 507., 179., 121., 50., &
          44., 77., 64., 64., 56., 40., 40., 38., 136., 116., 46., &
          243., 87., 30., 71., 256., 43., 25., 94., 43., 298., 36., &
          161., 74., 45., 36., 50., 48./
      DATA Y/80., 143., 67., 50., 464., 48., 63., 115., 69., 459., &
          104., 183., 106., 86., 57., 65., 50., 634., 260., 113., &
          64., 58., 89., 63., 77., 142., 60., 64., 52., 139., 130., &
          53., 291., 105., 111., 79., 288., 61., 57., 85., 50., 317., &
          46., 232., 93., 53., 54., 58., 75./
      DATA NPOP/196/, XMEAN/116.934/
!                                All data are input at once.
!                                Regression estimation, with estimated
!                                coefficient (Cochran, Exercise 7.3)
```

```
      IOPT = 3
      CALL SMPRR (NROW, X, Y, NPOP, XMEAN, STAT, IOPT=IOPT)
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (STAT(I),I=1,8), (STAT(I),I=13,20)
99999 FORMAT (/, '                    REGRESSION ESTIMATION', /, &
            ' Mean estimate = ', F8.1, '      Total estimate = ', &
            F8.1, /, ' Vhat of mean  = ', F8.1, '      Vhat of total ' &
            , ' = ', F8.1, /, ' Confidence limits for mean  ', F8.1, &
            ',', F8.1, /, ' Confidence limits for total ', F8.1, &
            ',', F8.1, /, ' Coefficient of variation of mean ', &
            'estimate = ', F8.1, /, ' CV of X =   ', F8.1, &
            '               CV of Y = ', F8.1, /, ' Mean of X = ', &
            F8.1, '           Mean of Y = ', F8.1, /, ' Estimated ', &
            'regression coefficient = ', F8.1, /, ' Sample size = ', &
            F8.1, '      Number missing = ', F8.1)
      END
```

Output

```
*** WARNING  ERROR 7 from SMPRR.  The coefficient of variation of one or
***          both of the variables exceeds 10%.  The confidence limits,
***          which are computed using a normal approximation, may not be
***          very accurate.

                     REGRESSION ESTIMATION
Mean estimate =    143.8      Total estimate =  28177.4
Vhat of mean  =      8.6      Vhat of total  = 329372.3
Confidence limits for mean     138.0,   149.5
Confidence limits for total  27052.6, 29302.3
Coefficient of variation of mean estimate =      2.0
CV of X =       89.3              CV of Y =     96.3
Mean of X =    103.1           Mean of Y =    127.8
Estimated regression coefficient =      1.2
Sample size =     49.0      Number missing =      0.0
```

Example 3

In this example, regression estimation with a preassigned coefficient is used, as in Exercise 7.4 of Cochran (1977).

```
      USE SMPRR_INT
      USE UMACH_INT

      INTEGER    NROW
      PARAMETER  (NROW=49)
!
      INTEGER    I, IOPT, NOUT, NPOP
      REAL       COEF, STAT(20), X(NROW), XMEAN, Y(NROW)
!
      DATA X/76., 138., 67., 29., 381., 23., 37., 120., 61., 387., &
          93., 172., 78., 66., 60., 46., 2., 507., 179., 121., 50., &
          44., 77., 64., 64., 56., 40., 40., 38., 136., 116., 46., &
          243., 87., 30., 71., 256., 43., 25., 94., 43., 298., 36., &
          161., 74., 45., 36., 50., 48./
      DATA Y/80., 143., 67., 50., 464., 48., 63., 115., 69., 459., &
          104., 183., 106., 86., 57., 65., 50., 634., 260., 113., &
```

```
          64., 58., 89., 63., 77., 142., 60., 64., 52., 139., 130., &
          53., 291., 105., 111., 79., 288., 61., 57., 85., 50., 317., &
          46., 232., 93., 53., 54., 58., 75./
      DATA NPOP/196/, XMEAN/116.934/
!                                All data are input at once.
!                                Regression estimation, with assigned
!                                coefficient (Cochran, Exercise 7.4)
      IOPT = 2
      COEF = 1.0
      CALL SMPRR (NROW, X, Y, NPOP, XMEAN, STAT, IOPT=IOPT, COEF=COEF)
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (STAT(I),I=1,8), (STAT(I),I=13,17), STAT(19), &
                         STAT(20)
99999 FORMAT (/, '                    REGRESSION ESTIMATION, FIXED ', &
            'COEF', /, ' Mean estimate = ', F8.1, '      Total ', &
            'estimate = ', F8.1, /, ' Vhat of mean  = ', F8.1, &
            '      Vhat of total  = ', F8.1, /, ' Confidence limits ' &
            , 'for mean  ', F8.1, ',', F8.1, /, ' Confidence limits ' &
            , 'for total ', F8.1, ',', F8.1, /, ' Coefficient of ', &
            'variation of mean estimate = ', F8.1, /, ' CV of X =   ' &
            , F8.1, '              CV of Y = ', F8.1, /, ' Mean of ' &
            , 'X = ', F8.1, '            Mean of Y = ', F8.1, /, &
            ' Sample size = ', F8.1, '      Number missing = ', F8.1)
      END
```

Output

```
*** WARNING  ERROR 7 from SMPRR.  The coefficient of variation of one or
***          both of the variables exceeds 10%.  The confidence limits,
***          which are computed using a normal approximation, may not be
***          very accurate.

       REGRESSION ESTIMATION, FIXED COEF
Mean estimate =    141.6      Total estimate =  27751.1
Vhat of mean  =     12.5      Vhat of total  = 481977.4
Confidence limits for mean     134.6,   148.5
Confidence limits for total  26390.4, 29111.8
Coefficient of variation of mean estimate =      2.5
CV of X =       89.3              CV of Y =     96.3
Mean of X =    103.1            Mean of Y =    127.8
Sample size =     49.0      Number missing =      0.0
```

SMPRS

Computes statistics for inferences regarding the population mean and total using ratio or regression estimation given continuous data from a stratified random sample.

Required Arguments

NROWS — Vector of length NSTRAT in which |NROWS(I)| is the number of items from the I-th stratum currently input in X and Y. (Input)
Each element of NROWS may be positive, zero, or negative. A negative value for

NROWS(I) means delete the –NROWS(I) elements of the I-th stratum in X and Y from the analysis.

X — Vector containing the data for the auxiliary variable in the stratified random sample. (Input)
The observations within any one stratum must appear contiguously in X. The first |NROWS(1)| elements of X are from the first stratum, and so on.

Y — Vector containing the data for the variable of interest in the stratified random sample. (Input)
The observations within any one stratum must appear contiguously in Y. The first |NROWS(1)| elements of Y are from the first stratum, and so on. The value of Y(I) corresponds to that of X(I).

NPOPS — Vector of length NSTRAT containing the sizes of the population in the strata. (Input)
The entries in NSTRAT must be ordered in correspondence with the ordering of strata in the other vectors. If the population strata sizes are not known, estimates must be entered in their place.

XMEANS — Vector of length NSTRAT containing, for each stratum, the population mean of the auxiliary variate, provided ITOPT = 0. (Input)
If ITOPT = 1, only XMEANS(1) is defined and it must contain the population mean of the auxiliary variate.

COEFS — Vector of length NSTRAT containing the ratio estimates or the regression coefficients. (Input, if IOPT = 1; output, if IOPT = 0 or 2 and IDO = 0 or 1; input/output, if IOPT = 0 or 2 and IDO = 2 or 3)
If IOPT = 0, COEFS contains ratio estimates. When ITOPT = 0, COEFS contains the estimate of the ratio for each stratum. When ITOPT = 1, only COEFS(1) is defined and contains the combined estimate of the ratio. If IOPT = 1, COEFS contains preassigned regression coefficients. When ITOPT = 0, COEFS contains the preassigned regression coefficient for each stratum. When ITOPT = 1, only COEFS(1) is defined and contains the preassigned regression coefficient common to all strata. If IOPT = 2, COEFS contains estimated regression coefficients. When ITOPT = 0, COEFS contains the estimated regression coefficient for each stratum. When ITOPT = 1, only COEFS(1) is defined and contains the estimated regression coefficient common to all strata.

XBARS — Vector of length NSTRAT containing the strata means for the auxiliary variable. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)

XVARS — Vector of length NSTRAT containing the within-strata variances of the auxiliary variable. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)

XCVS — Vector of length NSTRAT containing the within-strata coefficients of variation for the auxiliary variable. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)

YBARS — Vector of length NSTRAT containing the strata means for the variable of interest. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)

YVARS — Vector of length NSTRAT containing the within-strata variances of the variable of interest. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)

YCVS — Vector of length NSTRAT containing the within-strata coefficients of variation for the variable of interest. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)

XYCOVS — Vector of length NSTRAT containing the within-strata covariances of the auxiliary variable and the variable of interest. (Output, if IDO = 0 or 1; input/ output, if IDO = 2 or 3.)

NSAMPS — Vector of length NSTRAT containing the number of nonmissing observations from each stratum. (Output, if IDO= 0 or 1; input/output, if IDO = 2 or 3.)

STAT — Vector of length 12 containing the resulting statistics. (Output)
These are:

I	STAT(I)
1	Estimate of the mean.
2	Estimate of the total.
3	Variance estimate of the mean estimate.
4	Variance estimate of the total estimate.
5	Lower confidence limit for the mean.
6	Upper confidence limit for the mean.
7	Lower confidence limit for the total.
8	Upper confidence limit for the total.
9	Estimate of the coefficient of variation for the mean and total estimate.
10	Unstratified mean of the auxiliary variate.
11	Unstratified mean of the variable of interest.
12	The number of pairs in the sample that had one or both values missing.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of SMPRS for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to SMPRS will be made. Initialization and updating for the data in X and Y are performed.
2	This is an intermediate invocation of SMPRS, and updating for the data in X and Y is performed.
3	This is the final invocation of this routine. Updating for the data in X and Y and wrap-up computations are performed.

NSTRAT — Number of strata into which the population is divided. (Input)
In the vectors of length NSTRAT, the elements are all ordered in the same way. That is, the first stratum is always the first, the second is always the second, and so on.
Default: NSTRAT = size (NROWS,1).

IOPT — Estimation option. (Input)
Default: IOPT = 0.

IOPT	Action
0	Ratio estimation used for inference about the population mean and total.
1	Regression estimation used with the preassigned regression coefficient(s) contained in COEFS.
2	Regression estimation used with the regression coefficient(s) estimated from the data.

ITOPT — Estimation technique option. (Input)
Default: ITOPT = 0.

ITOPT	Action
0	Separate ratio or regression estimation.
1	Combined ratio or regression estimation.

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)
A CONPER percent confidence interval is computed; hence, CONPER must be greater

than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 – 2.0 * (100.0 – ONECL).
Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: `CALL SMPRS (NROWS, X, Y, NPOPS, XMEANS, COEFS, XBARS, XVARS, XCVS, YBARS, YVARS, YCVS, XYCOVS, NSAMPS, STAT [,…])`

Specific: The specific interface names are S_SMPRS and D_SMPRS.

FORTRAN 77 Interface

Single:
```
CALL SMPRS (IDO, NSTRAT, NROWS, X, Y, NPOPS, IOPT, ITOPT,
XMEANS, CONPER, COEFS, XBARS, XVARS, XCVS,
YBARS, YVARS, YCVS, XYCOVS, NSAMPS, STAT)
```

Double: The double precision name is DSMPRS.

Example 1

In the following example, we use a stratified sample from the data in Table 5.1 of Cochran (1977), which consists of the 1920 and the 1930 population (in 1000's) of 64 cities in the United States. The objective is to estimate the mean and total 1930 population of the 64 cities, using a sample of size 24 of the 1920 and 1930 populations. There are two strata: the largest 16 cities and the remaining cities. We use stratified sampling with equal sample sizes. The same example is also used to illustrate routine SMPSS (page 1052), except here we have an auxiliary variable.

In this example, separate ratio estimation is used.

```
      USE SMPRS_INT
      USE UMACH_INT
      INTEGER    NSTRAT
      PARAMETER  (NSTRAT=2)
!
      INTEGER    I, NOUT, NPOPS(NSTRAT), NROWS(NSTRAT), NSAMPS(NSTRAT)
      REAL       COEFS(NSTRAT), STAT(12), X(24), &
                 XBARS(NSTRAT), XCVS(NSTRAT), XMEANS(NSTRAT), &
                 XVARS(NSTRAT), XYCOVS(NSTRAT), Y(24), YBARS(NSTRAT), &
                 YCVS(NSTRAT), YVARS(NSTRAT)
!
      DATA X/773., 748., 734., 577., 507., 438., 415., 401., 387., &
          381., 324., 315., 258., 237., 235., 216., 201., 179., 136., &
          132., 118., 118., 106., 104./
      DATA Y/822., 781., 805., 1238., 634., 487., 442., 451., 459., &
          464., 400., 366., 302., 291., 272., 284., 270., 260., 139., &
          170., 154., 140., 163., 116./
!
      NPOPS(1) = 16
      NPOPS(2) = 48
```

```
!                                  All data are input at once.
      NROWS(1) = 12
      NROWS(2) = 12
!                                  Use separate ratio estimation.
      XMEANS(1) = 521.8
      XMEANS(2) = 165.4
!
      CALL SMPRS (NROWS, X, Y, NPOPS, XMEANS, COEFS, XBARS, XVARS, &
                  XCVS, YBARS, YVARS, YCVS, XYCOVS, NSAMPS, STAT)
!                                  Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (STAT(I),I=1,9), STAT(12), COEFS
99999 FORMAT (' Mean estimate = ', F8.3, '      Total estimate = ', &
            F8.1, /, ' Vhat of mean  = ', F8.5, '      Vhat of total ' &
            , ' = ', F8.1, /, ' Confidence limits for mean  ', F8.3, &
            ',', F8.3, /, ' Confidence limits for total ', F8.1, &
            ',', F8.1, /, ' C. V.       = ', F8.2, '      Number ', &
            'missing = ', F8.1, /, ' Estimated ratios = ', 2F10.3)
      END
```

Output

```
Mean estimate =  315.511      Total estimate =  20192.7
Vhat of mean  = 55.56254      Vhat of total  = 227584.2
Confidence limits for mean   300.901, 330.120
Confidence limits for total  19257.7, 21127.7
C. V.       =     2.36      Number missing =      0.0
Estimated ratios =      1.225     1.255
```

Description

Routine SMPRS computes point and interval estimates for the population mean and total from a stratified random sample of a variable of interest and an auxiliary variable. Routine SMPRS allows for either ratio estimation, regression estimation with preassigned coefficients, or regression estimation with estimated coefficients.

This routine follows the standard methods discussed in Chapters 6 and 7 of Cochran (1977). The statistics are similar to those discussed in the documentation for routine SMPRR (page 1030), except that they are computed from stratified data. The option parameter IOPT allows selection of either ratio or regression estimation, and the parameter ITOPT allows selection of separate or combined estimators. "Separate" estimators means that each stratum is allowed to have different ratios or regression coefficients, while "combined" means these are assumed to be the same over all strata.

The confidence limits for the mean and for the total are computed using the normal approximation. If the coefficient of variation of either variable exceeds 10%, then this approximation may not be very accurate.

The parameters IDO and NROW allow either all or part of the data to be brought in at one time.

Example 2

In the following example, we use a stratified sample from the data in Table 5.1 of Cochran (1977), which consists of the 1920 and the 1930 population (in 1000's) of 64 cities in the United

States. The objective is to estimate the mean and total 1930 population of the 64 cities, using a sample of size 24 of the 1920 and 1930 populations. There are two strata: the largest 16 cities and the remaining cities. We use stratified sampling with equal sample sizes. The same example is also used to illustrate routine SMPSS (page 1052), except here we have an auxiliary variable.

In this example, regression estimation is used, and it is assumed that the regression equation is the same in the two strata.

```
      USE SMPRS_INT
      USE UMACH_INT
      INTEGER    NSTRAT
      PARAMETER  (NSTRAT=2)
!
      INTEGER    I, IDO, IOPT, ITOPT, NOUT, NPOPS(NSTRAT), &
                 NROWS(NSTRAT), NSAMPS(NSTRAT)
      REAL       COEFS(NSTRAT), STAT(12), X(24), &
                 XBARS(NSTRAT), XCVS(NSTRAT), XMEANS(1), &
                 XVARS(NSTRAT), XYCOVS(NSTRAT), Y(24), YBARS(NSTRAT), &
                 YCVS(NSTRAT), YVARS(NSTRAT)
!
      DATA X/773., 748., 734., 577., 507., 438., 415., 401., 387., &
          381., 324., 315., 258., 237., 235., 216., 201., 179., 136., &
          132., 118., 118., 106., 104./
      DATA Y/822., 781., 805., 1238., 634., 487., 442., 451., 459., &
          464., 400., 366., 302., 291., 272., 284., 270., 260., 139., &
          170., 154., 140., 163., 116./
!
      NPOPS(1) = 16
      NPOPS(2) = 48
!                                  All data are input at once.
      NROWS(1) = 12
      NROWS(2) = 12
!                                  Use combined regression estimation.
      IOPT      = 2
      ITOPT     = 1
      XMEANS(1) = 254.5
!
      CALL SMPRS (NROWS, X, Y, NPOPS, XMEANS, COEFS, XBARS, XVARS, &
                  XCVS, YBARS, YVARS, YCVS, XYCOVS, NSAMPS, STAT, &
                  IOPT=IOPT, ITOPT=ITOPT)
!                                  Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (STAT(I),I=1,9), STAT(12), COEFS(1)
99999 FORMAT (' Mean estimate = ', F8.3, '      Total estimate = ', &
            F8.1, /, ' Vhat of mean  = ', F8.5, '      Vhat of total ' &
            , ' = ', F8.1, /, ' Confidence limits for mean  ', F8.3, &
            ',', F8.3, /, ' Confidence limits for total ', F8.1, &
            ',', F8.1, /, ' C. V.       = ', F8.1, '      Number ', &
            'missing = ', F8.1, /, ' Estimated combined regression ', &
            'coefficient = ', F8.3)
      END
```

Output

```
 Mean estimate =  315.517      Total estimate =  20193.1
Vhat of mean  = 54.84098      Vhat of total  = 224628.6
```

```
Confidence limits for mean   301.003, 330.031
Confidence limits for total  19264.2, 21122.0
C. V.       =      2.3      Number missing =      0.0
Estimated combined regression coefficient =    1.175
```

SMPSC

Computes statistics for inferences regarding the population mean and total using single stage cluster sampling with continuous data.

Required Arguments

NROWS — Vector of length NCLSTR in which |NROWS(I)| is the number of items from the I-th cluster currently input in Y. (Input)
Each element of NROWS may be positive, zero, or negative. A negative value for NROWS(I) means delete the –NROWS(I) elements of the I-th cluster in Y from the analysis.

Y — Vector containing the cluster sample. (Input)
The observations within any one cluster must appear contiguously in Y. The first |NROWS(1)| elements of Y are from the first cluster, and so on.

NCLPOP — Number of clusters in the sampled population. (Input)

NPOP — Number of elements in the population (sum of all the cluster sizes in the population). (Input)
NPOP is not required when IOPT = 3.

CLMEAN — Vector of length NCLSTR containing the cluster means. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)

CLVAR — Vector of length NCLSTR containing the within-cluster variances. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)

NSAMPS — Vector of length NCLSTR containing the number of nonmissing observations from each cluster. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)

STAT — Vector of length 11 containing the resulting statistics. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)
These are:

I	STAT(I)
1	Estimate of the mean.
2	Estimate of the total.
3	Variance estimate of the mean estimate.

I	STAT(I)
4	Variance estimate of the total estimate.
5	Lower confidence limit for the mean.
6	Upper confidence limit for the mean.
7	Lower confidence limit for the total.
8	Upper confidence limit for the total.
9	Estimate (expressed as a percentage) of the coefficient of variation of the mean and total estimate.
10	The total sample size.
11	The number of missing values.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of SMPSC for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to SMPSC will be made. Initialization and updating for the data in Y are performed.
2	This is an intermediate invocation of SMPSC and updating for the data in Y is performed.
3	This is the final invocation of this routine. Updating for the data in Y and wrap-up computations are performed.

NCLSTR — Number of clusters into which the sample is divided. (Input)
In the vectors of length NCLSTR, the elements are all ordered in the same way. That is, the first cluster is always the first, the second always the second, and so on.
Default: NCLSTR = size (NROWS,1).

IOPT — Estimation option. (Input)
Default: IOPT = 0.

IOPT	Action
0	Ratio-to-size estimation is used.
1	Unbiased estimation is used.
2	Probability-proportional-to-size estimation is used and all clusters in population are of known size.
3	Probability-proportional-to-size estimation is used and the cluster sizes are known only approximately or a measure of cluster size other than the number of elements per cluster is to be used.

SIZE — If IOPT = 3, vector of length NCLSTR containing a measure of cluster size for each cluster in the sample. (Input)
The sampled cluster size measures must be ordered in correspondence with the ordering of clusters in Y. SIZE is required only when IOPT = 3.

TSIZE — If IOPT = 3, measure of total size of all clusters in the population. (Input) TSIZE is required only when IOPT = 3.
Default: TSIZE = 1.0.

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)
A CONPER percent confidence interval is computed; hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 – 2.0 * (100.0 – ONECL).
Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: `CALL SMPSC (NROWS, Y, NCLPOP, NPOP, CLMEAN, CLVAR, NSAMPS, STAT [,…])`

Specific: The specific interface names are S_SMPSC and D_SMPSC.

FORTRAN 77 Interface

Single:
```
CALL SMPSC (IDO, NCLSTR, NROWS, Y, IOPT, NCLPOP, NPOP,
SIZE, TSIZE, CONPER, CLMEAN, CLVAR, NSAMPS, STAT)
```

Double: The double precision name is DSMPSC.

Example

In this example, we have a sample of two clusters from a population that contains 20 clusters. The sizes of the clusters in the sample are four and six, and there is a total of 100 elements in the population.

```
      USE SMPSC_INT
      USE UMACH_INT
      INTEGER    NCLSTR
      PARAMETER  (NCLSTR=2)
!
      INTEGER    NCLPOP, NOUT, NPOP, NROWS(NCLSTR), NSAMPS(NCLSTR)
      REAL       CLMEAN(NCLSTR), CLVAR(NCLSTR), SIZE(NCLSTR), &
                 STAT(11), TSIZE, Y(10)
!
      DATA Y/2.7, 5.1, 4.3, 2.8, 1.9, 6.2, 4.8, 5.1, 7.2, 6.5/
!
      NCLPOP = 20
      NPOP   = 100
!                                  All data are input at once.
      NROWS(1) = 4
      NROWS(2) = 6
      CALL SMPSC (NROWS, Y, NCLPOP, NPOP, CLMEAN, CLVAR, NSAMPS, STAT)
!                                  Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STAT
99999 FORMAT (' Mean estimate = ', F8.3, '      Total estimate = ', &
             F8.1, /, ' Vhat of mean  = ', F8.3, '      Vhat of total ' &
             , ' = ', F8.1, /, ' Confidence limits for mean  ', F8.3, &
             ',', F8.3, /, ' Confidence limits for total ', F8.1, &
             ',', F8.1, /, ' C. V.       = ', F8.1, '%', /, &
             ' Sample size =   ', F8.0, '      Number missing = ', &
             F8.0)
      END
```

Output

```
Mean estimate =    4.660      Total estimate =    466.0
Vhat of mean  =    0.504      Vhat of total  =   5035.5
Confidence limits for mean     3.269,   6.051
Confidence limits for total    326.9,   605.1
C. V.       =     15.2%
Sample size =        10.      Number missing =       0.
```

Description

Routine SMPSC computes point and interval estimates for the population mean and total from a single-stage cluster sample. The routine uses the standard methods discussed in Chapters 9 and 9A of Cochran (1977). The sample means for the individual clusters are accumulated in CLMEAN, and the corrected sums of squares are accumulated in CLVAR. In the postprocessing phase, the quantities in STAT are computed using the cluster statistics in CLMEAN, CLVAR, and NSAMPS. The parameters IDO and NROWS allow either all or part of the data to be brought in at one time.

Following the notation of Cochran (1977), let N be the number of clusters in the population, let M_i be the number of elements in the i-th cluster unit, let M_0 be the total number of elements in the population, let y_{ij} be the j-th element in the i-th cluster, and let n be the number of clusters in the sample. Any of three different estimators of the population total may be useful. An unbiased estimate of the total is

$$\frac{N}{n}\sum_{i=1}^{n}\sum_{j=1}^{M_i} y_{ij}$$

The ratio-to-size estimate is

$$M_0 \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} M_i}$$

The probability-proportional-to-size estimate is

$$\frac{M_0}{n}\sum_{i=1}^{n}\left(\frac{y_i}{M_i}\right)$$

The confidence limits for the mean and total are computed using the normal approximation.

SMPSR

Computes statistics for inferences regarding the population mean and total, given data from a simple random sample.

Required Arguments

NROW — The absolute value of NROW is the number of rows of data currently input in Y. (Input)
NROW may be positive, zero, or negative. Negative –NROW means delete the NROW rows of data from the analysis.

Y — Vector of length |NROW| containing the sample data. (Input)

NPOP — Size of the (full) population. (Input)

STAT — Vector of length 11 containing the resulting statistics. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)
These are:

I	STAT(I)
1	Estimate of the mean.
2	Estimate of the total.
3	Within-sample variance estimate.
4	Variance estimate of the mean estimate.
5	Variance estimate of the total estimate.

I	STAT(I)
6	Lower confidence limit for the mean.
7	Upper confidence limit for the mean.
8	Lower confidence limit for the total.
9	Upper confidence limit for the total.
10	The sample size.
11	The number of missing values.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of SMPSR for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to SMPSR will be made. Initialization and updating for the data in Y are performed.
2	This is an intermediate invocation of SMPSR, and updating for the data in Y is performed.
3	This is the final invocation of this routine. Updating for the data in Y and wrap-up computations are performed.

IOPT — Subpopulation option. (Input)
If IOPT = 0, no subpopulation is assumed. If IOPT = 1, the input data come from a subpopulation ("domain of study") of unknown size.
Default: IOPT = 0.

NSAMPO — Size of the sample from the full population, if a subpopulation is sampled (that is, if IOPT = 1). (Input)
Default: NSAMPO = ABS (NROW).

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)
A CONPER percent confidence interval is computed; hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 – 2.0 *

(100.0 – ONECL).
Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: `CALL SMPSR (NROW, Y, NPOP, STAT [,…])`

Specific: The specific interface names are S_SMPSR and D_SMPSR.

FORTRAN 77 Interface

Single: `CALL SMPSR (IDO, NROW, Y, NPOP, IOPT, NSAMPO, CONPER, STAT)`

Double: The double precision name is DSMPSR.

Example 1

This example uses artificial data to illustrate a simple use of SMPSR to compute point and interval estimates of the population mean and total. The sample size is 15, from a population of size 150.

```
      USE SMPSR_INT
      USE UMACH_INT
      INTEGER    NROW
      PARAMETER  (NROW=15)
!
      INTEGER    NOUT, NPOP, NSAMPO
      REAL       STAT(11), Y(NROW)
!
      DATA Y/21., 14., 17., 22., 19., 21., 20., 15., 24., 28., 20., &
          17., 16., 22., 19./
!
      NPOP   = 150
!                                All data are input at once.
!                                No subpopulation is assumed.
      CALL SMPSR (NROW, Y, NPOP, STAT)
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STAT
99999 FORMAT (' Mean estimate = ', F8.3, '      Total estimate = ', &
            F8.1, /, ' Within-sample variance estimate = ', F8.3, /, &
            ' VHAT of mean  = ', F8.5, '      VHAT of total  = ', &
            F8.1, /, ' Confidence limits for mean  ', F8.3, &
            ',', F8.3, /, ' Confidence limits for total ', F8.1, &
            ',', F8.1, /, ' Sample size = ', F8.1, '      Number ', &
            'missing = ', F8.0)
      END
```

Output

```
Mean estimate =   19.667      Total estimate =   2950.0
Within-sample variance estimate =   13.238
VHAT of mean  =  0.79429      VHAT of total  =  17871.4
Confidence limits for mean    17.755,  21.578
Confidence limits for total   2663.3,  3236.7
Sample size =     15.0      Number missing =       0.
```

Description

Routine SMPSR computes point and interval estimates for the population mean and total from a simple

random sample of one variable. The routine uses the standard methods discussed in Chapter 2 of Cochran (1977). The sample mean is accumulated in STAT(1) and the corrected sum of squares is accumulated in STAT(3). In the postprocessing phase, STAT(3) is divided by the sample size minus one, and then the other quantities in STAT are computed. The parameters IDO and NROW allow either all or part of the data to be brought in at one time.

By use of IOPT and NSAMPO, SMPSR can also be used to analyze data from a subpopulation or "domain of study". (See Cochran 1977, pages 34–38.) In the case of a subpopulation, only the estimates relating to the subpopulation total differ from the corresponding estimates when no subpopulation is assumed. Of course, if a subpopulation is of known size, it should be considered the full population.

Additional Example

Example 2

This example is a problem of estimation in a subpopulation described on page 37 of Cochran (1977). The example illustrates how the IDO and NROW parameters can be used to allow input other than the actual data. Cochran gives only the sample total and uncorrected sum of squares, so these values are transformed to the mean and corrected sum of squares prior to input as STAT(1) and STAT(3).

```
      USE SMPSR_INT
      USE UMACH_INT
      INTEGER    IDO, IOPT, NOUT, NPOP, NROW, NSAMPO
      REAL       SQRT, STAT(11), Y(1)
      INTRINSIC  SQRT
!
      NPOP   = 2422
!                                There are 180 items in the complete
!                                sample, but only a subpopulation is
!                                of interest.
      IOPT   = 1
      NSAMPO = 180
!                                For this example, STAT is
!                                initialized as if the data
!                                have been already processed and only
!                                the postprocessing computations are
!                                to be done.  There are 152 items of
!                                interest in the sample.  The sample
```

```
!                                  total is 343.5 and the uncorrected
!                                  sum of squares is 1491.38.
!                                  STAT(1) is initialized to the sample
!                                  mean by dividing the total by the
!                                  sample size, and STAT(3) is
!                                  initialized to the corrected sum of
!                                  squares.
      STAT(1)  = 343.5/152.0
      STAT(3)  = 1491.38 - 152.0*STAT(1)**2
      STAT(10) = 152.0
      STAT(11) = 0.0
      IDO      = 3
      NROW     = 0
      CALL SMPSR (NROW, Y, NPOP, STAT, IDO=IDO, IOPT=IOPT, NSAMPO=NSAMPO)
!                                  Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STAT(2), SQRT(STAT(5))
99999 FORMAT ('      Total estimate = ', F8.1, /, '      Standard ', &
            'deviation of the estimate = ', F8.1)
      END
```

Output

```
Total estimate =   4622.0
Standard deviation of the estimate =    375.3
```

SMPSS

Computes statistics for inferences regarding the population mean and total, given data from a stratified random sample.

Required Arguments

NROWS — Vector of length NSTRAT in which |NROWS(I)| is the number of items from the I-th stratum currently input in Y. (Input)
Each element of NROWS may be positive, zero, or negative. A negative value for NROWS(I) means delete the –NROWS(I) elements of the I-th stratum in Y from the analysis.

Y — Vector containing the stratified random sample. (Input)
The observations within any one stratum must appear contiguously in Y. The first |NROWS(1)| elements of Y are from the first stratum, and so on.

NPOPS — Vector of length NSTRAT containing the sizes of the population in the strata. (Input)
The entries must be ordered in correspondence with the ordering of strata in the other vectors. If the population strata sizes are not known, estimates must be entered in their place.

YBARS — Vector of length NSTRAT containing the strata means. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)

YVARS — Vector of length NSTRAT containing the within-strata variances. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)

NSAMPS — Vector of length NSTRAT containing the number of nonmissing observations from each stratum. (Output, if IDO= 0 or 1; input/output, if IDO = 2 or 3.)

STAT — Vector of length 13 containing the resulting statistics. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)
These are:

I	STAT(I)
1	Estimate of the mean.
2	Estimate of the total.
3	Variance estimate of the mean estimate.
4	Variance estimate of the total estimate.
5	Lower confidence limit for the mean.
6	Upper confidence limit for the mean.
7	Lower confidence limit for the total.
8	Upper confidence limit for the total.
9	Estimate of the coefficient of variation of the mean and total estimates.
10	Number of degrees of freedom associated with the variance estimates of the mean and total estimates. When IVOPT = 1, STAT(10) contains an effective number of degrees of freedom determined according to the Satterthwaite approximation.
11	Variance estimate of the mean estimate assuming that sampling was simple random instead of stratified random.
12	Pooled estimate of the common variance, when IVOPT = 0. If IVOPT = 1, STAT(12) is not defined.
13	The number of missing values.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of SMPSS for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to SMPSS will be made. Initialization and updating for the data in Y are performed.
2	This is an intermediate invocation of SMPSS, and updating for the data in Y is performed.
3	This is the final invocation of this routine. Updating for the data in Y and wrap-up computations are performed.

NSTRAT — Number of strata into which the population is divided. (Input)
In the vectors of length NSTRAT, the elements are all ordered in the same way. That is, the first stratum is always the first, the second always the second, and so on.
Default: NSTRAT = size (NROWS,1).

IVOPT — Within-stratum variance assumption indicator. (Input)
If IVOPT = 0, the true within-stratum variance is assumed constant, and a pooled estimate of that variance is returned in STAT(12). If IVOPT = 1, separate within-strata variance estimates are assumed.
Default: IVOPT = 0.

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)
A CONPER percent confidence interval is computed; hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 – 2.0 * (100.0 – ONECL).
Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: `CALL SMPSS (NROWS, Y, NPOPS, YBARS, YVARS, NSAMPS, STAT[,…])`

Specific: The specific interface names are S_SMPSS and D_SMPSS.

FORTRAN 77 Interface

Single:
```
CALL SMPSS (IDO, NSTRAT, NROWS, Y, NPOPS, IVOPT, CONPER,
YBARS, YVARS, NSAMPS, STAT)
```

Double: The double precision name is DSMPSS.

Example

In this example, we use a stratified sample from the data in Table 5.1 of Cochran (1977): the 1930 population (in 1000's) of 64 cities in the United States. The 64 cities are the "population", and our objective is to estimate the mean and total number of inhabitants in these 64 cities. There are two strata: the largest 16 cities and the remaining cities. We use stratified sampling with equal sample sizes. To choose the random sample, we use routine RNSRI (see Chapter 18, Random Number Generation), as follows:

```
      USE RNSET_INT
      USE RNSRI_INT
      INTEGER     ISEED, NSAMP, NPOP, INDEX(12)
      NSAMP = 12
      NPOP = 16
      ISEED = 123457
      CALL RNSET(ISEED)
      CALL RNSRI(NPOP, INDEX)
      WRITE(*, *) INDEX
      NPOP = 48
      CALL RNSRI(NPOP, INDEX)
      WRITE(*, *) INDEX
      END
```

This yields the population indices {2, 3, 4, 6, 8, 10, 11, 12, 13, 14, 15, 16} for the first stratum and {4, 8, 10, 11, 13, 16, 29, 30, 36, 37, 45, 46} for the second stratum. The corresponding values from Table 5.1 are encoded in the program below.

```
      USE SMPSS_INT
      USE UMACH_INT
      INTEGER    NSTRAT
      PARAMETER  (NSTRAT=2)
!
      INTEGER    I, IVOPT, NOUT, NPOPS(NSTRAT), NROWS(NSTRAT), &
                 NSAMPS(NSTRAT)
      REAL       STAT(13), Y(24), YBARS(NSTRAT), YVARS(NSTRAT)
!
      DATA Y/822., 781., 805., 1238., 634., 487., 442., 451., 459., &
          464., 400., 366., 302., 291., 272., 284., 270., 260., 139., &
          170., 154., 140., 163., 116./
!
      NPOPS(1) = 16
      NPOPS(2) = 48
      IVOPT    = 1
!                                  All data are input at once.
      NROWS(1) = 12
      NROWS(2) = 12
      CALL SMPSS (NROWS, Y, NPOPS, YBARS, YVARS, NSAMPS, STAT, IVOPT=IVOPT)
!                                  Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (STAT(I),I=1,11), STAT(13)
99999 FORMAT (' Mean estimate = ', F8.3, '      Total estimate = ', &
            F9.1, /, ' Vhat of mean  = ', F8.3, '      Vhat of total ' &
            , ' = ', F9.1, /, ' Confidence limits for mean  ', F8.3, &
            ',', F8.3, /, ' Confidence limits for total ', F8.1, &
            ',', F8.1, /, ' C. V.       = ', F8.1, '        Degrees ' &
```

```
     , 'of freedom = ', F8.1, /, ' SRS var. estimate = ', &
     F8.3, '  Number missing = ', F8.0)
END
```

Output

```
Mean estimate =  313.167      Total estimate =   20042.7
Vhat of mean  =  264.703      Vhat of total  = 1084224.6
Confidence limits for mean   279.180, 347.153
Confidence limits for total  17867.5, 22217.8
C. V.       =      5.2        Degrees of freedom =     19.6
SRS var. estimate = 1288.075  Number missing =       0.
```

Comments

Information Error

Type	Code	
4	1	The population size for each stratum is equal to one.

Description

Routine SMPSS computes point and interval estimates for the population mean and total from a stratified random sample of one variable. The routine uses the standard methods discussed in Chapters 5 and 5A of Cochran (1977). The sample means for the individual strata are accumulated in YBARS, and the corrected sums of squares are accumulated in YVARS. In the postprocessing phase, the quantities in STAT are computed using the strata statistics in YBARS, YVARS, and NSAMPS. The parameters IDO and NROWS allow either all or part of the data to be brought in at one time.

SMPST

Computes statistics for inferences regarding the population mean and total given continuous data from a two-stage sample with equisized primary units.

Required Arguments

NUNSAM — Number of primary units into which the sample is divided. (Input)

NELSAM — Number of elements in the sample in each sampled primary unit. (Input)

Y — Vector of length NOBS containing the elements of the two-stage sample. (Input)
The elements from each primary unit must occur contiguously within Y. Since there must be an equal number from each primary unit, Y must contain no missing values.

NUNPOP — Number of primary units in the sampled population. (Input)

NELPOP — Number of elements in each primary unit in the population. (Input)

PUMEAN — Vector of length NUNSAM containing the means of the primary units in the sample. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)
The estimates are ordered in correspondence with the ordering of primary units in Y.

PUVAR — Vector of length NUNSAM containing the sample variances of the primary units in the sample. (Output, if IDO= 0 or 1; input/output, if IDO= 2 or 3)
The estimates are ordered in correspondence with the ordering of primary units in Y.

STAT — Vector of length 9 containing the resulting statistics. (Output, if IDO= 0 or 1; input/output, if IDO = 2 or 3)

I	STAT(I)
1	Estimate of the mean.
2	Estimate of the total.
3	Variance of the mean estimate.
4	Variance estimate of the total estimate.
5	Lower confidence limit for the mean.
6	Upper confidence limit for the mean
7	Lower confidence limit for the total.
8	Upper confidence limit for the total.
9	Estimate (expressed as a percentage) of the coefficient of variation of the mean and total estimates.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of SMPST for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to SMPST will be made. Initialization and updating for the data in Y are performed.
2	This is an intermediate invocation of SMPST, and updating for the data in Y is performed.

IDO	Action
3	This is the final invocation of this routine. Updating for the data in Y and wrap-up computations are performed.

NOBS — The number of observations currently input in Y. (Input)
NOBS may be positive or zero. If NOBS = 0, IDO must equal 3, and only wrap-up computations are performed.
Default: NOBS = size (Y,1).

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)
A CONPER percent confidence interval is computed; hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 – 2.0 * (100.0 – ONECL).
Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: `CALL SMPST (NUNSAM, NELSAM, Y, NUNPOP, NELPOP, PUMEAN, PUVAR, STAT [,…])`

Specific: The specific interface names are S_SMPST and D_SMPST.

FORTRAN 77 Interface

Single: `CALL SMPST (IDO, NUNSAM, NELSAM, NOBS, Y, NUNPOP, NELPOP, CONPER, PUMEAN, PUVAR, STAT)`

Double: The double precision name is DSMPST.

Example

In this example, we have a sample of two primary units, with five subunits from each. The population consists of 10 primary units with 15 elements each.

```
      USE SMPST_INT
      USE UMACH_INT
      INTEGER    NELPOP, NELSAM, NOBS, NOUT, NUNPOP, NUNSAM
      REAL       PUMEAN(2), PUVAR(2), STAT(9), Y(10)
!
      DATA Y/2.7, 5.1, 4.3, 2.8, 1.9, 6.2, 4.8, 5.1, 7.2, 6.5/
!
      NUNSAM = 2
      NELSAM = 5
      NOBS   = 10
      NUNPOP = 10
      NELPOP = 15
!                                All data are input at once.
      CALL SMPST (NUNSAM, NELSAM, Y, NUNPOP, NELPOP, &
```

```
                  PUMEAN, PUVAR, STAT)
!                                 Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STAT
99999 FORMAT (' Mean estimate = ', F8.3, '      Total estimate = ', &
             F8.1, /, ' Vhat of mean  = ', F8.3, '      Vhat of total ' &
             , ' = ', F8.1, /, ' Confidence limits for mean  ', F8.3, &
             ',', F8.3, /, ' Confidence limits for total ', F8.1, &
             ',', F8.1, /, ' C. V.       = ', F8.1, '%')
      END
```

Output

```
Mean estimate =    4.660      Total estimate =    699.0
Vhat of mean  =    1.370      Vhat of total  =  30823.7
Confidence limits for mean     2.366,   6.954
Confidence limits for total    354.9,  1043.1
C. V.       =     25.1%
```

Description

Routine SMPST computes point and interval estimates for the population mean and total from a two-stage sample with primary units that are all equal in size. A two-stage sample might be taken if each unit ("primary unit") in the population can be divided into smaller units. Primary units are selected first, and then those selected are subsampled. The routine uses the standard methods discussed in Chapter 10 of Cochran (1977). The sample means for the individual primary units are accumulated in PUMEAN, and the corrected sums of squares are accumulated in PUVAR. In the postprocessing phase, the quantities in STAT are computed using the primary unit statistics. The parameters IDO and NOBS allow either all or part of the data to be brought in at one time.

Following the notation of Cochran (1977), let n (NUMSAM) be the number of primary units in the sample, let m (NELSAM) be the number of elements (subunits) subsampled from each primary unit, let N (NUMPOP) be the total number of primary units in the population, let M (NELPOP) be the total number of elements in each primary unit (in the population), and let y_{ij} be the j-th element in the i-th primary unit. The sample mean per subunit in the i-th primary unit is

$$\bar{y}_i = \frac{1}{m}\sum_{j=1}^{m} y_{ij}$$

The estimate of the population mean is

$$\bar{\bar{y}} = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}$$

The estimate of the variance of

$$\bar{\bar{y}} \text{ is } \frac{N-n}{nN(n-1)}\sum_{i=1}^{n}\left(\bar{y}_i - \bar{\bar{y}}\right)^2 + \frac{M-m}{m(m-1)MnN}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(y_{ij} - \bar{y}_i\right)^2$$

Chapter 13: Survival Analysis, Life Testing, and Reliability

Routines

Usage Notes

The routines described in this chapter have primary application in the areas of reliability and life testing, but they may find application in any situation in which time is a variable of interest. Kalbfleisch and Prentice (1980), Elandt-Johnson and Johnson (1980), Lee (1980), Gross and Clark (1975), Lawless (1982), and Chiang (1968) are general references for discussing the models and methods used here.

Kaplan-Meier (product-limit) estimates of the survival distribution in a single population is available through routine KAPMR (page 1063), and these can be printed using KTBLE (page 1068). Routine TRNBL (page 1073) computes generalized Kaplan-Meier estimates. Routine PHGLM (page 1078) computes the parameter estimates in a proportional hazards model. Routine SVGLM (page 1095) fits any of several generalized linear models, and STBLE (page 1111) computes estimates of survival probabilities based on the same models. Routine ACTBL (page 1121) computes and (optionally) prints an actuarial table based either upon a cohort followed over time or a cross-section of a population.

KAPMR

Computes Kaplan-Meier estimates of survival probabilities in stratified samples.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

IRT — Column number in X containing the response variable. (Input)
For the *i*-th right-censored observation, X(*i*, IRT) contains the right-censoring time. Otherwise, X(*i*, IRT) contains the failure time. (See ICEN.)

SPROB — NOBS by 2 matrix. (Output)
SPROB(*i*, 1) contains the estimated survival probability at time X(*i*, IRT) in the *i*-th observation's stratum, while SPROB(*i*, 2) contains Greenwood's estimate of the standard deviation of this estimated probability. If the *i*-th observation contains censor codes out of range or if a variable is missing, then the corresponding elements of SPROB are set to missing (NaN, not a number). Similarly, if an element in SPROB is not defined, then it is set to missing.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Column number in X containing the frequency of response for this observation. (Input)
If IFRQ = 0, a response frequency of 1 for each observation is assumed.
Default: IFRQ = 0.

ICEN — Column number in X containing the censoring code for this observation. (Input)
Default: ICEN = 0.
If ICEN = 0, a censoring code of 0 is assumed. Valid censoring codes are:

Code	Meaning
0	Exact failure at X(*i*, IRT).
1	Right censored. The response is greater than X(*i*, IRT).

If X(*i*, ICEN) is not 0 or 1, then the *i*-th observation is omitted from the analysis.

IGRP — Column number in X containing the stratum number for this observation. (Input)
If IGRP = 0, the data is assumed to be from one stratum. Otherwise, column IGRP of X contains a unique value for each stratum in the data. Kaplan-Meier estimates are

computed within each stratum.
Default: IGRP = 0.

ISRT — Sorting option. (Input)
If ISRT = 1, column IRT of X is assumed to be sorted in ascending order within each stratum. Otherwise, a detached sort will be performed by KAPMR . If sorting is performed by KAPMR, all censored individuals are assumed to follow tied failures.
Default: ISRT = 0.

LDSPRO — Leading dimension of SPROB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSPRO = size (SPROB,1).

NRMISS — Number of rows of data in X that contain any missing values. (Output)

FORTRAN 90 Interface

Generic: `CALL KAPMR (X, IRT, SPROB [,…])`

Specific: The specific interface names are S_KAPMR and D_KAPMR.

FORTRAN 77 Interface

Single:
```
CALL KAPMR (NOBS, NCOL, X, LDX, IRT, IFRQ, ICEN, IGRP,
ISRT, SPROB, LDSPRO, NRMISS)
```

Double: The double precision name is DKAPMR.

Example

The following example is taken from Kalbfleisch and Prentice (1980, page 1). The first column in *X* contains the death/censoring times for rats suffering from vaginal cancer. The second column contains information as to which of two forms of treatment were provided, while the third column contains the censoring code. Finally, the fourth column contains the frequency of each observation. The product-limit estimates of the survival probabilities are computed for both groups with one call to KAPMR. In this example, the output in SPROB has been equivalenced with columns 5 and 6 of *X* so that the input and output matrices could be printed together. Routine KAPMR could have been called with the ISRT = 1 option in effect if the censored observations had been sorted with respect to the failure time variable.

```
      USE KAPMR_INT
      USE WRRRL_INT
      USE UMACH_INT
      INTEGER    ICEN, IFRQ, IGRP, IRT, ISRT, LDSPRO, LDX, NCOL, NOBS
      PARAMETER  (ICEN=3, IFRQ=4, IGRP=2, IRT=1, ISRT=0, LDSPRO=33, &
                 LDX=33, NCOL=6, NOBS=33)
!
      INTEGER    NOUT, NRMISS
      REAL       SPROB(LDSPRO,2), X(LDX,NCOL)
```

```
      CHARACTER  XLABEL(7)*6, YLABEL(1)*6
!
      EQUIVALENCE (X(1,5), SPROB)
!
      DATA XLABEL/'OBS', 'TIME', 'GROUP', 'CENSOR', 'FREQ', 'S-HAT', &
          'SE'/
      DATA YLABEL/'NUMBER'/
      DATA X/143, 164, 188, 190, 192, 206, 209, 213, 216, 220, 227, &
          230, 234, 246, 265, 304, 216, 244, 142, 156, 163, 198, 205, &
          232, 233, 239, 240, 261, 280, 296, 323, 204, 344, 18*5, &
          15*7, 16*0, 2*1, 13*0, 4*1, 2, 20*1, 2, 4, 3*1, 2*2, 3*1, &
          66*0/
!
      CALL KAPMR (X, IRT, SPROB, IFRQ=IFRQ, ICEN=ICEN, IGRP=IGRP, &
                  NRMISS=NRMISS)
!
      CALL WRRRL ('X/SPROB', X, YLABEL, XLABEL, FMT='(W10.6)')
      CALL UMACH (2, NOUT)
      WRITE (NOUT,'(//'' NRMISS = '', I5)') NRMISS
      END
```

Output

```
                                X/SPROB
 OBS     TIME       GROUP      CENSOR        FREQ       S-HAT          SE
  1    143.000       5.000       0.000       1.000       0.947       0.051
  2    164.000       5.000       0.000       1.000       0.895       0.070
  3    188.000       5.000       0.000       2.000       0.789       0.094
  4    190.000       5.000       0.000       1.000       0.737       0.101
  5    192.000       5.000       0.000       1.000       0.684       0.107
  6    206.000       5.000       0.000       1.000       0.632       0.111
  7    209.000       5.000       0.000       1.000       0.579       0.113
  8    213.000       5.000       0.000       1.000       0.526       0.115
  9    216.000       5.000       0.000       1.000       0.474       0.115
 10    220.000       5.000       0.000       1.000       0.414       0.115
 11    227.000       5.000       0.000       1.000       0.355       0.112
 12    230.000       5.000       0.000       1.000       0.296       0.108
 13    234.000       5.000       0.000       1.000       0.237       0.101
 14    246.000       5.000       0.000       1.000       0.158       0.093
 15    265.000       5.000       0.000       1.000       0.079       0.073
 16    304.000       5.000       0.000       1.000       0.000         NaN
 17    216.000       5.000       1.000       1.000       0.474       0.115
 18    244.000       5.000       1.000       1.000       0.237       0.101
 19    142.000       7.000       0.000       1.000       0.952       0.046
 20    156.000       7.000       0.000       1.000       0.905       0.064
 21    163.000       7.000       0.000       1.000       0.857       0.076
 22    198.000       7.000       0.000       1.000       0.810       0.086
 23    205.000       7.000       0.000       1.000       0.759       0.094
 24    232.000       7.000       0.000       2.000       0.658       0.105
 25    233.000       7.000       0.000       4.000       0.455       0.111
 26    239.000       7.000       0.000       1.000       0.405       0.110
 27    240.000       7.000       0.000       1.000       0.354       0.107
 28    261.000       7.000       0.000       1.000       0.304       0.103
 29    280.000       7.000       0.000       2.000       0.202       0.090
 30    296.000       7.000       0.000       2.000       0.101       0.068
 31    323.000       7.000       0.000       1.000       0.051       0.049
```

```
32    204.000       7.000       1.000       1.000       0.810       0.086
33    344.000       7.000       1.000       1.000         NaN         NaN

NRMISS =     0
```

Comments

1. Workspace may be explicitly provided, if desired, by use of K2PMR/DK2PMR. The reference is:

   ```
   CALL K2PMR (NOBS, NCOL, X, LDX, IRT, IFRQ, ICEN, IGRP, ISRT,
   SPROB, LDSPRO, NRMISS, IGP, IPERM, INDDR, IWK, WK, IPER)
   ```

   The additional arguments are as follows:

   IGP — Work vector of length NOBS.

   IPERM — Work vector of length NOBS + NCOL.

   INDDR — Work vector of length NOBS.

   IWK — Work vector of length max(NOBS, NCOL).

   WK — Work vector of length 2 * max(NOBS, NCOL).

   IPER — Work vector of length NOBS.

2. Missing values may occur in any of the columns of X. Any row of X that contains missing values in the IRT, ICEN, or IFRQ columns (when the ICEN and IFRQ columns are present) is omitted from the analysis. Missing values in the IGRP column, if present, are classified into an additional "missing" group.

Description

Routine KAPMR computes Kaplan-Meier (or product-limit) estimates of survival probabilities for a sample of failure times that possibly contain right censoring. A survival probability $S(t)$ is defined as $1 - F(t)$, where $F(t)$ is the cumulative distribution function of the failure times (t). Greenwood's estimate of the standard errors of the survival probability estimates are also computed. (See Kalbfleisch and Prentice, 1980, pages 13 and 14.)

Let (t_i, δ_i), for $i = 1,\ldots, n$ denote the failure/censoring times and the censoring codes for the n observations in a single sample. Here, t_i = X(i, IRT) is a failure time if δ_i is 0, where δ_i = X(i, ICEN). Also, t_i is a censoring time if δ_i is 1. Rows in X containing values other than 0 or 1 for δ_i are ignored. Let the number of observations in the sample that have not failed by time $s_{(i)}$ be denoted by $n_{(i)}$, where $s_{(i)}$ is an ordered (from smallest to largest) listing of the distinct failure times (censoring times are omitted). Then the Kaplan-Meier estimate of the survival probabilities is a step function, which in the interval from $s_{(i)}$ to $s_{(i+1)}$ (including the lower endpoint) is given by

$$\hat{S}(t) = \prod_{j=1}^{i} \left(\frac{n_{(j)} - d_{(j)}}{n_{(j)}} \right)$$

where $d_{(j)}$ denotes the number of failures occurring at time $s_{(j)}$. Note that one row of X may correspond to more than one failed (or censored) observation when the frequency option is in effect (IFRQ is not zero). The Kaplan-Meier estimate of the survival probability prior to time $s_{(1)}$ is 1.0, while the Kaplan-Meier estimate of the survival probability after the last failure time is not defined.

Greenwood's estimate of the variance of

$$\hat{S}(t)$$

in the interval from $s_{(i)}$ to $s_{(i+1)}$ is given as

$$\text{est.var}(\hat{S}(t)) = \hat{S}^2(t) \sum_{j=1}^{i} \frac{d_{(j)}}{n_{(j)}(n_{(j)} - d_{(j)})}$$

Routine KAPMR computes the single sample estimates of the survival probabilities for all samples of data included in X during a single call. This is accomplished through the IGRP column of X, which if present, must contain a distinct code for each sample of observations. If IGRP = 0, there is no grouping column, and all observations are assumed to be from the same sample.

When failures and right-censored observations are tied and the data are to be sorted by KAPMR (ISRT is not 1), KAPMR assumes that the time of censoring for the tied-censored observations is immediately after the tied failure (within the same sample). When the ISRT = 1 option is in effect, the data are assumed to be sorted from smallest to largest according to column IRT of X within each stratum. Furthermore, a small increment of time is assumed (theoretically) to elapse between the failed and censored observations that are tied (in the same sample). Thus, when the ISRT = 1 option is in effect, the user must sort all of the data in X from smallest to largest according to column IRT (and column IGRP, if present). By appropriate sorting of the observations, the user can handle censored and failed observations that are tied in any manner desired.

KTBLE

Prints Kaplan-Meier estimates of survival probabilities in stratified samples.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

IRT — Column number of X containing the response variable. (Input)
For the i-th right-censored observation, X(i, IRT) contains the right-censoring time. Otherwise, X(i, IRT) contains the failure time. See argument ICEN.

SPROB — NOBS by 2 matrix. (Input)
SPROB (i, 1) contains the estimated survival probability at time X(i, IRT) in the i-th

observation's stratum, while SPROB(*i*, 2) contains Greenwood's estimate of the standard deviation of this estimated probability. SPROB will usually be computed by routine KAPMR (page 1063). It may contain missing values after the last failed observation in each group.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. For positive IFRQ, column number IFRQ of X contains the frequencies.
Default: IFRQ = 0.

ICEN — Column number of X containing the censoring code for this observation. (Input)
Default: ICEN = 0.
If ICEN = 0, a censoring code of 0 is assumed. Valid censoring codes are:

Code	Meaning
0	Exact failure at X(*i*, IRT).
1	Right censored. The response is greater than X(*i*, IRT).

If X(*i*, ICEN) is not zero or one, then the *i*-th observation is omitted from the analysis.

IGRP — Column number of X containing the stratum number for this observation. (Input)
If IGRP = 0, the data are assumed to be from one stratum. Otherwise, column IGRP of X contains a unique value for each stratum in the data. Kaplan-Meier estimates are computed within each stratum.
Default: IGRP = 0.

ISRT — Sorting option. (Input)
If ISRT = 1, column IRT of X is assumed to be sorted in ascending order within each stratum. Otherwise, a detached sort will be performed by KTBLE. If sorting is performed by KTBLE, all censored observations are assumed to follow failing observations with the same response time in X (*i*, IRT).
Default: ISRT = 0.

LDSPRO — Leading dimension of SPROB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSPRO = size (SPROB,1).

FORTRAN 90 Interface

Generic: `CALL KTBLE (X, IRT, SPROB [,…])`

Specific: The specific interface names are S_KTBLE and D_KTBLE.

FORTRAN 77 Interface

Single:
```
CALL KTBLE (NOBS, NCOL, X, LDX, IRT, IFRQ, ICEN, IGRP,
ISRT, SPROB, LDSPRO)
```

Double: The double precision name is DKTBLE.

Example

This example illustrates the typical use of KTBLE. First, routine KAPMR (page 1063) is used to compute the survival probabilities. This is followed by a call to KTBLE that performs the printing. The input data is given as:

143, 164, 188(2), 190, 192, 206, 209, 213, 216, 220, 227, 230, 234, 246, 265, 304, 216*, 244*, 142, 156, 163, 198, 205, 232(2), 233(4), 239, 240, 261, 280(2), 296(2), 323, 204*, 344*

where items marked with an * are right censored; and the frequency of each failure time, if different from 1, is given in parenthesis.

```
      USE KAPMR_INT
      USE KTBLE_INT
      INTEGER    ICEN, IFRQ, IGRP, IRT, ISRT, LDSPRO, LDX, NCOL, NOBS
      PARAMETER  (ICEN=3, IFRQ=4, IGRP=2, IRT=1, ISRT=0, LDSPRO=33, &
                 LDX=33, NCOL=4, NOBS=33)
!
      INTEGER    NRMISS
      REAL       SPROB(LDSPRO,2), X(LDX,NCOL)
!
      DATA X/143, 164, 188, 190, 192, 206, 209, 213, 216, 220, 227, &
          230, 234, 246, 265, 304, 216, 244, 142, 156, 163, 198, 205, &
          232, 233, 239, 240, 261, 280, 296, 323, 204, 344, 18*5, &
          15*7, 16*0, 2*1, 13*0, 4*1, 2, 20*1, 2, 4, 3*1, 2*2, 3*1/
!
      CALL KAPMR (X, IRT, SPROB, IFRQ=IFRQ, ICEN=ICEN, IGRP=IGRP)
!
      CALL KTBLE (X, IRT, SPROB, IFRQ=IFRQ, ICEN=ICEN, IGRP=IGRP)
      END
```

Output

```
          Kaplan Meier Survival Probabilities
            For Group Value =      5.00000

Number      Number                  Survival     Estimated
at risk     Failing        Time  Probability    Std. Error
     19           1         143      0.94737       0.05123
     18           1         164      0.89474       0.07041
     17           2         188      0.78947       0.09353
     15           1         190      0.73684       0.10102
     14           1         192      0.68421       0.10664
     13           1         206      0.63158       0.11066
     12           1         209      0.57895       0.11327
     11           1         213      0.52632       0.11455
     10           1         216      0.47368       0.11455
      8           1         220      0.41447       0.11452
      7           1         227      0.35526       0.11243
      6           1         230      0.29605       0.10816
      5           1         234      0.23684       0.10145
      3           1         246      0.15789       0.09343
      2           1         265      0.07895       0.07279
      1           1         304      0.00000           NaN

Total number in group    =       19
Total number failing     =       17
Product Limit Likelihood =      -49.1692


             Kaplan Meier Survival Probabilities
               For Group Value =      7.00000
Number      Number                  Survival     Estimated
at risk     Failing        Time  Probability    Std. Error
     21           1         142      0.95238       0.04647
     20           1         156      0.90476       0.06406
     19           1         163      0.85714       0.07636
     18           1         198      0.80952       0.08569
     16           1         205      0.75893       0.09409
     15           2         232      0.65774       0.10529
     13           4         233      0.45536       0.11137
      9           1         239      0.40476       0.10989
      8           1         240      0.35417       0.10717
      7           1         261      0.30357       0.10311
      6           2         280      0.20238       0.09021
      4           2         296      0.10119       0.06778
      2           1         323      0.05060       0.04928

Total number in group    =       21
Total number failing     =       19
Product Limit Likelihood =      -50.4277
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `K2BLE/DK2BLE`. The reference is:

```
CALL K2BLE (NOBS, NCOL, X, LDX, IRT, IFRQ, ICEN, IGRP, ISRT,
SPROB, LDSPRO, ALGL, IPERM, INDDR, WK, WK1, IWK)
```

The additional arguments are as follows:

ALGL — Work vector of length NOBS that contains the log likelihoods of the Kaplan-Meier estimates. If the number of groups is known to be *m* or less, then ALGL can be of length *m*.

IPERM — Work vector of length NOBS.

INDDR — Work vector of length NOBS.

WK — Work vector of length NOBS.

WK1 — Work vector of length 2 * max(NOBS, NCOL).

IWK — Work vector of length max(NOBS, NCOL).

2. Informational errors

Type	Code	
4	1	An invalid value for SPROB has been detected. The estimated survival probability must be between zero and one, inclusive, and nonincreasing with failure time within each group.
4	2	A negative frequency has been detected.
4	3	A missing value for SPROB has been detected but later failures occur. Missing values are not allowed prior to the last failed observation.

3. Missing values may occur in any of the columns of X. Any row of X that contains missing values in the IRT, ICEN, or IFRQ columns (when the ICEN and IFRQ columns are present) is omitted from the analysis. Missing values in the IGRP column, if present, are classified into an additional "missing" group.

Description

Routine KTBLE prints life tables based upon the Kaplan-Meier estimates of the survival probabilities (see routine KAPMR, page 1063). One table for each stratum is printed. In addition to the survival probabilities at each failure point, the following is also printed: the number of individuals remaining at risk, Greenwood's estimate of the standard errors for the survival probabilities, and the Kaplan-Meier log-likelihood. The Kaplan-Meier log-likelihood is computed as:

$$\ell = \sum_j d_{(j)} \ln d_{(j)} + (n_{(j)} - d_{(j)}) \ln(n_{(j)} - d_{(j)}) - n_{(j)} \ln n_{(j)}$$

where the sum is with respect to the distinct failure times $s_{(j)}$, $d_{(j)}$ is the number of failures occurring at time $s_{(j)}$, and $n_{(j)}$ is the number of observations that had not yet failed immediately prior to $s_{(j)}$. Note that sorting is performed by both KAPMR (page 1063), and by routine KTBLE.

The user may sort the data to be increasing in failure time and then use the ISRT = 1 option to avoid this double sorting.

TRNBL

Computes Turnbull's generalized Kaplan-Meier estimates of survival probabilities in samples with interval censoring.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

ILT — For interval-censored and left-censored observations, the column number in X that contains the upper endpoint of the failure interval. (Input)
See argument ICEN. If ILT = 0, left-censored and interval-censored observations cannot be input.

IRT — For interval-censored and right-censored observations, the column number in X that contains the lower endpoint of the failure interval. (Input)
See argument ICEN. IRT must not be zero.

NINTVL — Number of failure intervals found. (Output)

SPROB — NINTVL by 4 matrix. (Output)

Col.	Description
1	Lower endpoint of the failure interval
2	Upper endpoint of the failure interval
3	Estimated change in the survival probability density within the failure interval
4	Estimate of the survival probability for the interval

The estimated survival probability is a constant equal to SPROB(*i*, 4) from SPROB (*i*, 2) to SPROB(*i* + 1, 1). The estimated survival probability is 1 prior to SPROB(1, 1). The estimated survival probability is undefined in the interval SPROB(*i*, 1) to SPROB(*i*, 2). If the NINTVL-th interval is from SPROB(NINTVL, 1) to infinity, then SPROB(NINTVL, 2) is set to positive machine infinity.

ALGL — Optimized log-likelihood for the input data. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)
If IFRQ = 0, a response frequency of 1 for each observation is assumed. For positive IFRQ, column number IFRQ contains the frequency of response for each observation.
Default: IFRQ = 0.

ICEN — Censoring code option. (Input)
Default: ICEN = 0.
If ICEN = 0, a censoring code of 0 is assumed. For positive ICEN, column number ICEN contains the censoring code for each observation. Valid censoring codes are:

Code	Meaning
0	Exact failure at X(*i*, IRT).
1	Right censored. The response is greater than X(*i*, IRT).
2	Left censored. The response is less than or equal to X(*i*, ILT).
3	Interval censored. The response is greater than X(*i*, IRT), but less than or equal to X(*i*, ILT).

MAXIT — Maximum number of iterations. (Input)
Default: MAXIT = 30.

EPS — Convergence criterion. (Input)
Convergence is assumed when the relative change in the log-likelihood from one iteration to the next is less than EPS. EPS = 0.00001 is typical.
Default: EPS = 0.00001.

IPRINT — Printing option. (Input)
IPRINT = 0 means that no printing is performed. IPRINT = 1 means that printing is performed.
Default: IPRINT = 0.

LDSPRO — Leading dimension of SPROB exactly as specified in the dimension statement in the calling program. (Input)
If LDSPRO is less than NINTVL, only the first LDSPRO intervals are returned in SPROB.
Default: LDSPRO = size (SPROB,1).

NRMISS — Number of rows of data in X that contain missing values. (Output)
Any row of X that contains missing values in the ILT, IRT, ICEN, or IFRQ columns (when the ILT, ICEN or IFRQ is positive) is omitted from the analysis.

FORTRAN 90 Interface

Generic: `CALL TRNBL (X, ILT, IRT, NINTVL, SPROB, ALGL [,…])`

Specific: The specific interface names are `S_TRNBL` and `D_TRNBL`.

FORTRAN 77 Interface

Single:
```
CALL TRNBL (NOBS, NCOL, X, LDX, ILT, IRT, IFRQ, ICEN,
MAXIT, EPS, IPRINT, NINTVL, SPROB, LDSPRO, ALGL, NRMISS)
```

Double: The double precision name is `DTRNBL`.

Example

The following example contains exact failure, right-, left-, and interval-censored observations. The 20 observations yield 15 change intervals. The last interval is from 192 to ∞, and corresponds to a right-censored observation. When the last interval is infinite, as is the case here, the second column of SPROB contains +∞ in the NINTVL-th position. Left-or right-censored observations input in X are arbitrarily assigned the value 0.0 for the non-specified endpoint.

```
      USE WRRRN_INT
      USE TRNBL_INT
      INTEGER    ICEN, IFRQ, ILT, IPRINT, IRT, LDSPRO, LDX, NCOL, NOBS
      PARAMETER  (ICEN=4, IFRQ=3, ILT=1, IPRINT=1, IRT=2, LDSPRO=20, &
                  LDX=20, NCOL=4, NOBS=20)
!
      INTEGER    NINTVL, NRMISS
      REAL       ALGL, SPROB(LDSPRO,4), X(LDX,NCOL)
!
      DATA X/0.9, 1.9, 2.5, 3.5, 6.3, 7.1, 18., 25.1, 25.3, 30.3, 45.9, &
          63.5, 70.1, 73.0, 93.0, 94.4, 96.0, 0.0, 191.4, 0.0, 0.9, &
          0.0, 0.0, 0.0, 6.3, 1.9, 1.8, 25.1, 9.5, 30.3, 45.9, &
          60.7, 70.1, 71.0, 74.0, 94.4, 96.0, 96.0, 191.4, 192.0, &
          17*1.0, 5.0, 1.0, 1.0, 0.0, 2.0, 2.0, 2.0, 0.0, 3.0, 3.0, &
          0.0, 3.0, 0.0, 0.0, 3.0, 0.0, 3.0, 3.0, 0.0, 0.0, 1.0, 0.0, &
          1.0/
!
      CALL WRRRN ('X', X)
!
      CALL TRNBL (X, ILT, IRT, NINTVL, SPROB, ALGL, IFRQ=IFRQ, &
                  ICEN=ICEN, IPRINT=IPRINT)
!
      END
```

Output

```
                X
          1       2       3       4
 1      0.9     0.9     1.0     0.0
 2      1.9     0.0     1.0     2.0
 3      2.5     0.0     1.0     2.0
 4      3.5     0.0     1.0     2.0
 5      6.3     6.3     1.0     0.0
 6      7.1     1.9     1.0     3.0
 7     18.0     1.8     1.0     3.0
 8     25.1    25.1     1.0     0.0
 9     25.3     9.5     1.0     3.0
10     30.3    30.3     1.0     0.0
11     45.9    45.9     1.0     0.0
12     63.5    60.7     1.0     3.0
13     70.1    70.1     1.0     0.0
14     73.0    71.0     1.0     3.0
15     93.0    74.0     1.0     3.0
16     94.4    94.4     1.0     0.0
17     96.0    96.0     1.0     0.0
18      0.0    96.0     5.0     1.0
19    191.4   191.4     1.0     0.0
20      0.0   192.0     1.0     1.0

      Iteration   Log-Likelihood   Relative convergence
              0     -54.94             ............
              1     -52.14               0.5367E-01
              2     -52.09               0.8407E-03
              3     -52.09               0.1372E-03
              4     -52.09               0.2476E-04
              5     -52.08               0.4614E-05

SPROB
                 Lower        Upper     Interval     Survival
Interval      Endpoint     Endpoint  Probability  Probability
       1        0.9000       0.9000       0.0972       0.9028
       2        1.9000       1.9000       0.1215       0.7813
       3        6.3000       6.3000       0.0729       0.7083
       4        9.5000      18.0000       0.0000       0.7083
       5       25.1000      25.1000       0.0833       0.6250
       6       30.3000      30.3000       0.0417       0.5833
       7       45.9000      45.9000       0.0417       0.5417
       8       60.7000      63.5000       0.0417       0.5000
       9       70.1000      70.1000       0.0417       0.4583
      10       71.0000      73.0000       0.0417       0.4167
      11       74.0000      93.0000       0.0417       0.3750
      12       94.4000      94.4000       0.0417       0.3333
      13       96.0000      96.0000       0.1111       0.2222
      14      191.4000     191.4000       0.1111       0.1111
      15      192.0000          Inf       0.1111       0.0000
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `T2NBL/DT2NBL`. The reference is:

```
CALL T2NBL (NOBS, NCOL, X, LDX, ILT, IRT, IFRQ, ICEN, MAXIT,
EPS, IPRINT, NINTVL, SPROB, LDSPRO, ALGL, NRMISS, WK, IPERM,
INDDR, WWK, IWK)
```

The additional arguments are as follows:

WK — Work vector of length 7 * NOBS.

IPERM — Work vector of length NOBS.

INDDR — Work vector of length NOBS.

WWK — Work vector of length 2 * max(NOBS, 7).

IWK — Work vector of length max(NOBS, 7).

2. Informational errors

Type	Code	
3	3	The maximum number of iterations was exceeded. Convergence is assumed.
4	1	There are no valid observations.
4	2	There are no finite failure intervals present in the data.

Description

Routine TRNBL computes nonparametric maximum likelihood estimates of a survival distribution based upon a random sample of data containing exact failure, right-censored, leftcensored (interval censored with a left endpoint of zero), or interval-censored observations. The computational method of Turnbull (1976) is used in computing the probability estimates. The model used is also discussed by Peto (1973).

Routine TRNBL begins by finding a set of regions or "failure intervals" (to distinguish them from "observation failure intervals") on the positive real axis in which a change in the survival probability occurs. The survival probability is constant outside of these regions, and undefined within them. Each region (failure interval) is composed of a single left and a single right endpoint obtained from the left and right endpoints of the observation failure intervals (for exact failure times, the left and right endpoints are equal). The regions are defined by the fact that no observation interval endpoints are allowed within a region, except at its endpoints. Note that the endpoints of the intervals need not correspond to a single observation. Regions defined by endpoints from two distinct observations are often obtained.

Let p_i, $i = 1, \ldots,$ NINTVL denote the change in the survival probability within the i-th region, and let the region be denoted by c_i. Let $n =$ NOBS and suppose that the observation failure interval for observation j is denoted by I_j. The EM (expectation, maximization) algorithm of Dempster, Laird and Rubin (1977) is used to find the optimal

$$\hat{p}_i\text{'}s$$

The algorithm is defined as follows:

For given

$$\hat{p}_i$$

compute the expected contribution of the j-th observation to the i-th change interval as

$$\hat{\mu}_{ij} = \frac{f_j \hat{p}_i \delta_{ij}}{\sum_j f_j \hat{p}_i \delta_{ij}}$$

where $\delta_{ij} = 1$ if $c_i \subseteq I_j$ and $\delta_{ij} = 0$ otherwise, and f_j is the observation frequency.

For given expectations

$$\hat{\mu}_{ij}$$

compute the new probability estimate as

$$\hat{p}_i = \frac{\sum_j \hat{\mu}_{ij}}{\sum_j f_j}$$

Iterate in this manner until convergence. Convergence is assumed when the relative change in the log-likelihood

$$(\ell = \sum_j f_j \ln(\sum_i \delta_{ij} \hat{p}_j))$$

is small (less than EPS). Because the algorithm is slow to converge, 5 expectation-maximization cycles are considered to be one iteration of the algorithm. The initial estimate for all the

$$\hat{p}_i\text{'}s$$

is taken to be one divided by the number of regions (failure intervals).

PHGLM

Analyzes time event data via the proportional hazards model.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)
When ITIE = 1, the observations in X must be grouped by stratum and sorted from largest to smallest failure time within each stratum, with the strata separated.

IRT — Column number in X containing the response variable. (Input)
For point observations, X(i, IRT) contains the time of the i-th event. For right-censored observations, X(i, IRT) contains the right-censoring time. Note that because PHGLM only uses the order of the events, negative "times" are allowed.

NVEF — Vector of length NEF containing the number of variables associated with each effect in the model. (Input)

INDEF — Index vector of length NVEF(1) + … + NVEF(NEF) containing the column numbers of X associated with each effect. (Input)
The first NVEF(1) elements of INDEF contain the column numbers of X for the variables

in the first effect. The next NVEF(2) elements in INDEF contain the column numbers for the second effect, etc.

MAXCL — An upper bound on the sum of the number distinct values taken by the classification variables. (Input)

NCOEF — Number of estimated coefficients in the model. (Output)

COEF — NCOEF by 4 matrix containing the parameter estimates and associated statistics. (Output, if INIT = 0; input, if INIT = 1 and MAXIT = 0, input/output, if INIT = 1 and MAXIT > 0)

Col.	Statistic
1	Coefficient estimate $\hat{\beta}$
2	Estimated standard deviation of the estimated coefficient.
3	Asymptotic normal score for testing that the coefficient is zero against the two-sided alternative.
4	*p*-value associated with the normal score in column 3.

When COEF is input, only column 1 needs to be given.

ALGL — The maximized log-likelihood. (Output)

COV — NCOEF by NCOEF matrix containing the estimated asymptotic variance-covariance matrix of the parameters. (Output)
For MAXIT = 0, COV is the inverse of the Hessian of the negative of the log-likelihood, computed at the estimates input in COEF.

XMEAN — Vector of length NCOEF containing the means of the design variables. (Output)

CASE — NOBS by 5 matrix containing the case statistics for each observation. (Output if MAXIT > 0; used as working storage otherwise)

Col.	Statistic
1	Estimated survival probability at the observation time.
2	Estimated observation influence or leverage.
3	A residual estimate.
4	Estimated cumulative baseline hazard rate.
5	Observation proportionality constant.

GR — Vector of length NCOEF containing the last parameter updates (excluding step halvings). (Output)
For MAXIT = 0, GR contains the inverse of the Hessian times the gradient vector computed at the estimates input in COEF.

IGRP — Vector of length NOBS giving the stratum number used for each observation. (Output)
If RATIO is not –1.0, additional "strata" (other than those specified by column ISTRAT of X) may be generated. IGRP also contains a record of the generated strata. See the algorithm section for more detail.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Column number in X containing the frequency of response for each observation. (Input)
If IFRQ = 0, a response frequency of 1 for each observation is assumed.
Default: IFRQ = 0.

IFIX — Column number in X containing a constant to be added to the linear response. (Input)
Default: IFIX = 0.
The linear response is taken to be $w_i + z_i\hat{\beta}$

where w_i is the observation constant, z_i is the observation design row vector, and $\hat{\beta}$ is the vector of estimated parameters. The "fixed" constant allows one to test hypotheses about parameters via the log-likelihoods. If IFIX = 0, the fixed parameter is assumed to be 0.

ICEN — Column number in X containing the censoring code for each observation. (Input)
Default: ICEN = 0.

If ICEN = 0 a censoring code of 0 is assumed for all observations.

X(*i*, ICEN)	Censoring
0	Point observation at X(*i*, IRT).

1 Right censored. The response is greater than X(i, IRT).

ISTRAT — Column number in X containing the stratification variable. (Input)
If ISTRAT = 0, all observations are considered to be in one stratum. Otherwise, column ISTRAT in X contains a unique number for each stratum. The risk set for an observation is determined by the its stratum.
Default: ISTRAT = 0.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 30 will usually be sufficient. Use MAXIT = 0 to compute the Hessian and gradient, stored in COV and GR, at the initial estimates. When MAXIT = 0, INIT must be 1.
Default: MAXIT = 30.

EPS — Convergence criterion. (Input)
Convergence is assumed when the relative change in ALGL from one iteration to the next is less than EPS. If EPS is zero, EPS = 0.0001 is assumed.
Default: EPS = 0.0001.

RATIO — Ratio at which a stratum is split into two strata. (Input)
Default: RATIO = 1000.0.
Let

$$r_k = \exp(z_k \hat{\beta} + w_k)$$

be the observation proportionality constant, where z_k is the design row vector for the k-th observation and w_k is the optional fixed parameter specified by X(k, IFIX). Let $r_{\min}$ be the minimum value r_k in a stratum, where, for failed observations, the minimum is over all times less than or equal to the time of occurrence of the k-th observation. Let $r_{\max}$ be the maximum value of r_k for the remaining observations in the group. Then, if $r_{\min} >$ RATIO $r_{\max}$, the observations in the group are divided into two groups at k. RATIO = 1000 is usually a good value. Set RATIO = –1.0 if no division into strata is to be made.

NCLVAR — Number of classification variables. (Input)
Dummy variables are generated for classification variables using the IDUMMY = 2 option of IMSL routine GRGLM (see Chapter 2, Regression). See Comment 3.
Default: NCLVAR = 0.

INDCL — Index vector of length NCLVAR containing the column numbers of X that are the classification variables. (Input, if NCLVAR is positive, not used otherwise)
If NCLVAR is 0, INDCL is not referenced and can be dimensioned of length 1 in the calling program.

NEF — Number of effects in the model. (Input)
In addition to effects involving classification variables, simple covariates and the

product of simple covariates are also considered effects.
Default: NEF = size(NVEF,1).

INIT — Initialization option. (Input)
If INIT = 1, then the NCOEF elements of column 1 of COEF contain the initial estimates on input to PHGLM. For INIT = 0, all initial estimates are taken to be 0.
Default: INIT = 0.

ITIE — Option parameter containing the method to be used for handling ties. (Input)
Default: ITIE = 0.

ITIE	Method
0	Breslow's approximate method
1	Failures are assumed to occur in the same order as the observations input in X. The observations in X must be sorted from largest to smallest failure time within each stratum, and grouped by stratum. All observations are treated as if their failure/censoring times were distinct when computing the log-likelihood.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing is performed, but observational statistics are not printed.
2	All output statistics are printed.

NCLVAL — Vector of length NCLVAR containing the number of values taken by each classification variable. (Output, if NCLVAR is positive, not used otherwise)
NCLVAL(*i*) is the number of distinct values for the *i*-th classification variable. If NCLVAR is zero, NCLVAL is not used and can be dimensioned of length 1 in the calling program.

CLVAL — Vector of length NCLVAL(1) + NCLVAL(2) + … + NCLVAL(NCLVAR) containing the distinct values of the classification variables. (Output, if NCLVAR is positive, not used otherwise)
The first NCLVAL(1) elements of CLVAL contain the values for the first classification variable, the next NCLVAL(2) elements contain the values for the second classification variable, etc. If NCLVAR is zero, then NCLVAL is not referenced and can be dimensioned of length 1 in the calling program.

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOEF = size (COEF,1).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

LDCASE — Leading dimension of CASE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCASE = size (CASE,1).

NRMISS — Number of rows of data in X that contain missing values in one or more columns IRT, IFRQ, IFIX, ICEN, ISTRAT, INDCL, or INDEF of X. (Output)

FORTRAN 90 Interface

Generic: `CALL PHGLM (X, IRT, NVEF, INDEF, MAXCL, NCOEF, COEF, ALGL, COV, XMEAN, CASE, GR, IGRP [,…])`

Specific: The specific interface names are S_PHGLM and D_PHGLM.

FORTRAN 77 Interface

Single:
```
CALL PHGLM (NOBS, NCOL, X, LDX, IRT, IFRQ, IFIX, ICEN,
ISTRAT, MAXIT, EPS, RATIO, NCLVAR, INDCL, NEF, NVEF, INDEF,
INIT, ITIE, IPRINT, MAXCL, NCLVAL, CLVAL, NCOEF, COEF,
LDCOEF, ALGL, COV, LDCOV, XMEAN, CASE, LDCASE, GR, IGRP,
NRMISS)
```

Double: The double precision name is DPHGLM.

Example 1

The following data are taken from Lawless (1982, page 287) and involve the survival of lung cancer patients based upon their initial tumor types and treatment type. In the first example, the likelihood is maximized with no strata present in the data. This corresponds to Example 7.2.3 in Lawless (1982, page 367). The input data is printed in the output. The model is given as:

$$\ln(\lambda)=\beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \alpha_i + \gamma_j$$

where α_i and γ_j correspond to dummy variables generated from columns 6 and 7 of X, respectively, x_1 corresponds to column 3 of X, x_2 corresponds to column 4 of X, and x_3 corresponds to column 5 of X.

```
USE PHGLM_INT
USE WRRRL_INT
INTEGER    ICEN, IPRINT, IRT, LDCASE, LDCOEF, LDCOV, &
          LDX, MAXCL, NCLVAR, NCOL, NEF, NOBS

REAL       RATIO
PARAMETER  (ICEN=2, IPRINT=2, IRT=1, LDCOEF=7, LDX=40, &
           MAXCL=10, NCLVAR=2, NCOL=7, NEF=5, RATIO=10000.0, &
           LDCASE=LDX, LDCOV=LDCOEF, NOBS=LDX)
```

```
!
      INTEGER    IGRP(NOBS), INDCL(NCLVAR), INDEF(5), NCLVAL(NCLVAR), &
                 NCOEF, NRMISS, NVEF(NEF)
      REAL       ALGL, CASE(LDCASE,5), CLVAL(6), COEF(LDCOEF,4), &
                 COV(LDCOV,LDCOV), GR(LDCOV), X(LDX,NCOL), XMEAN(LDCOV)
      CHARACTER  NUMBER(1)*6
      DATA       NUMBER(1)/'NUMBER'/

!
      DATA X/411, 126, 118, 92, 8, 25, 11, 54, 153, 16, 56, 21, 287, &
          10, 8, 12, 177, 12, 200, 250, 100, 999, 231, 991, 1, 201, &
          44, 15, 103, 2, 20, 51, 18, 90, 84, 164, 19, 43, 340, 231, &
          5*0, 1, 16*0, 1, 5*0, 1, 11*0, 7, 6, 7, 4, 4, 7, 7, 8, 6, &
          3, 8, 4, 6, 4, 2, 5, 5, 4, 8, 7, 6, 9, 5, 7, 2, 8, 6, 5, 7, &
          4, 3, 3, 4, 6, 8, 7, 3, 6, 8, 7, 64, 63, 65, 69, 63, 48, &
          48, 63, 63, 53, 43, 55, 66, 67, 61, 63, 66, 68, 41, 53, 37, &
          54, 52, 50, 65, 52, 70, 40, 36, 44, 54, 59, 69, 50, 62, 68, &
          39, 49, 64, 67, 5, 9, 11, 10, 58, 9, 11, 4, 14, 4, 12, 2, &
          25, 23, 19, 4, 16, 12, 12, 8, 13, 12, 8, 7, 21, 28, 13, 13, &
          22, 36, 9, 87, 5, 22, 4, 15, 4, 11, 10, 18, 7*1, 7*2, 2*3, &
          5*4, 7*1, 4*2, 3*3, 5*4, 21*0, 19*1/
      DATA NVEF/1, 1, 1, 1, 1/, INDEF/3, 4, 5, 6, 7/, INDCL/6, 7/
!
      LABEL = 'NUMBER'
      CALL WRRRL ('The First 10 Rows of the Input Data', &
                  X, NUMBER, NUMBER, 10, NCOL, LDX, FMT='(I7)')
!
      CALL PHGLM (X, IRT, NVEF, INDEF, MAXCL, NCOEF, COEF, ALGL, &
                 COV, XMEAN, CASE, GR, IGRP, ICEN=ICEN, RATIO=RATIO,&
                 NCLVAR=NCLVAR, INDCL=INDCL, NEF=NEF, IPRINT=IPRINT, &
                 NCLVAL=NCLVAL, CLVAL=CLVAL, NRMISS=NRMISS)
!
      END
```

Output

```
                The First 10 Rows of the Input Data
          1        2        3        4        5        6        7
 1      411        0        7       64        5        1        0
 2      126        0        6       63        9        1        0
 3      118        0        7       65       11        1        0
 4       92        0        4       69       10        1        0
 5        8        0        4       63       58        1        0
 6       25        1        7       48        9        1        0
 7       11        0        7       48       11        1        0
 8       54        0        8       63        4        2        0
 9      153        0        6       63       14        2        0
10       16        0        3       53        4        2        0

                     Initial Estimates
     1        2        3        4        5        6        7
 0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000

 Method  Iteration  Step size  Maximum scaled     Log
                               coef. update      likelihood
   Q-N        0                                  -102.4
```

Q-N	1	1.0000	0.5034	-91.04
Q-N	2	1.0000	0.5782	-88.07
N-R	3	1.0000	0.1131	-87.92
N-R	4	1.0000	0.6958E-01	-87.89
N-R	5	1.0000	0.8144E-03	-87.89

Log-likelihood -87.88779

Coefficient Statistics

	Coefficient	Standard error	Asymptotic z-statistic	Asymptotic p-value
1	-0.585	0.137	-4.272	0.000
2	-0.013	0.021	-0.634	0.526
3	0.001	0.012	0.064	0.949
4	-0.367	0.485	-0.757	0.449
5	-0.008	0.507	-0.015	0.988
6	1.113	0.633	1.758	0.079
7	0.380	0.406	0.936	0.349

Asymptotic Coefficient Covariance

	1	2	3	4	5
1	0.1873E-01	0.2530E-03	0.3345E-03	0.5745E-02	0.9750E-02
2		0.4235E-03	-0.4120E-04	-0.1663E-02	-0.7954E-03
3			0.1397E-03	0.8111E-03	-0.1831E-02
4				0.2350	0.9799E-01
5					0.2568

	6	7
1	0.4264E-02	0.2082E-02
2	-0.3079E-02	-0.2898E-02
3	0.5995E-03	0.1684E-02
4	0.1184	0.3735E-01
5	0.1253	-0.1944E-01
6	0.4008	0.6289E-01
7		0.1647

Case Analysis

	Survival Probability	Influence	Residual	Cumulative hazard	Proportionality constant
1	0.00	0.04	2.05	6.10	0.3
2	0.30	0.11	0.74	1.21	0.61
3	0.34	0.12	0.36	1.07	0.33
4	0.43	0.16	1.53	0.84	1.83
5	0.96	0.56	0.09	0.05	2.05
6	0.74	NaN	0.13	0.31	0.42
7	0.92	0.37	0.03	0.08	0.42
8	0.59	0.26	0.14	0.53	0.27
9	0.26	0.12	1.20	1.36	0.88
10	0.85	0.15	0.97	0.17	5.76
11	0.55	0.31	0.21	0.60	0.36
12	0.74	0.21	0.96	0.31	3.12
13	0.03	0.06	3.02	3.53	0.86
14	0.94	0.09	0.17	0.06	2.71
15	0.96	0.16	1.31	0.05	28.89
16	0.89	0.23	0.59	0.12	4.82

```
17          0.18          0.09          2.62          1.71             1.54
18          0.89          0.19          0.33          0.12             2.68
19          0.14          0.23          0.72          1.96             0.37
20          0.05          0.09          1.66          2.95             0.56
21          0.39          0.22          1.17          0.94             1.25
22          0.00          0.00          1.73         21.11             0.08
23          0.08           NaN          2.19          2.52             0.87
24          0.00          0.00          2.46          8.89             0.28
25          0.99          0.31          0.05          0.01             4.28
26          0.11          0.17          0.34          2.23             0.15
27          0.66          0.25          0.16          0.41             0.38
28          0.87          0.22          0.15          0.14             1.02
29          0.39           NaN          0.45          0.94             0.48
30          0.98          0.25          0.06          0.02             2.53
31          0.77          0.26          1.03          0.26             3.90
32          0.63          0.35          1.80          0.46             3.88
33          0.82          0.26          1.06          0.19             5.47
34          0.47          0.26          1.65          0.75             2.21
35          0.51          0.32          0.39          0.67             0.58
36          0.22          0.18          0.49          1.53             0.32
37          0.80          0.26          1.08          0.23             4.77
38          0.70          0.16          0.26          0.36             0.73
39          0.01          0.23          0.87          4.66             0.19
40          0.08          0.20          0.81          2.52             0.32

                        Last Coefficient Update
         1             2             3             4             5             6
-1.016E-07   1.918E-09  -1.305E-08  -7.190E-07  -2.854E-07   2.108E-08


         7
-6.947E-08

                    Covariate Means
   1       2       3       4       5       6       7
5.65   56.58   15.65    0.35    0.28    0.12    0.53

Distinct Values For Each Class Variable
Variable  1:     1.0         2.0         3.0         4.0
Variable  2:      0.         1.0

                       Stratum Numbers For Each Observation
1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20
1  1  1  1  1  1  1  1  1   1  1  1  1  1  1  1  1  1  1  1

21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1

Number of Missing Values           0
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `P2GLM/DP2GLM`. The reference is:

```
CALL P2GLM (NOBS, NCOL, X, LDX, IRT, IFRQ, IFIX, ICEN, ISTRAT,
MAXIT, EPS, RATIO, NCLVAR, INDCL, NEF, NVEF, INDEF, INIT, ITIE,
IPRINT, MAXCL, NCLVAL, CLVAL, NCOEF, COEF, LDCOEF, ALGL, COV,
LDCOV, XMEAN, CASE, LDCASE, GR, IGRP, NRMISS, OBS, SMG, SMH,
IPTR, IDT, IWK)
```

The additional arguments are as follows:

OBS — Work vector of length NCOEF + 1.

SMG — Work vector of length NCOEF.

SMH — Work vector of length max(NCOEF * NCOEF, 2).

IPTR — Work vector of length NOBS + NCOEF.

IDT — Work vector of length NOBS.

IWK — Work vector of length 3 * max(NOBS, NCOL)

2. Informational errors

Type	Code	
3	1	Too many iterations required. Convergence assumed.
3	2	Too many step halvings. Convergence assumed.
3	3	Additional strata were formed as required because of the detection of infinite parameter estimates.
4	4	The number of distinct values of the classification variables exceeds MAXCL.
4	5	The model specified by NEF, NVEF, and INDEF yields no covariates.
4	6	After eliminating observations with missing values, no valid observations remain.
4	7	After eliminating observations with missing values, only one covariate vector remains.
4	8	The number of distinct values for each classification variable must be greater than one.
4	9	LDCOEF or LDCOV must be greater or equal to NCOEF.

3. Dummy variables are generated for the classification variables as follows: An ascending list of all distinct values of the classification variable is obtained and stored in CLVAL. Dummy variables are then generated for each but the last of these distinct values. Each dummy variable is zero unless the classification variable equals the list value corresponding to the dummy variable, in which case, the dummy variable is one. See argument IDUMMY for IDUMMY = 2 in routine GRGLM in Chapter 2, Regression.

4. The "product" of a classification variable with a covariate yields dummy variables equal to the product of the covariate with each of the dummy variables associated with the classification variable.

5. The "product" of two classification variables yields dummy variables in the usual manner. Each dummy variable associated with the first classification variable

multiplies each dummy variable associated with the second classification variable. The resulting dummy variables are such that the index of the second classification variable varies fastest.

Description

Routine PHGLM computes parameter estimates and other statistics in Proportional Hazards Generalized Linear Models. These models were first proposed by Cox (1972). Two methods for handling ties are allowed in PHGLM. Time-dependent covariates are not allowed. The user is referred to Cox and Oakes (1984), Kalbfleisch and Prentice (1980), Elandt-Johnson and Johnson (1980), Lee (1980), or Lawless (1982), among other texts, for a thorough discussion of the Cox proportional hazards model.

Let $\lambda(t, z_i)$ represent the hazard rate at time t for observation number i with covariables contained as elements of row vector z_i. The basic assumption in the proportional hazards model (the proportionality assumption) is that the hazard rate can be written as a product of a time varying function $\lambda_0(t)$, which depends only on time, and a function $f(z_i)$, which depends only on the covariable values. The function $f(z_i)$ used in PHGLM is given as $f(z_i) = \exp(w_i + \beta z_i)$ where w_i is a fixed constant assigned to the observation, and β is a vector of coefficients to be estimated. With this function one obtains a hazard rate $\lambda(t, z_i) = \lambda_0(t) \exp(w_i + \beta z_i)$. The form of $\lambda_0(t)$ is not important in proportional hazards models.

The constants w_i may be known theoretically. For example, the hazard rate may be proportional to a known length or area, and the w_i can then be determined from this known length or area. Alternatively, the w_i may be used to fix a subset of the coefficients β (say, β_1) at specified values. When w_i is used in this way, constants $w_i = \beta_1 z_{1\,i}$ are used, while the remaining coefficients in β are free to vary in the optimization algorithm. If user-specified constants are not desired, the user should set IFIX to 0 so that $w_i = 0$ will be used.

With this definition of $\lambda(t, z_i)$, the usual partial (or marginal, see Kalbfleisch and Prentice (1980)) likelihood becomes

$$L = \prod_{i=1}^{n_d} \frac{\exp(w_i + \beta z_i)}{\sum_{j \in R(t_i)} \exp(w_j + \beta z_j)}$$

where $R(t_i)$ denotes the set of indices of observations that have not yet failed at time t_i (the risk set), t_i denotes the time of failure for the i-th observation, n_d is the total number of observations that fail. Right-censored observations (i.e., observations that are known to have survived to time t_i, but for which no time of failure is known) are incorporated into the likelihood through the risk set $R(t_i)$. Such observations never appear in the numerator of the likelihood. When ITIE = 0, all observations that are censored at time t_i are not included in $R(t_i)$, while all observations that fail at time t_i are included in $R(t_i)$.

If it can be assumed that the dependence of the hazard rate upon the covariate values remains the same from stratum to stratum, while the time-dependent term, $\lambda_0(t)$, may be different in different strata, then PHGLM allows the incorporation of strata into the likelihood as follows. Let k index the m = NSTRAT strata. Then, the likelihood is given by

$$L_s = \prod_{k=1}^{m}\left[\prod_{i=1}^{n_k}\frac{\exp(w_{ki}+\beta z_{ki})}{\sum_{j\in R(t_{ki})}\exp(w_{kj}+\beta z_{kj})}\right]$$

In PHGLM, the log of the likelihood is maximized with respect to the coefficients β. A quasi-Newton algorithm approximating the Hessian via the matrix of sums of squares and cross products of the first partial derivatives is used in the initial iterations (the "Q-N" method in the output). When the change in the log-likelihood from one iteration to the next is less than 100*EPS, Newton-Raphson iteration is used (the "N-R" method). If, during any iteration, the initial step does not lead to an increase in the log-likelihood, then step halving is employed to find a step that will increase the log-likelihood.

Once the maximum likelihood estimates have been computed, PHGLM computes estimates of a probability associated with each failure. Within stratum k, an estimate of the probability that the i-th observation fails at time t_i given the risk set $R(t_{ki})$ is given by

$$p_{ki} = \frac{\exp(w_{ki}+z_{ki}\beta)}{\sum_{j\in R(t_{ki})}\exp(w_{kj}+z_{kj}\beta)}$$

A diagnostic "influence" or "leverage" statistic is computed for each noncensored observation as:

$$l_{ki} = -g'_{ki}H_s^{-1}g'_{ki}$$

where H_s is the matrix of second partial derivatives of the log-likelihood, and

$$g'_{ki}$$

is computed as:

$$g'_{ki} = z_{ki} - \frac{z_{ki}\exp(w_{ki}+z_{ki}\beta)}{\sum_{j\in R(t_{ki})}\exp(w_{kj}+z_{kj}\beta)}$$

Influence statistics are not computed for censored observations.

A "residual" is computed for each of the input observations according to methods given in Cox and Oakes (1984, page 108). Residuals are computed as

$$r_{ki} = \exp(w_{ki}+z_{ki}\hat{\beta}) \quad \sum_{j\in R(t_{ki})}\frac{d_{kj}}{\sum_{l\in R(t_{kj})}\exp(w_{kl}+z_{kl}\hat{\beta})}$$

where d_{kj} is the number of tied failures in group k at time t_{kj}. Assuming that the proportional hazards assumption holds, the residuals should approximate a random sample (with censoring) from the unit exponential distribution. By subtracting the expected values, centered residuals can be obtained. (The j-th expected order statistic from the unit exponential with censoring is given as

$$e_j = \sum_{l\le j}\frac{1}{h-l+1}$$

where h is the sample size, and censored observations are not included in the summation.)

An estimate of the cumulative baseline hazard within group k is given as

$$\hat{H}_{k0}(t_{ik}) = \sum_{t_{kj} \le t_{ki}} \frac{d_{kj}}{\sum_{l \in R(t_{kj})} \exp(w_{kl} + z_{kl}\hat{\beta})}$$

The observation proportionality constant is computed as

$$\exp(w_{ki} + z_{ki}\hat{\beta})$$

Programming Notes

1. The covariate vectors z_{ki} are computed from each row of the input matrix X via routine GRGLM (see Chapter 2, Regression). Thus, class variables are easily incorporated into the z_{ki}. The reader is referred to the document for GRGLM in the regression chapter for a more detailed discussion. Note that PHGLM calls GRGLM with the option IDUMMY = 2.

2. The average of each of the explanatory variables is subtracted from the variable prior to computing the product $z_{ki}\beta$. Subtraction of the mean values has no effect on the computed log-likelihood or the estimates since the constant term occurs in both the numerator and denominator of the likelihood. Subtracting the mean values does help to avoid invalid exponentiation in the algorithm and may also speed convergence.

3. Routine PHGLM allows for two methods of handling ties. In the first method (ITIE = 1), the user is allowed to break ties in any manner desired. When this method is used, it is assumed that the user has sorted the rows in X from largest to smallest with respect to the failure/censoring times X(i, IRT) within each stratum (and across strata), with tied observations (failures or censored) broken in the manner desired. The same effect can be obtained with ITIE = 0 by adding (or subtracting) a small amount from each of the tied observations failure/ censoring times t_i = X(i, IRT) so as to break the ties in the desired manner.

   The second method for handling ties (ITIE = 0) uses an approximation for the tied likelihood proposed by Breslow (1974). The likelihood in Breslow's method is as specified above, with the risk set at time tiincluding all observations that fail at time t_i, while all observations that are censored at time tiare not included. (Tied censored observations are assumed to be censored immediately prior to the time t_i).

4. If INIT = 1, then it is assumed that the user has provided initial estimates for the model coefficients β in the first column of the matrix COEF. When initial estimates are provided by the user, care should be taken to ensure that the estimates correspond to the generated covariate vector z_{ki}. If INIT = 0, then initial estimates of zero are used for all of the coefficients. This corresponds to no effect from any of the covariate values.

5. If a linear combination of covariates is monotonically increasing or decreasing with increasing failure times, then one or more of the estimated coefficients is infinite and extended maximum likelihood estimates must be computed. Such estimates may be written as

$$\hat{\beta} = \hat{\beta}_f + \rho\hat{\gamma}$$

where $\rho = \infty$ at the supremum of the likelihood so that

$$\hat{\beta}_f$$

is the finite part of the solution. In PHGLM, it is assumed that extended maximum likelihood estimates must be computed if, within any group k, for any time t,

$$\min_{t_{ki}<t} \exp(w_{ki} + z_{ki}\hat{\beta}) > \rho \max_{t_{ki}<t} \exp(w_{ki} + z_{ki}\hat{\beta})$$

where ρ = RATIO is specified by the user. Thus, for example, if $\rho = 10000$, then PHGLM does not compute extended maximum likelihood estimates until the estimated proportionality constant

$$\exp(w_{ki} + z_{ki}\hat{\beta})$$

is 10000 times larger for all observations prior to t than for all observations after t. When this occurs, PHGLM computes estimates for

$$\hat{\beta}_f$$

by splitting the failures in stratum k into two strata at t (see Bryson and Johnson 1981). Censored observations in stratum k are placed into a stratum based upon the associated value for

$$\exp(w_{ki} + z_{ki}\hat{\beta})$$

The results of the splitting are returned in IGRP.

The estimates

$$\hat{\beta}_f$$

based upon the stratified likelihood represent the finite part of the extended maximum likelihood solution. Routine PHGLM does not compute

$$\hat{\gamma}$$

explicitly, but an estimate for

$$\hat{\gamma}$$

may be obtained in some circumstances by setting RATIO = –1 and optimizing the log-likelihood without forming additional strata. The solution

$$\hat{\beta}$$

obtained will be such that

$$\hat{\beta} = \hat{\beta}_f + \rho\hat{\gamma}$$

for some finite value of $\rho > 0$. At this solution, the Newton-Raphson algorithm will not have "converged" because the Newton-Raphson step sizes returned in GR will be large, at least for some variables. Convergence will be declared, however, because the relative change in the log-likelihood during the final iterations will be small.

Additional Example

Example 2

This example illustrates the use of PHGLM when there are strata present in the data. The observations from Example 1 are arbitrarily grouped into four strata (the first ten observations form stratum 1, the next 10 for stratum 2, etc.). Otherwise, the problem is unchanged. The resulting coefficients are very similar to those obtained when there is no stratification variable. The model is the same as in Example 1.

```
      USE PHGLM_INT
      INTEGER    LDCASE, LDCOEF, LDCOV, LDX, MAXCL, NCLVAR, NCOL, NEF, &
                 NOBS
      REAL       RATIO
      PARAMETER  (LDCOEF=7, LDX=40, MAXCL=10, NCLVAR=2, NCOL=8, NEF=5, &
                LDCASE=LDX, LDCOV=LDCOEF, NOBS=LDX, RATIO=10000.0)
!                                SPECIFICATIONS FOR PARAMETERS
      INTEGER    ICEN, IPRINT, IRT, ISTRAT
      PARAMETER  (ICEN=2, IPRINT=2, IRT=1, ISTRAT=8)
!                                SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER    IGRP(NOBS), NCLVAL(NCLVAR), NCOEF, NRMISS
      REAL       ALGL, CASE(LDCASE,5), CLVAL(6), COEF(LDCOEF,4), &
                COV(LDCOV,LDCOV), GR(LDCOV), XMEAN(LDCOV)
!                                SPECIFICATIONS FOR SAVE VARIABLES
      INTEGER    INDCL(NCLVAR), INDEF(NEF), NVEF(NEF)
      REAL       X(LDX,NCOL)
      SAVE       INDCL, INDEF, NVEF, X
!                                SPECIFICATIONS FOR SUBROUTINES
!
      DATA X/411, 126, 118, 92, 8, 25, 11, 54, 153, 16, 56, 21, 287, &
          10, 8, 12, 177, 12, 200, 250, 100, 999, 231, 991, 1, 201, &
          44, 15, 103, 2, 20, 51, 18, 90, 84, 164, 19, 43, 340, 231, &
          5*0, 1, 16*0, 1, 5*0, 1, 11*0, 7, 6, 7, 4, 4, 7, 7, 8, 6, &
          3, 8, 4, 6, 4, 2, 5, 5, 4, 8, 7, 6, 9, 5, 7, 2, 8, 6, 5, 7, &
          4, 3, 3, 4, 6, 8, 7, 3, 6, 8, 7, 64, 63, 65, 69, 63, 48, &
          48, 63, 63, 53, 43, 55, 66, 67, 61, 63, 66, 68, 41, 53, 37, &
          54, 52, 50, 65, 52, 70, 40, 36, 44, 54, 59, 69, 50, 62, 68, &
          39, 49, 64, 67, 5, 9, 11, 10, 58, 9, 11, 4, 14, 4, 12, 2, &
          25, 23, 19, 4, 16, 12, 12, 8, 13, 12, 8, 7, 21, 28, 13, 13, &
          22, 36, 9, 87, 5, 22, 4, 15, 4, 11, 10, 18, 7*1, 7*2, 2*3, &
          5*4, 7*1, 4*2, 3*3, 5*4, 21*0, 19*1, 10*1, 10*2, 10*3, 10*4/
      DATA NVEF/1, 1, 1, 1, 1/, INDEF/3, 4, 5, 6, 7/, INDCL/6, 7/
!
     CALL PHGLM (X, IRT, NVEF, INDEF, MAXCL, NCOEF,COEF, ALGL, COV, XMEAN,&
                 CASE, GR, IGRP, ICEN=ICEN, ISTRAT=ISTRAT, RATIO=RATIO, &
```

```
                NCLVAR=NCLVAR, INDCL=INDCL, IPRINT=IPRINT, NCLVAL=NCLVAL, &
                CLVAL=CLVAL, NRMISS=NRMISS)
!
      END
```

Output

```
                    Initial Estimates
     1        2        3        4        5        6        7
0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000

Method  Iteration  Step size  Maximum scaled     Log
                               coef. update      likelihood
   Q-N        0                                   -55.90
   Q-N        1      1.0000      0.6748           -45.79
   Q-N        2      1.0000      0.7105           -42.85
   N-R        3      1.0000      0.2315           -42.59
   N-R        4      1.0000      0.1674           -42.55
   N-R        5      1.0000      0.3372E-02       -42.55

 Log-likelihood       -42.54570

                Coefficient Statistics
                    Standard    Asymptotic    Asymptotic
    Coefficient         error   z-statistic       p-value
1        -0.716         0.170        -4.222         0.000
2        -0.033         0.030        -1.122         0.262
3         0.001         0.015         0.048         0.961
4        -0.100         0.999        -0.100         0.921
5        -0.405         0.729        -0.555         0.579
6         1.136         0.769         1.478         0.139
7        -0.087         1.454        -0.060         0.952

Asymptotic Coefficient Covariance
              1             2             3             4             5
1    0.2877E-01    0.8662E-03    0.3119E-03    0.5057E-02    0.2480E-01
2                  0.8842E-03   -0.8137E-04   -0.7623E-02   -0.6925E-03
3                                0.2158E-03   -0.2567E-02   -0.3738E-02
4                                              0.9975        0.5109
5                                                            0.5319

               6             7
1   -0.7669E-02    0.6405E-02
2   -0.8800E-03    0.4120E-02
3    0.1170E-02   -0.3699E-02
4    0.1944        0.8056
5    0.1802        0.4905
6    0.5909        0.1858
7                  2.114

                           Case Analysis
       Survival                                 Cumulative  Proportionality
    Probability     Influence      Residual         hazard         constant
 1         0.00          0.00          2.01          7.83             0.26
 2         0.09          0.06          1.32          2.42             0.55
 3         0.20          0.04          0.40          1.59             0.25
```

```
 4          0.40          0.04          1.69          0.91          1.87
 5          0.92          0.47          0.21          0.09          2.36
 6          0.73           NaN          0.14          0.31          0.44
 7          0.82          0.47          0.09          0.20          0.44
 8          0.55          0.67          0.06          0.61          0.10
 9          0.02          0.07          1.59          3.94          0.40
10          0.73          0.10          1.50          0.31          4.79
11          0.39          0.68          0.17          0.93          0.19
12          0.60          0.14          1.12          0.51          2.19
13          0.00          0.00          2.32          6.32          0.37
14          0.90          0.16          0.15          0.10          1.49
15          0.98          0.04          0.75          0.02         35.42
16          0.75          0.21          1.12          0.29          3.83
17          0.25          0.07          1.55          1.39          1.12
18          0.75          0.21          0.63          0.29          2.14
19          0.10          0.18          0.69          2.31          0.30
20          0.03          0.11          1.48          3.60          0.41
21          0.50          0.61          1.00          0.70          1.44
22          0.00          0.00          1.28         13.59          0.09
23          0.33           NaN          1.92          1.09          1.76
24          0.05          0.00          1.32          2.94          0.45
25          0.95          0.15          0.47          0.05          9.84
26          0.33          0.24          0.23          1.09          0.21
27          0.62          0.40          0.22          0.47          0.47
28          0.76          0.13          0.71          0.27          2.63
29          0.50           NaN          0.37          0.70          0.53
30          0.87          0.23          0.49          0.14          3.53
31          0.88          0.35          0.67          0.13          5.07
32          0.71          0.22          1.56          0.34          4.54
33          0.97          0.52          0.20          0.03          7.00
34          0.44          0.03          2.64          0.83          3.19
35          0.56          0.20          0.29          0.57          0.50
36          0.11          0.00          0.61          2.24          0.27
37          0.94          0.19          0.82          0.07         12.50
38          0.79          0.43          0.24          0.23          1.05
39          0.00          0.00          1.69         11.13          0.15
40          0.01          0.00          1.28          4.54          0.28

                        Last Coefficient Update
          1             2             3             4             5             6
 -7.363E-07    8.762E-09    1.252E-08   -1.697E-06   -1.642E-06    1.075E-06

          7
 -1.772E-06

                    Covariate Means
    1       2       3       4       5       6       7
 5.65   56.58   15.65    0.35    0.28    0.12    0.53

 Distinct Values For Each Class Variable
 Variable  1:     1.0         2.0         3.0         4.0
 Variable  2:      0.         1.0

                      Stratum Numbers For Each Observation
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  2  2  2  2  2
```

```
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
 3  3  3  3  3  3  3  3  3  3  4  4  4  4  4  4  4  4  4   4

Number of Missing Values           0
```

SVGLM

Analyzes censored survival data using a generalized linear model.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

MODEL — Model option parameter. (Input)
MODEL specifies the distribution of the response variable and the relationship of the linear model to a distribution parameter.

MODEL	Distribution
0	Exponential
1	Linear hazard
2	Log-normal
3	Normal
4	Log-logistic
5	Logistic
6	Log least extreme value
7	Least extreme value
8	Log extreme value
9	Extreme value
10	Weibull

For further discussion of the models and parameterizations used, see the Description section.

ILT — For interval-censored and left-censored observations, the column number in X that contains the upper endpoint of the failure interval. (Input)
See argument ICEN. If ILT = 0, left-censored and interval-censored observations cannot be input.

IRT — For interval-censored and right-censored observations, the column number in X that contains the lower endpoint of the failure interval. (Input)
For exact-failure observations, X(*i*, IRT) contains the exact-failure time. IRT must not be zero. See argument ICEN.

MAXCL — An upper bound on the sum of the number of distinct values taken by the classification variables. (Input)

NCOEF — Number of estimated coefficients in the model. (Output, if INIT = 0; input, if INIT = 1)

COEF — NCOEF by 4 matrix containing parameter estimates and associated statistics. (Output, if INIT = 0; input/output, if INIT = 1; input, if MAXIT = 0)

Col.	Statistic
1	Coefficient estimate.
2	Estimated standard deviation of the estimated coefficient.
3	Asymptotic normal score for testing that the coefficient is zero.
4	*p*-value associated with the normal score in column 3.

When COEF is input, only column 1 is referenced as input data, and columns 2 to 4 need not be set. When present in the model, the initial coefficient in COEF estimates a "nuisance" parameter, and the remaining coefficients estimate parameters associated with the "linear" model, beginning with the intercept, if present. Nuisance parameters are as follows:

Model	Nuisance Parameter
1	Coefficient of the quadratic term in time, θ
2 – 9	Scale parameter, σ
10	Shape parameter, θ

ALGL — Maximized log-likelihood. (Output)

COV — NCOEF by NCOEF matrix containing the estimated asymptotic covariance matrix of the coefficients. (Output)
COV is computed as the inverse of the matrix of second partial derivatives of negative one times the log-likelihood. When MAXIT = 0, COV is computed at the initial estimates.

XMEAN — Vector of length NCOEF containing the means of the design variables. (Output)

CASE — NOBS by 5 vector containing the case analysis. (Output)

Col.	Statistic
1	Estimated predicted value
2	Estimated influence or leverage
3	Residual estimate
4	Estimated cumulative hazard
5	For non-censored observations, the estimated density at the observation failure time and covariate values. For censored observations, the corresponding estimated probability.

If MAXIT = 0, CASE is a NOBS by 1 vector containing the estimated probability (for censored observations) or the estimated density (for non censored observations).

GR — Vector of length NCOEF containing the last parameter updates, excluding step halvings. (Output)
GR is computed as the inverse of the matrix of second partial derivatives times the vector of first partial derivatives of the log-likelihood. When MAXIT = 0, the derivatives are computed at the initial estimates.

IADDS — Vector of length NOBS indicating which observations have and have not been included in the model. (Output, if MAXIT > 0; input/output, if MAXIT = 0)

Value	Status of Observation
0	Observation *i* has been included in the model.
1	Observation *i* has not been included in the model due to missing values in the X matrix.
2	Observation *i* has not been included in the model because of infinite estimates in extended maximum likelihood estimation. If MAXIT = 0, then the IADDS array must be initialized prior to calling SVGLM.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Column number in X containing the frequency of response for each observation. (Input)
If IFRQ = 0, a response frequency of 1 for each observation is assumed.
Default: IFRQ = 0.

IFIX — Column number in X containing a constant to be added to the linear response. (Input)
Default: IFIX = 0.
The estimated linear response is taken to be $w_i + z_i\hat{\beta}$
where w_i is the observation constant, z_i is the observation design vector, $\hat{\beta}$

is the vector of estimated parameters output in the first column of COEF, and i indexes the observations. The "fixed" constant allows one to test hypotheses about parameters via the log-likelihoods. If IFIX = 0, the fixed parameter is assumed to be 0.

ICEN — Column number in X containing the censoring code for each observation. (Input)
Default: ICEN = 0.
If ICEN = 0, a censoring code of 0 is assumed. Valid censoring codes are:

X(*i*, ICEN)	**Censoring**
0	Exact failure at X(i, IRT).
1	Right censored. The response is greater than X(i, IRT).
2	Left censored. The response is less than or equal to X(i, ILT).
3	Interval censored. The response is greater than X(i, IRT), but less than or equal to X(i, ILT).

INFIN — Method to be used for handling infinite estimates. (Input)
Default: INFIN = 0.

INFIN	**Method**
0	Remove a rightor left-censored observation from the loglikelihood whenever the probability of the observation exceeds 0.995. At convergence, use linear programming to check that all removed observations actually have infinite linear response

$$z_i\hat{\beta}$$

Set IADDS(i) for observation i to 2 if the linear response is infinite. If not all removed observations have infinite linear response, recompute the estimates based upon the observations with finite

$$z_i \hat{\beta}$$

1 Iterate without checking for infinite estimates.

See the algorithm section for more discussion.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 30 will usually be sufficient. Use MAXIT = 0 to compute the Hessian and score vector at the initial estimates.
Default: MAXIT = 30.

EPS — Convergence criterion. (Input)
Convergence is assumed when the maximum relative change in any coefficient estimate is less than EPS from one iteration to the next, or when the relative change in the log-likelihood, ALGL, from one iteration to the next is less than EPS/100. If EPS is negative, EPS = 0.001 is assumed.
Default: EPS = 0.001.

INTCEP — Intercept option. (Input)
Default: INTCEP = 1.

INTCEP	Action
0	No intercept is in the model (unless otherwise provided for by the user).
1	An intercept is automatically included in the model.

NCLVAR — Number of classification variables. (Input)
Dummy or indicator variables are generated for classification variables using the IDUMMY = 2 option of routine GRGLM (see Chapter 2, Regression). See Comment 3.
Default: NCLVAR = 0.

INDCL — Index vector of length NCLVAR containing the column numbers of X that are classification variables. (Input, if NCLVAR is positive, not used otherwise)
If NCLVAR is 0, INDCL is not referenced and can be dimensioned of length 1 in the calling program.

NEF — Number of effects in the model. (Input)
In addition to effects involving classification variables, simple covariates and the product of simple covariates are also considered effects.
Default: NEF = 0.

NVEF — Vector of length NEF containing the number of variables associated with each effect in the model. (Input, if NEF is positive; not used otherwise)

If NEF is zero, NVEF is not used and can be dimensioned of length 1 in the calling program.

INDEF — Index vector of length NVEF(1) + NVEF(2) + … + NVEF(NEF) containing the column numbers in X associated with each effect. (Input, if NEF is positive; not used otherwise)
The first NVEF(1) elements of INDEF give the column numbers in X of the variables in the first effect. The next NVEF(2) elements of INDEF give the column numbers for the second effect, etc. If NEF is zero, INDEF is not used and can be dimensioned of length one in the calling program.

INIT — Initialization option. (Input)
Default: INIT = 0.

INIT	Action
0	Unweighted linear regression is used to obtain initial estimates.
1	The NCOEF elements in the first column of COEF contain initial estimates of the parameters on input to SVGLM (requiring that the user know NCOEF prior to calling SVGLM).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing is performed, but observational statistics are not printed.
2	All output statistics are printed.

NCLVAL — Vector of length NCLVAR containing the number of values taken by each classification variable. (Output, if NCLVAR is positive; not used otherwise)
NCLVAL(*i*) is the number of distinct values for the *i*-th classification variable. If NCLVAR is zero, NCLVAL is not used and can be dimensioned of length 1 in the calling program.

CLVAL — Vector of length NCLVAL(1) + NCLVAL(2) + … + NCLVAL(NCLVAR) containing the distinct values of the classification variables in ascending order. (Output, if NCLVAR is positive, not used otherwise)
The first NCLVAL(1) elements contain the values for the first classification variables, the next NCLVAL(2) elements contain the values for the second classification variable, etc. If NCLVAR is zero, then CLVAL is not referenced and can be dimensioned of length 1 in the calling program.

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOEF = size (COEF,1).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

LDCASE — Leading dimension of CASE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCASE = size (CASE,1).

NRMISS — Number of rows of data in X that contain missing values in one or more columns ILT, IRT, IFRQ, ICOEF, ICEN, INDCL or INDEF of X. (Output)

FORTRAN 90 Interface

Generic: `CALL SVGLM (X, MODEL, ILT, IRT, MAXCL, NCOEF, COEF, ALGL, COV, XMEAN, CASE, GR, IADDS [,…])`

Specific: The specific interface names are S_SVGLM and D_SVGLM.

FORTRAN 77 Interface

Single:
```
CALL SVGLM (NOBS, NCOL, X, LDX, MODEL, ILT, IRT, IFRQ,
IFIX, ICEN, INFIN, MAXIT, EPS, INTCEP, NCLVAR, INDCL, NEF,
NVEF, INDEF, INIT, IPRINT, MAXCL, NCLVAL, CLVAL, NCOEF,
COEF, LDCOEF, ALGL, COV, LDCOV, XMEAN, CASE, LDCASE, GR,
IADDS, NRMISS)
```

Double: The double precision name is DSVGLM.

Example 1

This example is from Lawless (1982, page 287) and involves the mortality of patients suffering from lung cancer. (The first ten rows of the input data are printed in the output.) An exponential distribution is fit for model

$$\eta = \mu + \beta_1 x_3 + \beta_2 x_4 + \beta_3 x_5 + \alpha_i + \gamma_k$$

where α_i is associated with a classification variable with 4 levels, and γ_k is associated with a classification variable with 2 levels. Note that because the computations are performed in single precision, there will be some small variation in the estimated coefficients across different machine environments.

```
USE SVGLM_INT
USE WRRRL_INT
INTEGER    ICEN, ILT, IPAR, IPRINT, IRT, LDCASE, LDCOEF, &
           LDCOV, LDX, MAXCL, MODEL, NCLVAR, NCOL, NEF, NOBS
PARAMETER  (ICEN=2, ILT=0, IPAR=0, IPRINT=2, IRT=1, LDCASE=40, &
           LDCOEF=8, LDCOV=8, LDX=40, MAXCL=6, MODEL=0, &
```

```
                NCLVAR=2, NCOL=7, NEF=5, NOBS=40)
      CHARACTER  *6 NUMBER(1)
!
      INTEGER    IADDS(NOBS), INDCL(NCLVAR), INDEF(5), NCLVAL(NCLVAR), &
                 NCOEF, NRMISS, NVEF(NEF)
      REAL       ALGL, CASE(LDCASE,5), CLVAL(MAXCL), COEF(LDCOEF,4), &
                COV(LDCOV,LDCOV), GR(LDCOV), X(LDX,NCOL), XMEAN(LDCOV)
!
      DATA X/411, 126, 118, 92, 8, 25, 11, 54, 153, 16, 56, 21, 287, &
          10, 8, 12, 177, 12, 200, 250, 100, 999, 231, 991, 1, 201, &
          44, 15, 103, 2, 20, 51, 18, 90, 84, 164, 19, 43, 340, 231, &
          5*0, 1, 16*0, 1, 5*0, 1, 11*0, 7, 6, 7, 4, 4, 7, 7, 8, 6, &
          3, 8, 4, 6, 4, 2, 5, 5, 4, 8, 7, 6, 9, 5, 7, 2, 8, 6, 5, 7, &
          4, 3, 3, 4, 6, 8, 7, 3, 6, 8, 7, 64, 63, 65, 69, 63, 48, &
          48, 63, 63, 53, 43, 55, 66, 67, 61, 63, 66, 68, 41, 53, 37, &
          54, 52, 50, 65, 52, 70, 40, 36, 44, 54, 59, 69, 50, 62, 68, &
          39, 49, 64, 67, 5, 9, 11, 10, 58, 9, 11, 4, 14, 4, 12, 2, &
          25, 23, 19, 4, 16, 12, 12, 8, 13, 12, 8, 7, 21, 28, 13, 13, &
          22, 36, 9, 87, 5, 22, 4, 15, 4, 11, 10, 18, 7*1, 7*2, 2*3, &
          5*4, 7*1, 4*2, 3*3, 5*4, 21*0, 19*1/
      DATA NVEF/1, 1, 1, 1, 1/, INDEF/3, 4, 5, 6, 7/, INDCL/6, 7/
      NUMBER(1) = 'NUMBER'
!
      CALL WRRRL ('First 10 rows of the input data.', X, &
                  NUMBER, NUMBER, 10, NCOL, LDX)
!
      CALL SVGLM (X, MODEL, ILT, IRT, MAXCL, NCOEF, COEF, ALGL, COV, &
                  XMEAN, CASE, GR, IADDS, ICEN=ICEN, NCLVAR=NCLVAR, &
                  INDCL=INDCL, NEF=NEF, NVEF=NVEF, INDEF=INDEF, &
                  IPRINT=IPRINT, NCLVAL=NCLVAL, CLVAL=CLVAL, &
                  NRMISS=NRMISS)
!
      END
```

Output

```
          First 10 rows of the input data.
        1      2      3      4      5      6      7
 1  411.0    0.0    7.0   64.0    5.0    1.0    0.0
 2  126.0    0.0    6.0   63.0    9.0    1.0    0.0
 3  118.0    0.0    7.0   65.0   11.0    1.0    0.0
 4   92.0    0.0    4.0   69.0   10.0    1.0    0.0
 5    8.0    0.0    4.0   63.0   58.0    1.0    0.0
 6   25.0    1.0    7.0   48.0    9.0    1.0    0.0
 7   11.0    0.0    7.0   48.0   11.0    1.0    0.0
 8   54.0    0.0    8.0   63.0    4.0    2.0    0.0
 9  153.0    0.0    6.0   63.0   14.0    2.0    0.0
10   16.0    0.0    3.0   53.0    4.0    2.0    0.0

                      Initial Estimates
     1       2       3       4       5       6       7       8
-5.054   0.000   0.000   0.000   0.000   0.000   0.000   0.000

Method  Iteration  Step size  Maximum scaled     Log
                              coef. update      likelihood
   Q-N        0                                  -224.0
```

Q-N	1	1.0000	0.9839	-213.4
N-R	2	1.0000	3.603	-207.3
N-R	3	1.0000	10.12	-204.3
N-R	4	1.0000	0.1430	-204.1
N-R	5	1.0000	0.1174E-01	-204.1

Log-likelihood -204.1392

Coefficient Statistics

	Coefficient	Standard error	Asymptotic z-statistic	Asymptotic p-value
1	-1.103	1.314	-0.8939	0.4016
2	-0.540	0.108	-4.995	0.0000
3	-0.009	0.020	-0.459	0.6460
4	-0.003	0.012	-0.291	0.7710
5	-0.363	0.445	-0.816	0.4149
6	0.127	0.486	0.261	0.7939
7	0.869	0.586	1.483	0.1385
8	0.270	0.388	0.695	0.4873

Asymptotic Coefficient Covariance

	1	2	3	4	5
1	1.727	-8.1873E-02	-1.9753E-02	-2.2481E-03	-6.5707E-02
2		1.1690E-02	6.4506E-05	2.8955E-04	-3.8734E-04
3			3.8676E-04	-3.9067E-05	-1.2359E-03
4				1.3630E-04	7.5656E-04
5					0.1976

	6	7	8
1	-0.1038	-0.1554	-4.2370E-05
2	8.5772E-03	1.8120E-02	6.5272E-03
3	-3.2789E-04	-1.6986E-03	-2.7794E-03
4	-1.6742E-03	6.2668E-04	1.5432E-03
5	9.0035E-02	0.1122	4.3157E-02
6	0.2365	0.1142	-1.3527E-02
7		0.3436	5.1948E-02
8			0.1507

Case Analysis

	Predicted	Influence	Residual	Cumulative Hazard	Density or Probability
1	262.7	0.0450	-0.565	1.565	0.0008
2	153.8	0.0042	0.181	0.819	0.0029
3	270.5	0.0482	0.564	0.436	0.0024
4	55.3	0.0844	-0.663	1.663	0.0034
5	61.7	0.3765	0.870	0.130	0.0142
6	230.4	0.0025	-0.108	0.108	0.8972
7	232.0	0.1960	0.953	0.047	0.0041
8	272.8	0.1677	0.802	0.198	0.0030
9	95.9	0.0505	-0.596	1.596	0.0021
10	16.8	0.0005	0.045	0.955	0.0230
11	234.0	0.1911	0.761	0.239	0.0034
12	29.1	0.0156	0.278	0.722	0.0167
13	102.2	0.4609	-1.807	2.807	0.0006
14	34.8	0.0686	0.713	0.287	0.0216

```
15          5.3         0.0838         -0.521          1.521         0.0415
16         25.7         0.0711          0.533          0.467         0.0244
17         65.6         0.4185         -1.698          2.698         0.0010
18         38.4         0.0886          0.688          0.312         0.0191
19        261.0         0.0155          0.234          0.766         0.0018
20        167.2         0.0338         -0.495          1.495         0.0013
21         85.8         0.0082         -0.166          1.166         0.0036
22        947.8         0.0005         -0.054          1.054         0.0004
23        105.9         0.6402         -2.181          2.181         0.1129
24        305.2         0.5757         -2.247          3.247         0.0001
25         24.6         0.3203          0.959          0.041         0.0390
26        572.8         0.0762          0.649          0.351         0.0012
27        217.5         0.1246          0.798          0.202         0.0038
28         96.6         0.1494          0.845          0.155         0.0089
29        173.4         0.1096         -0.594          0.594         0.5522
30         38.7         0.1928          0.948          0.052         0.0245
31         22.5         0.0040          0.112          0.888         0.0183
32         30.7         0.2270         -0.661          1.661         0.0062
33         20.8         0.0058          0.134          0.866         0.0202
34         54.6         0.1094         -0.648          1.648         0.0035
35        168.6         0.0923          0.502          0.498         0.0036
36        256.8         0.0341          0.361          0.639         0.0021
37         21.9         0.0069          0.134          0.866         0.0192
38        124.3         0.0680          0.654          0.346         0.0057
39        417.9         0.0087          0.186          0.814         0.0011
40        257.1         0.0025          0.101          0.899         0.0016

                          Last Coefficient Update
          1           2           3           4           5           6
-1.031E-05  -1.437E-06   3.098E-07   4.722E-08  -1.844E-05  -1.671E-06

          7           8
-2.520E-06   8.139E-06

                    Covariate Means
   1       2       3       4       5       6       7
5.65   56.58   15.65    0.35    0.28    0.12    0.53

         Distinct Values For Each Class Variable
 Variable  1:     1.0         2.0         3.0         4.0
 Variable  2:      0.         1.0

                          Observation Codes
1  2  3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

21 22 23 24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
 0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

Number of Missing Values           0
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `S2GLM/DS2GLM`. The reference is:

```
CALL S2GLM (NOBS, NCOL, X, LDX, MODEL, ILT, IRT,IFRQ, IFIX,
ICEN, INFIN, MAXIT, EPS,INTCEP, NCLVAR, INDCL, NEF, NVEF,
INDEF,INIT,IPRINT, MAXCL, NCLVAL, CLVAL, NCOEF, COEF, LDCOEF,
ALGL, COV, LDCOV, XMEAN, CASE, LDCASE, GR, IADDS, NRMISS, NMAX,
OBS, ADD, XD, WK, KBASIS, RWK, IWK)
```

The additional arguments are as follows:

NMAX — Maximum number of observations that can be handled in the linear programming. (Input)
If workspace is not explicitly provided, NMAX is set to NMAX = (n – 6)/(6 + NCOEF) in SVGLM and NMAX= (n – 6)/(10 + 2 * NCOEF) in DSVGLM, where n is the number of units of workspace remaining after allocating workspace for OBS, RWK, and IWK. If INFIN .EQ. 1, then set NMAX to 0.

OBS — Work array of length 2 * NCOEF + 2.

ADD — Logical work array of length NMAX. If ADD(I) = .TRUE., the I-th observation deleted from the model was returned to the model. ADD is not needed and can be a array of length 1 in the calling program if NMAX = 0.

XD — Work array of length NMAX * NCOEF. XD is not needed and can be a array of length 1 in the calling program if NMAX = 0.

WK — Work array of length 4 * NMAX. WK is not needed and can be a array of length 1 in the calling program if NMAX = 0.

KBASIS — Work array of length NMAX. KBASIS is not needed and can be a array of length 1 in the calling program if NMAX = 0.

RWK — Work array of length 2 * NOBS + 3 * NCOEF.

IWK — Work array of length 5 * NOBS + 6.

2. Informational errors

Type	Code	
3	1	There were too many iterations required. Convergence is assumed.
3	2	There were too many step halvings. Convergence is assumed.
3	4	The censoring interval has a length of 0. The censoring code for this observation is set to 0.
3	5	COEF(1, 1) > 1.0. The expected value for the log logistic distribution, MODEL = 4, does not exist. Predicted values are not calculated.
3	6	COEF(1, 1) > 1.0. The expected value for the log extreme value distribution, MODEL = 8, does not exist. Predicted values are not calculated.
4	7	The number of distinct values of the classification variables exceeds MAXCL. MAXCL must be increased for the computations to proceed.
4	8	The number of distinct values for each classification variable must be greater than one.

4	9	INIT = 1 and the number of coefficients input in NCOEF does not equal the number of coefficients required by the specified model.
4	10	For the exponential model, NCOEF has been determined to equal 0. With no coefficients in the model, processing cannot continue.
4	11	LDCOEF or LDCOV is less than NCOEF.
4	12	NOBS − NRMISS must be greater than or equal to 2 in order to estimate the coefficients.
4	13	The number of observations to be deleted has exceeded NMAX. Rerun with a different model or increase the workspace.

3. Dummy variables are generated for the classification variables as follows: An ascending list of all distinct values of each classification variable is obtained and stored in CLVAL. Dummy variables are then generated for each but the last of these distinct values. Each dummy variable is zero unless the classification variable equals the list value corresponding to the dummy variable, in which case the dummy variable is one. See Argument IDUMMY for IDUMMY = 2 in routine GRGLM (see Chater 2, Regression).

4. The "product" of a classification variable with a covariate yields dummy variables equal to the product of the covariate with each of the dummy variables associated with the classification variable.

5. The "product" of two classification variables yields dummy variables in the usual manner. Each dummy variable associated with the first classification variable multiplies each dummy variable associated with the second classification variable. The resulting dummy variables are such that the index of the second classification variable varies fastest.

Description

Routine SVGLM computes maximum likelihood estimates of parameters and associated statistics in generalized linear models commonly found in survival (reliability) analysis. Although the terminology used will be from the survival area, the methods discussed have application in many areas of data analysis, including reliability analysis and event history analysis. Indeed, these methods may be used anywhere a random variable from one of the discussed distributions is parameterized via one of the models available in SVGLM. Thus, while it is not advisable to do so, standard multiple linear regression may be performed by routine SVGLM. Estimates for any of ten standard models can be computed. Exact, leftcensored, right-censored, or interval-censored observations are allowed. (Note that left censoring is the same as interval censoring with left endpoint equal to the left endpoint of the support of the distribution.)

Let $\eta = x^T\beta$ be the linear parameterization, where x is a design vector obtained in SVGLM via routine GRGLM (see Chapter 2, Regression) from a row of X, and β is a vector of parameters associated with the linear model. Let T denote the random response variable and $S(t)$ denote the probability that $T > t$. All models considered also allow a fixed parameter w_i for observation i (input in column IFIX of X). Use of this parameter is discussed below. There may also be nuisance parameters $\theta > 0$, or $\sigma > 0$ to be estimated (along with β) in the various models. Let Φ denote the cumulative normal distribution. The survival models available in SVGLM are:

Model	Name	$S(t)$
0	Exponential	$\exp\{-\exp(w_i+\eta)\}$
1	Linear hazard	$\exp\{-(t+\frac{\theta t^2}{2})\exp[(w_i+\eta)]\}$
2	Log-normal	$1-\Phi(\frac{\ln(t)-\eta-w_i}{\sigma})$
3	Normal	$1-\Phi(\frac{t-\eta-w_i}{\sigma})$
4	Log-logistic	$\{1+\exp(\frac{\ln(t)-\eta-w_i}{\sigma})\}^{-1}$
5	Logistic	$\{1+\exp(\frac{t-\eta-w_i}{\sigma})\}^{-1}$
6	Log least extreme value	$\exp\{-\exp(\frac{\ln(t)-\eta-w_i}{\sigma})\}$
7	Least extreme value	$\exp\{-\exp(\frac{t-\eta-w_i}{\sigma})\}$
8	Log extreme value	$1-\exp\{-\exp[-\frac{\ln(t)-\eta-w_i}{\sigma})]\}$
9	Extreme value	$1-\exp\{-\exp[-(\frac{t-\eta-w_i}{\sigma})]\}$
10	Weibull	$\exp\{-\left(\frac{t}{\exp(w_i+\eta)}\right)^{\theta}\}$

Note that the log-least-extreme-value model is a reparameterization of the Weibull model. Moreover, models 0, 1, 2, 4, 6, 8, and 10 require that $T > 0$, while all of the remaining models allow any value for T, $-\infty < T < \infty$.

Each row in the data matrix can represent a single observation, or, through the use of column `IFRQ`, it can represent several observations. Classification variables and their products are easily incorporated into the models via the usual GLM type specifications through the use of variables `NCLVAR` and `INDCL`, and the model variables `NEF`, `NVEF`, and `INDEF`.

The constant parameter w_i is input in X and may be used for a number of purposes. For example, if the parameter in an exponential model is known to depend upon the size of the area tested, volume of a radioactive mass, or population density, etc., then a multiplicative factor of the exponential parameter $\lambda = \exp(x\beta)$ may be known apriori. This factor can be input in w_i (w_i is the log of the factor). An alternate use of w_i is as follows: It may be that $\lambda = \exp(x_1\beta_1 + x_2\beta_2)$, where β_2 is known. Letting $w_i = x_2\beta_2$, estimates for β_1 can be obtained via `SVGLM` with the known fixed values for β_2. Standard methods can then be used to test hypotheses about β_2 via computed log-likelihoods.

Computational details

The computations proceed as follows:

1. The input arguments are checked for consistency and validity.

2. Estimates for the means of the explanatory variables x (as generated from the model specification via GRGLM, see Chapter 2, Regression) are computed. Let f_i denote the frequency of the observation. Means are computed as

$$\bar{x} = \frac{\sum_i f_i x_i}{\sum_i f_i}$$

3. If INIT = 0, initial estimates of the parameters for all but the exponential models (MODEL = 0, 1) are are obtained as follows:

   A. Routine KAPMR (page 1063) is used to compute a nonparametric estimate of the survival probability at the upper limit of each failure interval. (Because upper limits are used, intervaland left-censored data are taken to be exact failures at the upper endpoint of the failure interval.) The Kaplan-Meier estimate is computed under the assumption that all failure distributions are identical (i.e., all β's but the intercept, if present, are assumed to be zero).

   B. If INTCEP = 0, a simple linear regression is performed predicting

$$S^{-1}(\hat{S}(t)) - w_i = \alpha + \phi t^*$$

   where t^* is computed at the upper endpoint of each failure interval, $t^* = t$ in models 3, 5, 7, and 9, and $t^* = \ln(t)$ in models 2, 4, 6, 8, and 10, and w_i is the fixed constant, if present. If INTCEP is zero, α is fixed at zero, and the model

$$S^{-1}(\hat{S}(t)) - \hat{\phi} t^* - w_i = x^T \beta$$

   is fit instead of the model above. In this model, the coefficients β are used in place of the location estimate α above. Here,

$$\hat{\phi}$$

   is estimated from the simple linear regression with α = 0.

   C. If the intercept is in the model, then in log-location-scale models (models 1–8),

$$\hat{\sigma} = \hat{\phi}$$

   and the initial estimate of the intercept, if present, is taken to be

$$\hat{\alpha}$$

   In the Weibull model,

$$\hat{\theta} = 1/\hat{\phi}$$

   and the intercept, if present, is taken to be

$$\hat{\alpha}$$

   Initial estimates of all parameters β, other than the intercept, are taken to be zero.

If no intercept is in the model, the scale parameter is estimated as above, and the estimates

$$\hat{\beta}$$

from Step B are used as initial estimates for the β's.

For exponential models (MODEL = 0, 1), the average total time on test statistic is used to obtain an estimate for the intercept. Specifically, let T_t denote the total number of failures divided by the total time on test. The initial estimate for the intercept is then $\ln(T_t)$. Initial estimates for the remaining parameters β are taken as zero, and, if MODEL = 1, the initial estimate for the linear hazard parameter θ is taken to be a small positive number. When the intercept is not in the model, the initial estimate for the parameter θ is taken as a small positive number, and initial estimates of the parameters β are computed via multiple linear regression as above.

4. A quasi-Newton algorithm is used in the initial iterations based upon a Hessian estimate

$$\hat{H}_{\kappa_j \kappa_l} = \sum_i \ell'_{i\alpha_j} \ell'_{i\alpha_l}$$

where

$$\ell'_{i\alpha_j}$$

is the partial derivative of the *i*-th term in the log-likelihood with respect to the parameter α_j, and α_j denotes one of the parameters to be estimated.

When the relative change in the log-likelihood from one iteration to the next is 0.1 or less, exact second partial derivatives are used for the Hessian so that Newton-Raphson iteration is used.

If the initial step size results in an increase in the log-likelihood, the full step is used. If the log-likelihood decreases for the initial step size, the step size is halved, and a check for an increase in the log-likelihood performed. Step-halving is performed (as a simple line search) until an increase in the log-likelihood is detected, or until the step size is less that 0.0001 (where the initial step size is 1).

5. Convergence is assumed when the maximum relative change in any coefficient update from one iteration to the next is less than EPS, or when the relative change in the loglikelihood from one iteration to the next is less than EPS/100. Convergence is also assumed after MAXIT iterations, or when step halving leads to a step size of less than .0001, with no increase in the log-likelihood.

6. If requested (INFIN = 0), then the methods of Clarkson and Jennrich (1988) are used to check for the existence of infinite estimates in

$$\eta_i = x_i^T \beta$$

As an example of a situation in which infinite estimates can occur, suppose that observation j is right censored with $t_j > 15$ in a normal distribution model in which we fit the mean as

$$\mu_j = x_j^T \beta = \eta_j$$

where x_j is the observation design vector. If design vector x_j for parameter β_m is such that $x_{jm} = 1$ and $x_{im} = 0$ for all $i \neq j$, then the optimal estimate of β_m occurs at

$$\hat{\beta}_m = \infty$$

leading to an infinite estimate of both β_m and η_j. In SVGLM, such estimates may be "computed."

In all models fit by SVGLM, infinite estimates can only occur when the optimal estimated probability associated with the leftor right-censored observation is 1. If INFIN = 0, left-or right-censored observations that have estimated probability greater than 0.995 at some point during the iterations are excluded from the log-likelihood, and the iterations proceed with a log-likelihood based upon the remaining observations. This allows convergence of the algorithm when the maximum relative change in the estimated coefficients is small and also allows for a more precise determination of observations with infinite

$$\eta_i = x_i^T \beta$$

At convergence, linear programming is used to ensure that the eliminated observations have infinite η_i. If some (or all) of the removed observations should not have been removed (because their estimated η_i's must be finite), then the iterations are restarted with a log-likelihood based upon the finite η_i observations. See Clarkson and Jennrich (1988) for more details.

When INFIN = 1, no observations are eliminated during the iterations. In this case, when infinite estimates occur, some (or all) of the coefficient estimates

$$\hat{\beta}$$

will become large, and it is likely that the Hessian will become (numerically) singular prior to convergence.

7. The case statistics are computed as follows:

   Let

   $$\ell_i(\theta_i)$$

   denote the log-likelihood of the i-th observation evaluated at θ_i, let

   $$\ell_i'$$

   denote the vector of derivatives of

   $$\ell_i$$

   with respect to all parameters,

   $$\ell'_{\eta,i}$$

denote the derivative of

$$\ell_i$$

with respect to $\eta = x^T\beta$, H denote the Hessian, and E denote expectation. Then, the columns of CASE are:

A. Predicted values are computed as $E(T|x)$ according to standard formulas. If MODEL is 4 or 8, and if $\sigma \geq 1$, then the expected values cannot be computed because they are infinite.

B. Following Cook and Weisberg (1982), we take the influence (or leverage) of the i-th observation to be

$$(\ell'_i)^T H^{-1} \ell'_i$$

This quantity is a one-step approximation to the change in the estimates when the i-th observation is deleted (ignoring the nuisance parameters).

C. The "residual" is computed as

$$\ell'_{\hat{\eta},i}$$

D. The cumulative hazard is computed at the observation covariate values and, for interval observations, the upper endpoint of the failure interval. The cumulative hazard can also be used as a "residual" estimate. If the model is correct, the cumulative hazards should follow a standard exponential distribution. See Cox and Oakes (1984).

E. The density (for exact failures) or the interval probability (for censored observations) is computed for given x.

Programming Notes

Classification variables are specified by parameters NCLVAR and INDCL. Indicator variables are created for the classification variables using routine GRGLM (see Chapter 2, Regression) with IDUMMY = 2.

Additional Example

Example 2

As a second example, the MAXIT = 0 option is used for the model in Example 1 with the coefficients restricted such that $\mu = -1.25$, $\beta_1 = -.6$, and the remaining 6 coefficients are zero. A chi-squared statistic with 8 degrees of freedom for testing that the coefficients are specified as above (versus the alternative that they are not as specified) may be computed from the output as

$$\chi^2 = g^T \hat{\Sigma}^{-1} g$$

where

$$\hat{\Sigma}$$

is output in COV, and g is output in GR. The resulting test statistic (6.107), based upon no iterations, is comparable to the likelihood ratio test statistic that may be computed from the log-likelihood output in Example 2 (–206.6835) and the log-likelihood output in Example 1 (–204.1392).

$$\chi^2_{LR} = 2\left(206.6835 - 204.1392\right) = 5.0886$$

Neither test statistic is significant at the $\alpha = 0.05$ level.

```
      USE IMSL_LIBRARIES
      INTEGER    ICEN, ILT, INIT, IPAR, IPRINT, IRT, LDCASE, &
                 LDCOEF, LDCOV, LDX, MAXCL, MAXIT, MODEL, NCLVAR, &
                 NCOL, NEF, NOBS
      PARAMETER  (ICEN=2, ILT=0, INIT=1, IPAR=0, IPRINT=2,IRT=1, &
                  LDCASE=40, LDCOEF=8, LDCOV=8, LDX=40, MAXCL=6, &
                  MAXIT=0, MODEL=0, NCLVAR=2, NCOL=7, NEF=5, NOBS=40)
!
      INTEGER    IADDS(NOBS), INDCL(NCLVAR), INDEF(5), IRANK, &
                 NCLVAL(NCLVAR), NCOEF, NRMISS, NVEF(NEF)
      REAL       ALGL, CASE(LDCASE,5), CHISQ, CLVAL(MAXCL), &
                 COEF(LDCOEF,4), COV(LDCOV,LDCOV), GR(LDCOV, 1), &
                 GRD(LDCOV, 1), X(LDX,NCOL), XMEAN(LDCOV)
!
      DATA X/411, 126, 118, 92, 8, 25, 11, 54, 153, 16, 56, 21, 287, &
          10, 8, 12, 177, 12, 200, 250, 100, 999, 231, 991, 1, 201, &
          44, 15, 103, 2, 20, 51, 18, 90, 84, 164, 19, 43, 340, 231, &
          5*0, 1, 16*0, 1, 5*0, 1, 11*0, 7, 6, 7, 4, 4, 7, 7, 8, 6, &
          3, 8, 4, 6, 4, 2, 5, 5, 4, 8, 7, 6, 9, 5, 7, 2, 8, 6, 5, 7, &
          4, 3, 3, 4, 6, 8, 7, 3, 6, 8, 7, 64, 63, 65, 69, 63, 48, &
          48, 63, 63, 53, 43, 55, 66, 67, 61, 63, 66, 68, 41, 53, 37, &
          54, 52, 50, 65, 52, 70, 40, 36, 44, 54, 59, 69, 50, 62, 68, &
          39, 49, 64, 67, 5, 9, 11, 10, 58, 9, 11, 4, 14, 4, 12, 2, &
          25, 23, 19, 4, 16, 12, 12, 8, 13, 12, 8, 7, 21, 28, 13, 13, &
          22, 36, 9, 87, 5, 22, 4, 15, 4, 11, 10, 18, 7*1, 7*2, 2*3, &
          5*4, 7*1, 4*2, 3*3, 5*4, 21*0, 19*1/
      DATA NVEF/1, 1, 1, 1, 1/, INDEF/3, 4, 5, 6, 7/, INDCL/6, 7/
!
      NCOEF = 8
      CALL SSET (NCOEF, 0.0, COEF(3:,1), 1)
      CALL ISET (NOBS, 0, IADDS, 1)
      COEF(1,1) = -1.25
      COEF(2,1) = -0.60
      CALL SVGLM (X, MODEL, ILT, IRT, MAXCL, NCOEF, COEF, ALGL, COV, &
                  XMEAN, CASE, GR(1:1), IADDS, ICEN=ICEN, MAXIT=MAXIT, &
                  NCLVAR=NCLVAR, INDCL=INDCL, NEF=NEF, NVEF=NVEF,&
                  INDEF=INDEF, INIT=INIT, IPRINT=IPRINT, &
                  NCLVAL=NCLVAL, CLVAL=CLVAL)
!                              Compute Chi-squared
      CALL CHFAC (COV, IRANK, COV)
      CALL GIRTS (COV, GR, IRANK, GRD, IPATH=2)
!
      CHISQ = SDOT(NCOEF,GRD(1:1), 1, GRD(1:1), 1)
      WRITE (6,99999) ' Chi-squared statistic with 8 degrees of '// &
                   'freedom ', CHISQ
!
```

```
99999 FORMAT (/, A, G12.4)
      END
```

Output

```
Log-likelihood       -206.6835

                  Coefficient Statistics
                      Standard     Asymptotic     Asymptotic
     Coefficient         error    z-statistic        p-value
1          -1.25         1.383         -0.904          0.366
2          -0.60         0.112         -5.365          0.000
3           0.00         0.021          0.000          1.000
4           0.00         0.011          0.000          1.000
5           0.00         0.429          0.000          1.000
6           0.00         0.530          0.000          1.000
7           0.00         0.775          0.000          1.000
8           0.00         0.405          0.000          1.000

                                Hessian
            1              2              3              4              5
1       1.914      -8.1835E-02    -2.3464E-02    -1.1634E-03    -9.0646E-02
2                   1.2507E-02     2.0883E-06     3.1320E-04    -5.3147E-04
3                                  4.6174E-04    -5.5344E-05    -8.1929E-04
4                                                 1.1797E-04     6.0699E-04
5                                                                0.1839

                6              7              8
1     -0.1641        -0.1681         7.7768E-02
2      1.0372E-02     1.9269E-02     5.9762E-03
3      5.1246E-04    -1.6419E-03    -4.0106E-03
4     -2.0693E-03     6.9029E-04     1.7020E-03
5      9.9640E-02     0.1191         3.5786E-02
6      0.2808         0.1264        -2.2602E-02
7                     0.6003         4.6015E-02
8                                    0.1641

 Estimated Probability (censored) or Estimated Density (non-censored)
     1        2        3        4        5        6        7        8
0.0007   0.0029   0.0026   0.0024   0.0211   0.8982   0.0041   0.0021

      9       10       11       12       13       14       15       16
0.00240    .0222   0.0021   0.0151   0.0008   0.0200   0.0433   0.0120

     17       18       19       20       21       22       23       24
0.0011   0.0190   0.0015   0.0015   0.0036   0.0004   0.0371   0.0001

     25       26       27       28       29       30       31       32
0.0792   0.0015   0.0055   0.0115   0.6424   0.0247   0.0184   0.0042

     33       34       35       36       37       38       39       40
0.0163   0.0039   0.0019   0.0021   0.0193   0.0056   0.0011   0.0016

                    Newton-Raphson Step
    1       2       3       4       5       6       7       8
0.171   0.062  -0.011  -0.003  -0.336   0.133   1.297   0.298
```

```
                    Covariate Means
    1       2       3       4       5       6       7
 5.65   56.58   15.65    0.35    0.28    0.12    0.53

 Distinct Values For Each Class Variable
 Variable  1:     1.0         2.0         3.0         4.0
 Variable  2:      0.         1.0

                                  Observation Codes
1  2  3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

21 22 23 24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
 0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

Number of Missing Values           0

Chi-squared statistic with 8 degrees of freedom    6.107
```

STBLE

Estimates survival probabilities and hazard rates for various parametric models.

Required Arguments

XPT — NOBS by NCOL matrix, each row of which contains the covariates for a group for which survival estimates are desired. (Input)

MODEL — Model option parameter. (Input)
MODEL specifies the distribution of the response variable and the relationship of the linear model to a distribution parameter.

MODEL	Distribution
0	Exponential
1	Linear hazard
2	Log-normal
3	Normal
4	Log-logistic
5	Logistic
6	Log least extreme value
7	Least extreme value

8	Log extreme value
9	Extreme value
10	Weibull

For further discussion of the models, see the Algorithm section.

TIME — Beginning of the time grid for which the survival estimates are desired. (Input)
Survival probabilities and hazard rates are computed for each covariate vector over the grid of time points TIME + i * DELTA for i = 0, 1, …, NPT − 1.

NPT — Number of points on the time grid for which survival probabilities are desired. (Input)

DELTA — Increment between time points on the time grid. (Input)

IFIX — Column number in XPT containing a constant to be added to the linear response. (Input)
The estimated linear response is w + COEF(1) * z(1) + COEF(2) * z(2) + … + COEF(NCOEF) * z(NCOEF), where z is the design vector for the I-th observation obtained from a row of XPT. w = XPT(I, IFIX) if IFIX is positive, and w = 0 otherwise.

NCOEF — Number of coefficients in the model. (Input)

COEF — Vector of length NCOEF containing the model parameter estimates. (Input)
Usually routine SVGLM (page 1095) is first called to estimate COEF as the first column of matrix COEF in SVGLM. When present in the model, the initial coefficient in COEF is a "nuisance" parameter, and the remaining coefficients are parameters associated with the "linear" model, beginning with the intercept, if present. Nuisance parameters are as follows:

Model	**Nuisance Parameter**
1	Coefficient of the quadratic term in time, θ
2–9	Scale parameter, σ
10	Shape parameter, θ

There is no nuisance parameter for model 0.

SPROB — NPT by 2 * NOBS + 1 matrix. (Output)
SPROB(i, 2) contains the estimated survival probability at time SPROB(i, 1) = TIME + (i − 1) * DELTA for observations with covariates given in row 1 of XPT. SPROB(i, 3) contains the estimate for the hazard rate at this time point. Columns 4 and 5 contain the estimated survival probabilities and hazard rates for observations with covariates given

in the second row in XPT, etc., up to columns 2 * NOBS and 2 * NOBS + 1, which contain these statistics for observations with covariates in the last row of XPT.

XBETA — Vector of length NOBS containing the estimated linear response w + COEF(1) * z(1) + … + COEF(NCOEF) * z(NCOEF) for each row of XPT. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (XPT,1).

NCOL — Number of columns in XPT. (Input)
Default: NCOL = size (XPT,2).

LDXPT — Leading dimension of XPT exactly as specified in the dimension statement of the calling program. (Input)
Default: LDX = size (XPT,1).

INTCEP — Intercept option. (Input)
Default: INTCEP = 1.

INTCEP	Action
0	No intercept is in the model (unless otherwise provided for by the user).
1	An intercept is automatically included in the model.

NCLVAR — Number of classification variables. (Input)
Dummy or indicator variables are generated for classification variables using the IDUMMY = 2 option of routine GRGLM (see Chapter 2, Regression). See Comment 2.
Default: NCLVAR = 0.

INDCL — Index vector of length NCLVAR containing the column numbers of X that are classification variables. (Input, if NCLVAR is positive, not used otherwise)
If NCLVAR is 0, INDCL is not referenced and can be dimensioned of length 1 in the calling program.

NCLVAL — Vector of length NCLVAR containing the number of values taken on by each classification variable. (Input, if NCLVAR is positive, not referenced otherwise)
NCLVAL(I) is the number of distinct values for the I-th classification variable. NCLVAL is not referenced and can be dimensioned of length 1 in the calling program if NCLVAR is zero.

CLVAL — Vector of length NCLVAL(1) + NCLVAL(2) + … + NCLVAL(NCLVAR) containing the distinct values of the classification variables. (Input, if NCLVAR is positive; not used otherwise)
The first NCLVAL(1) elements contain the values for the first classification variables, the next NCLVAL(2) elements contain the values for the second classification variable,

etc. If NCLVAR is zero, then CLVAL is not referenced and can be dimensioned of length 1 in the calling program.

NEF — Number of effects in the model. (Input)
In addition to effects involving classification variables, simple covariates and the product of simple covariates are also considered effects.
Default: NEF = 0.

NVEF — Vector of length NEF that contains the number of variables associated with each effect. (Input, if NEF is greater than 0; not referenced otherwise)
NVEF is not referenced and can be dimensioned of length 1 in the calling program if NEF is zero.

INDEF — Vector of length NVEF(1) + … + NVEF(NEF) that contains the column numbers in X associated with each effect. (Input, if NEF is greater than 0; not used otherwise)
The first NVEF(1) elements of INDEF contain the column numbers in XPT for the variables in the first effect. The next NVEF(2) elements in INDEF contain the column numbers for the second effect, etc.. If NCLVAR is zero, INDEF is not referenced and can be dimensioned of length 1 in the calling program.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing is performed.

LDSPRO — Leading dimension of SPROB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSPRO = size (SPROB,1).

FORTRAN 90 Interface

Generic: `CALL STBLE (XPT, MODEL, TIME, NPT, DELTA, IFIX, NCOEF, COEF, SPROB, XBETA [,…])`

Specific: The specific interface names are S_STBLE and D_STBLE.

FORTRAN 77 Interface

Single:
```
CALL STBLE (NOBS, NCOL, XPT, LDXPT, MODEL, TIME, NPT,
DELTA, IFIX, INTCEP, NCLVAR, INDCL, NCLVAL, CLVAL, NEF,
NVEF, INDEF, NCOEF, COEF, IPRINT,SPROB, LDSPRO, XBETA)
```

Double: The double precision name is DSTBLE.

Example

The example is a continuation of the first example given for routine SVGLM (page 1095). Prior to calling STBLE, SVGLM is invoked to compute the parameter estimates. The example is taken from Lawless (1982, page 287) and involves the mortality of patients suffering from lung cancer.

```
!
      USE SVGLM_INT
      USE SCOPY_INT
      USE STBLE_INT

      INTEGER    ICEN, IFIX, ILT, INFIN, IPRINT, IRT, LDCASE, &
                 LDCOEF, LDCOV, LDSPRO, LDX, LDXPT, MAXCL, &
                 MODEL, NCLVAR, NCOL, NEF, NOBS, NPT, NOBS
      REAL       DELTA, TIME, XPWR
      PARAMETER  (DELTA=20.0, ICEN=2, IFIX=0, ILT=0, INFIN=0, IPRINT=1, &
                  IRT=1, LDCASE=40, LDCOEF=9, LDCOV=9, LDX=40, LDXPT=2, &
                  MAXCL=6, MODEL=0, NCLVAR=2, NCOL=7, NEF=5, NOBS=40, &
                  NPT=10, TIME=10.0, XPWR=0.0, LDSPRO=NPT)
!
      INTEGER    IADDS(NOBS), INDCL(NCLVAR), INDEF(5), NCLVAL(NCLVAR), &
                NCOEF, NRMISS, NVEF(NEF)
      REAL       ALGL, CASE(LDCASE,5), CLVAL(MAXCL), COEF(LDCOEF,4), &
                COV(LDCOV,LDCOV), GR(LDCOV), SPROB(LDSPRO,2*NOBS+1), &
                X(LDX,NCOL), XBETA(NOBS), XMEAN(LDCOV), XPT(LDXPT,NCOL)
!
      DATA X/411, 126, 118, 92, 8, 25, 11, 54, 153, 16, 56, 21, 287, &
          10, 8, 12, 177, 12, 200, 250, 100, 999, 231, 991, 1, 201, &
          44, 15, 103, 2, 20, 51, 18, 90, 84, 164, 19, 43, 340, 231, &
          5*0, 1, 16*0, 1, 5*0, 1, 11*0, 7, 6, 7, 4, 4, 7, 7, 8, 6, &
          3, 8, 4, 6, 4, 2, 5, 5, 4, 8, 7, 6, 9, 5, 7, 2, 8, 6, 5, 7, &
          4, 3, 3, 4, 6, 8, 7, 3, 6, 8, 7, 64, 63, 65, 69, 63, 48, &
          48, 63, 63, 53, 43, 55, 66, 67, 61, 63, 66, 68, 41, 53, 37, &
          54, 52, 50, 65, 52, 70, 40, 36, 44, 54, 59, 69, 50, 62, 68, &
          39, 49, 64, 67, 5, 9, 11, 10, 58, 9, 11, 4, 14, 4, 12, 2, &
          25, 23, 19, 4, 16, 12, 12, 8, 13, 12, 8, 7, 21, 28, 13, 13, &
          22, 36, 9, 87, 5, 22, 4, 15, 4, 11, 10, 18, 7*1, 7*2, 2*3, &
          5*4, 7*1, 4*2, 3*3, 5*4, 21*0, 19*1/
      DATA NVEF/1, 1, 1, 1, 1/, INDEF/3, 4, 5, 6, 7/, INDCL/6, 7/
!
      CALL SVGLM (X, MODEL, ILT, IRT, MAXCL, NCOEF, COEF, ALGL, &
                  COV, XMEAN, CASE, GR, IADDS, ICEN=ICEN, &
                  NCLVAR=NCLVAR, INDCL=INDCL, NEF=NEF, NVEF=NVEF, &
                  INDEF=INDEF, NCLVAL=NCLVAL, CLVAL=CLVAL)
!
      CALL SCOPY (NCOL, X(1:,1), LDX, XPT(1:,1), LDXPT)
      CALL SCOPY (NCOL, X(2:,1), LDX, XPT(2:,1), LDXPT)
!
      CALL STBLE (XPT, MODEL, TIME, NPT, DELTA, IFIX, &
                  NCOEF, COEF, SPROB, XBETA, NCLVAR=NCLVAR, INDCL=INDCL,&
                  NCLVAL=NCLVAL, CLVAL=CLVAL, NEF=NEF, NVEF=NVEF,&
                  INDEF=INDEF, IPRINT=IPRINT)
!
      END
```

Output

```
                              group   1
                               xpt
  1           2           3           4           5           6
411           0           7          64           5           1

7
0


                     design vector
1           2           3           4           5           6
1           7          64           5           1           0

7           8
0           1

xbeta =        -5.57097

                          group   2
                            xpt
  1           2           3           4           5           6
126           0           6          63           9           1

7
0


                     design vector
1           2           3           4           5           6
1           6          63           9           1           0

7           8
0           1

xbeta =        -5.03551

              survival and hazard estimates
                          (sprob)
  time            s1            h1            s2            h2
 10.00        0.9626      0.003807        0.9370      0.006503
 30.00        0.8921      0.003807        0.8228      0.006503
 50.00        0.8267      0.003807        0.7224      0.006503
 70.00        0.7661      0.003807        0.6343      0.006503
 90.00        0.7099      0.003807        0.5570      0.006503
110.00        0.6579      0.003807        0.4890      0.006503
130.00        0.6096      0.003807        0.4294      0.006503
150.00        0.5649      0.003807        0.3770      0.006503
170.00        0.5235      0.003807        0.3310      0.006503
190.00        0.4852      0.003807        0.2907      0.006503
```

Note that in simple exponential models the hazard rate is constant over time.

Comments

1. Workspace may be explicitly provided, if desired, by use of `S2BLE/DS2BLE`. The reference is:

```
CALL S2BLE (NOBS, NCOL, XPT, LDXPT, MODEL, TIME, NPT, DELTA,
IFIX, INTCEP, NCLVAR, INDCL, NCLVAL,CLVAL, NEF, NVEF, INDEF,
NCOEF, COEF, IPRINT, SPROB, LDSPRO, XBETA, CHWK, Z, RWK)
```

The additional arguments are as follows:

CHWK — CHARACTER * 10 work vector of length NCOL.

Z — Work vector of length NCOEF.

RWK — Work vector of length MAX(7, NCOL) if IPRINT = 1, or of length 1 if IPRINT = 0.

2. Dummy variables are generated for the classification variables as follows: The list of all distinct values of each classification variable is as stored in CLVAL. Dummy variables are generated for each but the last of these distinct values. Each dummy variable is zero unless the classification variable equals the list value corresponding to the dummy variable, in which case the dummy variable is one. See argument IDUMMY for IDUMMY = 2 in routine GRGLM (see Chapter 2, Regression).

3. Informational errors

Type	Code	
3	1	Some survival probabilities are less than or equal to zero. The corresponding hazard values cannot be computed.
4	2	The specified number of coefficients, NCOEF, is incorrect.
4	3	The model specified is not defined for negative time.

Description

Routine STBLE computes estimates of survival probabilities and hazard rates for the parametric survival/reliability models fit by routine SVGLM (page 1095) for one or more vectors of covariate values. Because estimates for the parameters of the model must be given, routine SVGLM is usually invoked to obtain these estimates prior to invoking STBLE.

Let $\eta = x^T\beta$ be the linear parameterization, where x is a design vector obtained in STBLE via routine GRGLM (see Chapter 2, Regression) from a row of XPT, and β is a vector of parameters associated with the linear model. Let T denote the random response variable and $S(t)$ denote the probability that $T > t$. All models considered also allow a fixed parameter w (input in column IFIX of XPT). Use of this parameter is discussed in the document for routine SVGLM . There may also be nuisance parameters $\theta > 0$, or $\sigma > 0$. Let Φ denote the cumulative normal distribution. The survival models available in STBLE are

Model	Name	$S(t)$
0	Exponential	$\exp\{-t\exp(w+\eta)\}$
1	Linear hazard	$\exp\{-(t+\frac{\theta t^2}{2})\exp[(w+\eta)]\}$
2	Log-normal	$1-\Phi(\frac{\ln(t)-\eta-w}{\sigma})$
3	Normal	$1-\Phi(\frac{t-\eta-w}{\sigma})$

4	Log-logistic	$\{1+\exp(\frac{\ln(t)-\eta-w}{\sigma})\}^{-1}$
5	Logistic	$\{1+\exp(\frac{t-\eta-w}{\sigma})\}^{-1}$
6	Log least extreme value	$\exp\{-\exp(\frac{\ln(t)-\eta-w}{\sigma})\}$
7	Least extreme value	$\exp\{-\exp(\frac{t-\eta-w}{\sigma})\}$
8	Log extreme value	$1-\exp\{-\exp[-(\frac{\ln(t)-\eta-w}{\sigma})]\}$
9	Extreme value	$1-\exp\{-\exp[-(\frac{t-\eta-w}{\sigma})]\}$
10	Weibull	$\exp\{-\left(\frac{t}{\exp(w+\eta)}\right)^{\theta}\}$

Let $\lambda(t)$ denote the hazard rate at time t. Then $\lambda(t)$ and $S(t)$ are related as

$$S(t)=\exp\{-\int_{-\infty}^{t}\lambda(s)ds\}$$

Models 0, 1, 2, 4, 6, 8, and 10 require that $T > 0$ (in which case, we assume $\lambda(s) = 0$ for $s < 0$), while the remaining models allow arbitrary values for T, $-\infty < T < \infty$. The computations proceed in routine STBLE as follows:

1. The input arguments are checked for consistency and validity.
2. For each row of XPT, the explanatory variables are generated from the classification and variables and the covariates using routine GRGLM with the IDUMMY = 2 option. (When IDUMMY is two, GRGLM assigns an indicator variable the value 1.0 when the observation is in the class, assigns the value 0.0 otherwise, and omits the last indicator variable from the design vector. See the manual documentation for GRGLM.) Given the explanatary variables x, η is computed as $\eta = x^T\beta$, where β in input in COEF.
3. For each time point requested in the time grid, the survival probabilities and hazard rates are computed.

ACTBL

Produces population and cohort life tables.

Required Arguments

IMTH — Type of life table. (Input)
IMTH = 0 indicates a population (current) table. IMTH = 1 indicates a cohort table.

N — Number of age classes. (Input)

NPOP — Population size. (Input, if IMTH = 0; not used otherwise)
For IMTH = 0, the population size at the beginning of the first age interval. The value is somewhat arbitrary. NPOP = 10000 is reasonable. Not used if IMTH = 1.

AGE — Vector of length N + 1 containing the lowest age in each age interval, and in AGE(N + 1), the endpoint of the last age interval. (Input)
Negative AGE(1) indicates that the age intervals are all of length |AGE(1)| and that the initial age interval is from 0.0 to |AGE(1)|. In this case, all other elements of AGE need not be specified. AGE(N + 1) need not be specified when IMTH = 1.

A — Vector of length N containing the fraction of those dying within each interval who die before the interval midpoint. (Input)
A common choice for all A(I) is 0.5. This choice may also be specified by setting A(1) to any negative value. In this case, the remaining values of A need not be specified.

IPOP — Vector of length N containing the cohort sizes during each interval. (Input)
If IMTH = 0, then IPOP(I) contains the size of the population at the midpoint of interval I. If IMTH = 1, then IPOP(I) contains the size of the cohort at the beginning of interval I. When IMTH = 0, the population sizes in IPOP may need to be adjusted to correspond to the number of deaths in IDTH . See the algorithm section of the document for more information.

IDTH — Vector of length N containing the number of deaths in each age interval. (Input, if IMTH = 0; not used otherwise)
If IMTH = 1, IDTH is not used and may be dimensioned of length 1.

TABLE — N by 12 matrix containing the life table. (Output)
The rows of TABLE correspond to the age intervals.

Col.	Description
1	Lowest age in the age interval.
2	Fraction of those dying within the interval who die before the interval midpoint.
3	Number surviving to the beginning of the interval.
4	Number of deaths in the interval.
5	Death rate in the interval. If IMTH = 1, this column is set to NaN (not a number).
6	Proportion dying in the interval.
7	Standard error of the proportion dying in the interval.
8	Proportion of survivors at the beginning of the interval.
9	Standard error of the proportion of survivors at the beginning of the interval.
10	Expected lifetime at the beginning of the interval.
11	Standard error of the expected life at the beginning of the interval.

12 Total number of time units lived by all of the population in the interval.

Optional Arguments

IPRINT — Printing option. (Input)
If IPRINT = 1, the life table is printed. Otherwise, no printing is done.
Default: IPRINT = 0.

LDTABL — Leading dimension of TABLE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDTABL = size (TABLE,1).

FORTRAN 90 Interface

Generic: `CALL ACTBL (IMTH, N, NPOP, AGE, A, IPOP, IDTH, TABLE [,…])`

Specific: The specific interface names are S_ACTBL and D_ACTBL.

FORTRAN 77 Interface

Single:
```
CALL ACTBL (IMTH, N, NPOP, AGE, A, IPOP, IDTH, IPRINT
TABLE, LDTABL)
```

Double: The double precision name is DACTBL.

Example

The following example is taken from Chiang (1968). The cohort life table has thirteen equally spaced intervals, so AGE(1) is set to –5.0. Similarly, the probabilities of death prior to the middle of the interval are all taken to be 0.5, so A(1) is set to –1.0. Since IPRINT = 1, the life table is printed by ACTBL.

```
!                                  Specifications
      USE ACTBL_INT

      INTEGER    IMTH, IPRINT, LDTABL, N, NPOP
      PARAMETER  (IMTH=1, IPRINT=1, N=13, NPOP=10000, LDTABL=N)
!
      INTEGER    IDTH(13), IPOP(13)
      REAL       A(1), AGE(1), TABLE(13,12)
!
      DATA AGE/-5.0/, A/-1.0/
      DATA IPOP/270, 268, 264, 261, 254, 251, 248, 232, 166, 130, 76, &
          34, 13/
!
      CALL ACTBL (IMTH, N, NPOP, AGE, A, IPOP, IDTH, TABLE, &
                  IPRINT=IPRINT)
!
      END
```

Output

Life Table

Age Class	Age	PDHALF	Alive	Deaths	Death Rate
1	0	0.5	270	2	NaN
2	5	0.5	268	4	NaN
3	10	0.5	264	3	NaN
4	15	0.5	261	7	NaN
5	20	0.5	254	3	NaN
6	25	0.5	251	3	NaN
7	30	0.5	248	16	NaN
8	35	0.5	232	66	NaN
9	40	0.5	166	36	NaN
10	45	0.5	130	54	NaN
11	50	0.5	76	42	NaN
12	55	0.5	34	21	NaN
13	60	0.5	13	13	NaN

Age Class	P(D)	Std(P(D))	P(S)	Std(P(S))	Lifetime
1	0.007	0.00522	1.000	0.00000	43.19
2	0.015	0.00741	0.993	0.00522	38.49
3	0.011	0.00652	0.978	0.00897	34.03
4	0.027	0.01000	0.967	0.01092	29.40
5	0.012	0.00678	0.941	0.01437	25.14
6	0.012	0.00686	0.930	0.01557	20.41
7	0.065	0.01560	0.919	0.01665	15.62
8	0.284	0.02962	0.859	0.02116	11.53
9	0.217	0.03199	0.615	0.02962	10.12
10	0.415	0.04322	0.481	0.03041	7.23
11	0.553	0.05704	0.281	0.02737	5.59
12	0.618	0.08334	0.126	0.02019	4.41
13	1.000	0.00000	0.048	0.01303	2.50

Age Class	Std(Life)	Time Units
1	0.6993	1345.0
2	0.6707	1330.0
3	0.6230	1312.5
4	0.5940	1287.5
5	0.5403	1262.5
6	0.5237	1247.5
7	0.5149	1200.0
8	0.4982	995.0
9	0.4602	740.0
10	0.4328	515.0
11	0.4361	275.0
12	0.4167	117.5
13	0.0000	32.5

Description

Routine ACTBL computes population (current) or cohort life tables based upon the observed population sizes at the middle (IMTH = 0) or the beginning (IMTH = 1) of some userspecified age intervals. The number of deaths in each of these intervals must also be observed.

The probability of dying prior to the middle of the interval, given that death occurs somewhere in the interval, may also be specified. Often, however, this probability is taken to be 0.5. For a discussion of the probability models underlying the life table here, see the references.

Let t_i, for $i = 0, 1, \ldots, t_n$ denote the time grid defining the n age intervals, and note that the length of the age intervals may vary. Following Gross and Clark (1975, page 24), let d_i denote the number of individuals dying in age interval i, where age interval i ends at time t_i. If IMTH = 0, the death rate at the middle of the interval is given by $r_i = d_i/(M_i h_i)$, where M_i is the number of individuals alive at the middle of the interval, and $h_i = t_i - t_{i\text{-}1}$, $t_0 = 0$. The number of individuals alive at the beginning of the interval may be estimated by $P_i = M_i + (1 - a_i)d_i$ where a_i is the probability that an individual dying in the interval dies prior to the interval midpoint. When IMTH = 1, P_i is input directly while the death rate in the interval, r_i, is not needed.

The probability that an individual dies during the age interval from $t_{i\text{-}1}$ to t_i is given by $q_i = d_i/P_i$. It is assumed that all individuals alive at the beginning of the last interval die during the last interval. Thus, $q_n = 1.0$. The asymptotic variance of q_i can be estimated by

$$\sigma_i^2 = q_i(1-q_i)/P_i$$

When IMTH = 0, the number of individuals alive in the middle of the time interval (input in IPOP(I)) must be adjusted to correspond to the number of deaths observed in the interval. Routine ACTBL assumes that the number of deaths observed in interval h_i occur over a time period equal to h_i. If d_i is measured over a period u_i, where $u_i \neq d_i$, then IPOP(I) must be adjusted to correspond to d_i by multiplication by u_i/h_i, i.e., the value M_i input into ACTBL as IPOP(I) is computed as

$$M_i^* = M_i u_i / h_i$$

Let S_i denote the number of survivors at time t_i from a hypothetical (IMTH = 0) or observed (IMTH = 1) population. Then, S_0 = NPOP when IMTH = 0, and S_0 = IPOP(1) for IMTH = 1, and S_i is given by $S_i = S_{i-1} - \delta_{i\text{-}1}$ where $\delta_i = S_i q_i$ is the number of individuals who die in the i-th interval. The proportion of survivors in the interval is given by $V_i = S_i/S_0$ while the asymptotic variance of V_i can be estimated as follows.

$$\text{var}(V_i) = V_i^2 \sum_{j=1}^{i-1} \frac{\sigma_j^2}{(1-q_j)^2}$$

The expected lifetime at the beginning of the interval is calculated as the total lifetime remaining for all survivors alive at the beginning of the interval divided by the number of survivors at the beginning of the interval. If e_i denotes this average expected lifetime, then the variance of e_i can be estimated as (see Chiang 1968)

$$\text{var}(e_i) = \frac{\sum_{j=i}^{n-1} P_j^2 \sigma_j^2 [e_{j+1} + h_{j+1}(1-a_j)]^2}{P_j^2}$$

where $\text{var}(e_n) = 0.0$.

Finally, the total number of time units lived by all survivors in the time interval can be estimated as:

$$U_i = h_i[S_i - \delta_i(1 - a_i)]$$

Chapter 14: Multidimensional Scaling

Routines

Usage Notes

The routines described in this chapter all involve multidimensional scaling. Routine `MSIDV` (page 1131) performs computations for the individual differences metric scaling models. The utility routines are useful for associated computations as well as for programming other methods of multidimensional scaling.

The following is a brief introduction to multidimensional scaling meant to acquaint the user with the purposes of the routines described in this chapter. Also of interest is the table at the end of this section giving the notation used. A more complete description of procedures in multidimensional scaling may be found in the references, as well as in the algorithm sections for the routines.

Multidimensional Scaling Data Types

A "dissimilarity" is a subject's measure of the "distance" between two objects. For example, a subject's estimate of the distance between two cities is a dissimilarity measure that may, or may not, be the actual distance between the cities (depending upon the subjects familiarity with the two cities). Dissimilarities usually have less relationship to distance. For example, the subject may estimate, on a given scale, the difference between two smells, two tastes, two colors, two shapes, etc. As a concrete example, the subject is asked to compare two wines and indicate whether they have very similar tastes (scale value 0), or very different tastes (scale value 10), or are somewhere in between. In this case, no objective measure of "distance" is available, yet the dissimilarity may

be measured. In all cases, however, the larger the difference between the objects, the larger the dissimilarity measure.

If instead the measure increases as the objects become more similar, then a "similarity" measure rather than a "dissimilarity" measure is obtained. Most routines in this chapter require dissimilarities as input so that similarities must be converted to dissimilarities before most routines in this chapter can be used. Routine MSSTN (page 1150) provides two common methods for performing these conversions.

In general, dissimilarities between all objects in a set are measured (yielding a matrix of dissimilarities), and the multidimensional scaling problem is to locate the objects in a Euclidean (or other) space of known dimension given the matrix of dissimilarities. The estimates of object locations should yield predicted distances between the objects that "closely approximate" the observed dissimilarities. In many multidimensional scaling methods, "closely approximates" means that a predefined measure of the discrepancy (the "stress") is minimized. The simplest stress measure is the sum of the squared differences between the observed dissimilarities and the distances predicted by the estimated object locations. This stress measure, as well as all other stress measures used in this chapter, is discussed more fully in the manual document for routine MSTRS (page 1166).

Note that the predicted distances between objects may not be Euclidean distance. Indeed, in one of the more popular multidimensional scaling models, the individual differences model, weighted Euclidean distance is used. Let λ_{1k} and λ_{2k}, $k = 1, \ldots, d$, be the location estimates of two objects (stimuli) in a d dimensional space. Then, the weighted Euclidean distance used in the individual difference model is given by

$$\delta_{12} = \sqrt{\sum_{k=1}^{d} w_k (\lambda_{1k} - \lambda_{2k})^2}$$

Many other distance models are possible. The models used in this chapter are discussed in the manual document for routine MSDST (page 1146).

A dissimilarity is a subject's estimate of the difference ("distance") between two objects. From the observed dissimilarities, a predicted distance between the objects is obtained by estimating the location of the objects in a Euclidean space of given dimension. In metric scaling, the dissimilarity may be a ratio measure (in which case a dissimilarity of zero means that the objects are in the same location) or an interval measure (in which case "distance" plus a constant is observed). When an interval measure is observed, the interval constant, c, must also be estimated in order to relate the dissimilarity to the predicted distance. For ratio measures, c is not required. A couple of methods for estimating c are used by the routines in this chapter. These methods are explained in the routines that use them.

In nonmetric scaling, the dissimilarity is an ordinal (rank) or categorical measure. In this case, the stress function need only assure that the predicted distances satisfy, as closely as possible, the ordinal or categorical relationships observed in the data. Thus, the stress should be zero if the predicted distances maintain the observed rankings in the dissimilarities in ordinal data. The meaning of a stress in categorical data is more obtuse and is discussed further below.

In ordinal data, the stress function is computed as follows: First, the dissimilarities are transformed so that they correspond as closely as possible to the predicted distances, but such that the observed ordinal relationships are maintained. The transformed dissimilarities are called "disparities", and the stress function is computed from the disparities and the predicted distances. (In ratio and

interval data, disparities may be taken as the dissimilarities.) Thus, if the predicted distances preserve the observed ordinal relationships, a stress of zero will be computed. If the predicted distances do not preserve these relationships, then new estimates for the distances based upon the disparities can be computed. These can be followed by new estimates of the disparities. When the new estimates do not lead to a lower stress, convergence of the algorithm is assumed.

In categorical data, all that is observed is a category for the "distance" between the objects, and there are no known relationships between the categories. In categorical data, the disparities are such that the categories are preserved. A score minimizing the stress is found for each category. As with ordinal data, new distances are computed from this score, followed by new scores for the categories, etc., with convergence occurring when the stress cannot be lowered further. In categorical data, a stress of zero should be relatively uncommon.

The individual differences model assumes that the squared distance between stimuli i and j for subject l,

$$\delta_{ijl}^2$$

is given as

$$\delta_{ijl}^2 = \sum_{k=1}^{d} w_{lk}(\lambda_{ik} - \lambda_{jk})^2$$

where d is the number of dimensions (always assumed to be known), λ_{ik} is the location of the i-th stimulus in the k-th dimension, and w_{lk} is the weight given by subject l to the k-th dimension. Let

$$\bar{\delta}_{i\bullet l}^2$$

denote the average of the squared distances in the i-th row of the dissimilarity matrix for the l-th subject, let

$$\bar{\delta}_{\bullet jl}^2$$

be similarly defined for the j-th column, and let

$$\bar{\delta}_{\bullet\bullet l}^2$$

denote the average of all squared distances for the l-th subject. Then, the product moment (double centering) transformation is given by

$$p_{ijl} = -(\delta_{ijl}^2 - \bar{\delta}_{i\bullet l}^2 - \bar{\delta}_{\bullet jl}^2 + \bar{\delta}_{\bullet\bullet l}^2)/2.0$$

The advantage of the product-moment transformations is that the "product-moment" (double centered) matrices $P_l = (p_{ijl})$ can be expressed as

$$P_l = \Lambda[\text{diag}(W_l)]\Lambda^T$$

where $\Lambda = (\lambda_{ik})$ is the configuration matrix, and where $\text{diag}(W_l)$ is a diagonal matrix with the subject weights for subject l, w_{lk}, along the diagonal. If one assumes that the dissimilarities are measured without error, then the dissimilarities can be used in place of the distances, and the above relationship allows one to compute both $\text{diag}(W_l)$ and Λ directly from the product-moment matrices so obtained. If error is present but small, then very good estimates of Λ and $\text{diag}(W_l)$ can

still be obtained (see De Leeuw and Pruzansky 1978). Routine MSDBL (page 1154) computes the product-moment matrices while MSINI (page 1159) computes the above estimates for X and diag(W_l).

Data Structures

The data input to a multidimensional scaling routine is, conceptually, one or more dissimilarity (or similarity) matrices where a dissimilarity matrix contains the dissimilarity measure between the i-th and j-th stimuli (objects) in position (i, j) of the matrix. In multidimensional scaling, the dissimilarity matrix need not be symmetric (asymmetric distances can also be modelled, see routine MSDST, page 1146) but if it is, only elements above the diagonal need to be observed. Moreover, in the multidimensional "unfolding" models, the distances between all pairs of objects are not observed. Rather, all (or at least many) of the dissimilarities between one set of objects and a second set are measured. When these types of input are combined with the fact that missing values are also allowed in many multidimensional scaling routines, it is easy to see that data structures required in multidimensional scaling can be quite complicated. Three types of structures are allowed for the routines described in this chapter. These are discussed below.

Let X denote a matrix containing the input dissimilarities. The columns of X correspond to the different subjects, and a subjects dissimilarity matrix is contained within the column. Thus, X is a matrix containing a set of dissimilarity matrices, one dissimilarity matrix within each column. For any one problem, the form (structure) of all dissimilarity matrices input in X must be consistent over all subjects. The form can vary from problem to problem, however. In the following, X contains only one column and the index for subject is ignored to simplify the notation. The three storage forms used by the routines described in this chapter are

1. **Square symmetric**: For this form, each column of X contains the upper triangular part of the dissimilarity matrix, excluding the diagonal elements (which should be zero anyway). Specifically, X(1) contains the (1, 2) element of the dissimilarity matrix, X(2) contains the (1, 3) element, X(3) contains the (2, 3) element, etc. Let q denote the number of stimuli in the matrix. All $q(q - 1)/2$ off-diagonal elements are stored.

2. **Square asymmetric**: X contains all elements of each square matrix, including the diagonal elements, which are not used. The dissimilarities are stored in X as if X were dimensioned $q \times q$. The diagonal elements are ignored.

3. **Rectangular**: This corresponds to the "unfolding models" in which not all of the dissimilarities in each matrix are observed. In this storage mode, the row stimuli do not correspond to the column stimuli. Because of the form of the data, no diagonal elements are present, and the data are stored in X as if X were dimensioned $r \times s$ where r is the number of row stimuli and s is the number of column stimuli.

Missing values are also allowed. They are indicated in X in either of two ways: 1) The standard IMSL missing value indicator NaN (not a number) may be used to indicate missing values, or 2) negative elements of X are taken to be missing dissimilarities.

Table 14.1 gives some notation commonly used in this chapter. In general, an element of a matrix is denoted by the lowercase matrix name with subscripts. The notation is generally consistent, but there are some variations when variation seems appropriate.

Table 14.1: Commonly Used Notation

Symbol	Fortran	Meaning
δ_{ijl}	DIST	Distance between objects *i* and *j* for subject *l*.
δ^*_{ijl}	DISP	Disparity for objects *i* and *j* for subject *l*.
X	X	The input array of dissimilarities.
D	NDIM	The number of dimensions in the solution.
W	W	The matrix of subject weights.
diag(W_l)		The diagonal matrix of subject weights for subject *l*.
π	WS	The matrix of stimulus weights.
Λ	CFL	The configuration matrix.
α_h	A	The intercept for strata *h*.
β_h	B	The slope for strata *h*.
ν_h	WT	The stratum weight for stratum *h*.
n_h	NCOM	The number nonmissing dissimilarities in stratum *h*.
P_l	P	The product-moment matrix for subject *l*.
ϕ	STRSS	The stress criterion (over all strata).
ϕ_l	STRS	The stress within stratum *l*.
p	POWER	The power to use in the stress criterion.
q	NSTIM	The total number of stimuli.
η	NSUB	The number of matrices input.
Γ		Normalized eigenvectors.
	IFORM	Option giving the form of the dissimilarity input.
	ICNVT	Option giving the method for converting to dissimilarities.
	MODEL	Vector giving the parameters in the distance model.
	ISTRS	Option giving the stress formula to use.
	ITRANS	Option giving the transformation to use.
	IDISP	The method to be used in estimating disparities.
	EPS	Convergence tolerance.

MSIDV

Performs individual-differences multidimensional scaling for metric data using alternating least squares.

Required Arguments

NSTIM — Number of stimuli in each similarity/dissimilarity matrix. (Input)

X — NSUB similarity or dissimilarity matrices in symmetric storage mode. (Input)
Each matrix must occupy consecutive memory positions, and must be stored as a column in X. X must be dimensioned as

```
DIMENSION X(NC2,NSUB)
```

where NC2 = NSTIM * (NSTIM – 1)/2. Each matrix is stored without the diagonal elements by column as upper triangular matrices. For example, a 3 by 3 matrix would be stored with the (1, 2), (1, 3), (2, 3) elements as the first three elements of the first column of X.

NDIM — Number of dimensions desired in the solution. (Input)

DIST — Vector of length NSUB * NC2, where NC2 = NSTIM * (NSTIM – 1)/2, containing the predicted distances. (Output)
DIST contains the distances as predicted by the estimated parameters in the model. DIST has the same storage mode as X and may be treated as a series of NSUB matrices in symmetric storage mode but without the diagonal elements.

CFL — Matrix of size NSTIM by NDIM containing the configuration of points obtained from the multidimensional scaling. (Output)

A — Vector of length NSUB containing the intercepts for each subject. (Output)

B — Vector of length NSUB containing the slopes for each subject. (Output)

WT — Vector of length NSUB containing the criterion function weights for each subject. (Output)

STRS — Vector of length NSUB containing the value of the weighted optimized criterion within each subject. (Output)

STRSS — Value of the weighted optimized criterion function (summed over subjects). (Output)

RESID — NSUB * NC2 vector containing the observation residuals. (Output)
Here, NC2 = NSTIM (NSTIM – 1)/2.

Optional Arguments

NSUB — Number of matrices to be used in the analysis. (Input)
Default: NSUB = size (X,2).

ICNVT — Option for converting from similarity to dissimilarity data. (Input)
If ICNVT = 0, the input data contains dissimilarities and no conversion is performed. If ICNVT = 1, the data are converted from similarity to dissimilarity data by subtracting each similarity from the largest similarity for the subject. If ICNVT = 2, the data are converted to dissimilarities by reciprocating each similarity.
Default: ICNVT = 0.

MODEL — Model option parameter. (Input)
MODEL = 0 means the Euclidean model is used, otherwise, the individual differences model is used.
Default: MODEL = 0.

ISTRS — Option giving the stress formula to be used. (Input)
Default: ISTRS = 0.
Stress formulas differ in the weighting given to each subject. The valid values of ISTRS are:

ISTRS	Weighting used
0	Inverse of within-subject variance of observed dissimilarities about the predicted distances
1	Inverse of within-subject sum of squared dissimilarities
2	Inverse of within-subject variance of dissimilarities about the subject mean

See the Algorithm section for further discussion of the stress formula weights.

ITRANS — Option giving the transformation to be used on the observed and predicted dissimilarities when computing the criterion function. (Input)
Default: ITRANS = 0.

ITRANS	Transformation
0	Squared distances
1	Distances (that is, no transformation is performed)
2	Log of the distances

See the Algorithm section for further discussion of stress formula transformations.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.

1 Printing is performed, but the output is abbreviated.

2 All printing is performed.

LDCFL — Leading dimension of CFL exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCFL = size (CFL,1).

W — NSUB by NDIM matrix containing the subject weights. (Output when MODEL is not zero, not referenced otherwise)
W is not used and may be a 1x1 array if MODEL = 0.

LDW — Leading dimension of W exactly as specified in the dimension statement in the calling program. (Input)
Default: LDW = size (W,1).

FORTRAN 90 Interface

Generic: `CALL MSIDV (NSTIM, X, NDIM, DIST, CFL, A, B, WT, STRS, STRSS, RESID [,…])`

Specific: The specific interface names are S_MSIDV and D_MSIDV.

FORTRAN 77 Interface

Single:
```
CALL MSIDV (NSTIM, NSUB, X, ICNVT, MODEL, ISTRS,
ITRANS,NDIM, IPRINT, DIST, CFL, LDCFL, W, LDW, A, B,
WT, STRS, STRSS, RESID)
```

Double: The double precision name is DMSIDV.

Example 1

The following example concerns some intercity distance rankings. The data are described by Young and Lewyckyj (1979, page 83). The driving mileages between various cities in the United States are ranked, yielding a symmetric ordinal dissimilarity matrix. These rankings are used as input to MSIDV. A Euclidean model is fit. The resulting two-dimensional scaling yields results closely resembling the locations of the major cites in the U.S. Note that MSIDV assumes continuous, not ranked, data.

The original rankings are given as:

$$\begin{pmatrix} - & 4 & 22 & 8 & 34 & 6 & 10 & 35 & 36 & 3 \\ & - & 13 & 15 & 31 & 21 & 9 & 32 & 30 & 5 \\ & & - & 12 & 11 & 29 & 27 & 16 & 19 & 26 \\ & & & - & 24 & 18 & 25 & 28 & 33 & 23 \\ & & & & - & 39 & 42 & 2 & 17 & 37 \\ & & & & & - & 20 & 44 & 45 & 14 \\ & & & & & & - & 43 & 40 & 1 \\ & & & & & & & - & 7 & 41 \\ & & & & & & & & - & 38 \\ & & & & & & & & & - \end{pmatrix}$$

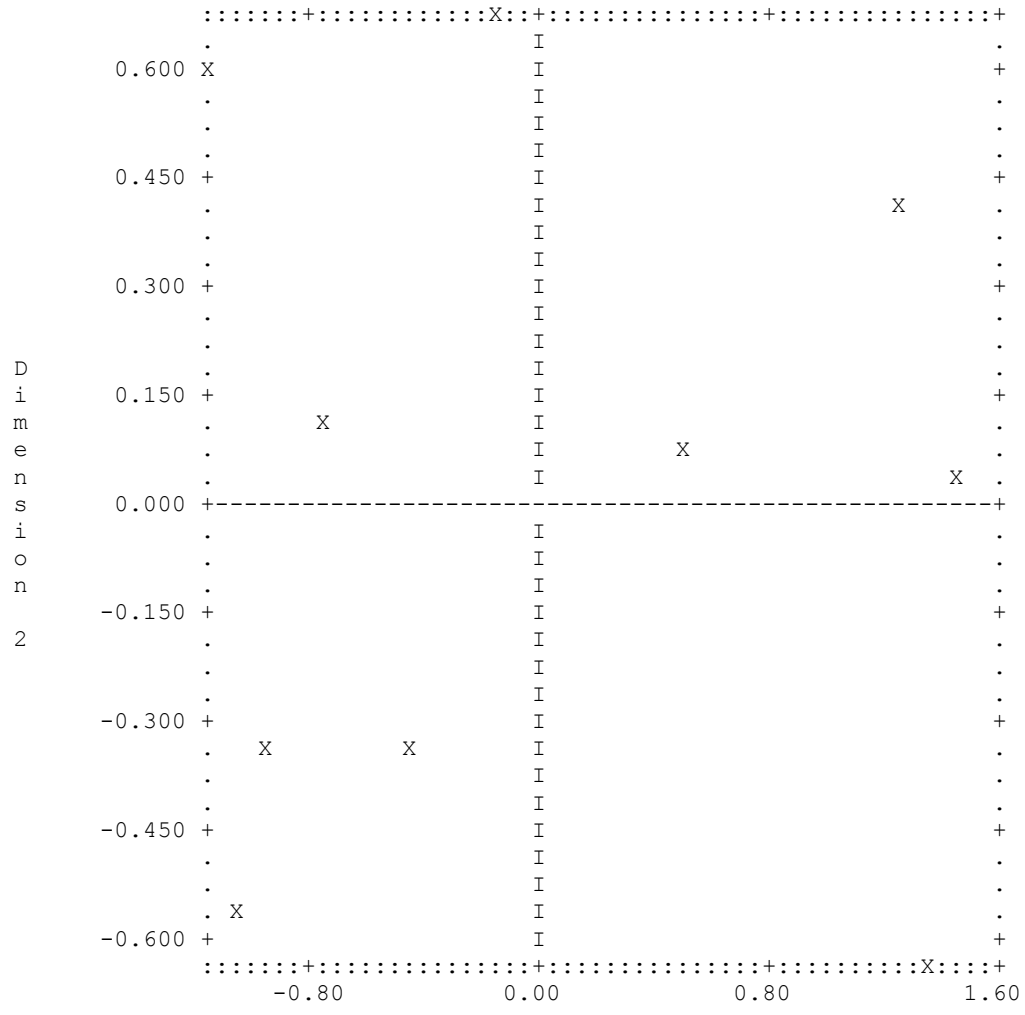
```
      USE PGOPT_INT
      USE MSIDV_INT

      INTEGER    IPRINT, ISTRS, LDCFL, LDW, LNX, NDIM, NSTIM, NSUB, IPAGE
      PARAMETER  (IPRINT=2, ISTRS=1, LDCFL=10, LNX=45, NDIM=2, &
                  NSTIM=10, NSUB=1)
!
      REAL       A(1), B(1), CFL(LDCFL,NDIM), DIST(45), RESID(LNX), &
                 STRS(1), STRSS, WT(1), X(45,1)
!
      DATA X/4, 22, 13, 8, 15, 12, 34, 31, 11, 24, 6, 21, 29, 18, 39, &
          10, 9, 27, 25, 42, 20, 35, 32, 16, 28, 2, 44, 43, 36, 30, &
          19, 33, 17, 45, 40, 7, 3, 5, 26, 23, 37, 14, 1, 41, 38/
!                                Call PGOPT to set page length for
!                                the plotting
      IPAGE =50

      CALL PGOPT (-2, IPAGE)
!
      CALL MSIDV (NSTIM, X, NDIM, DIST, CFL, A, B, WT, STRS, &
                 STRSS, RESID, ISTRS=ISTRS, IPRINT=IPRINT)
!
      END
```

Output

```
 Initial parameter estimates.
        CFL
          1       2
 1  -0.762   0.124
 2  -0.451  -0.349
 3   0.496   0.073
 4  -0.151   0.651
 5   1.237   0.392
 6  -1.114   0.588
 7  -1.077  -0.566
 8   1.461   0.034
 9   1.321  -0.614
10  -0.961  -0.333

                      Iteration history
```

```
 Iter      Source        Stress   Stress change  Maximum Change
  1     INIT   STRSS  0.3755E-02
  1     CONFIG STRSS  0.3399E-02    0.3559E-03      0.8062E-03
  1     LINES  STRSS  0.3142E-02    0.2564E-03      0.8062E-03
  2     CONFIG STRSS  0.3068E-02    0.7382E-04      0.1022E-04
  2     LINES  STRSS  0.3047E-02    0.2156E-04      0.1022E-04
```

Plot(s) of the configuration matrix (CFL)

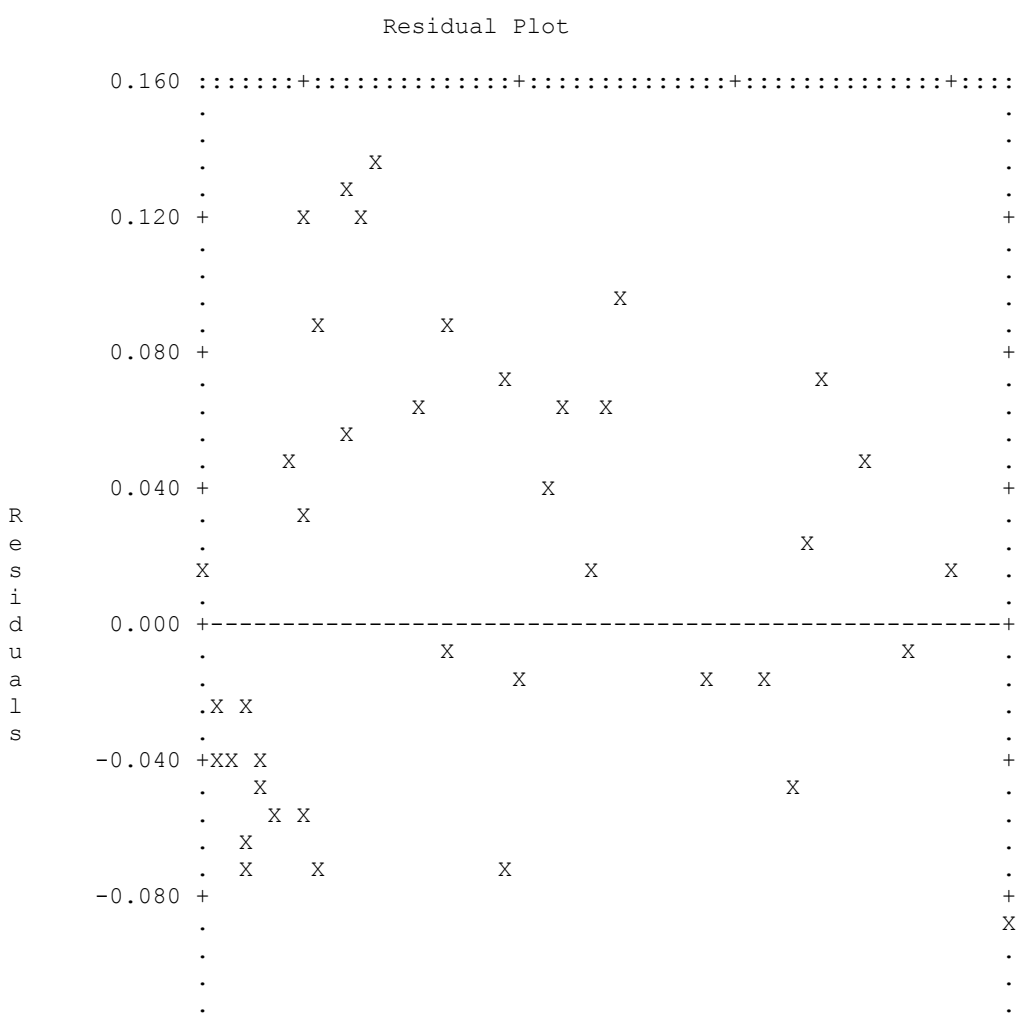


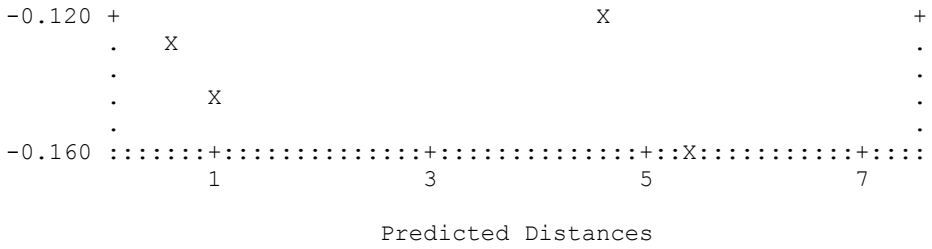
Final parameter estimates.

```
 NCOM
   45

        CFL
         1       2
```

```
  1  -0.738   0.095
  2  -0.447  -0.337
  3   0.497   0.077
  4  -0.153   0.661
  5   1.237   0.399
  6  -1.132   0.609
  7  -1.074  -0.571
  8   1.445   0.035
  9   1.325  -0.624
 10  -0.960  -0.343

     A
 -0.04255

   B
 0.4019

   WT
 0.01248

  STRS
 0.003047

STRESS =     3.04681E-03

                  Residuals
 Subject  Row Stimulus  Column Stimulus Residual
     1           2                1       -0.0436
     1           3                1        0.1230
     1           3                2       -0.1422
     1           4                1       -0.1318
     1           4                2       -.0697
     1           4                3       -0.0581
     1           5                1        0.0950
     1           5                2        0.0631
     1           5                3       -0.0456
     1           5                4        0.0639
     1           6                1       -0.0742
     1           6                2        0.1268
     1           6                3        0.0681
     1           6                4        0.1212
     1           6                5       -0.0495
     1           7                1       -0.0376
     1           7                2       -0.0216
     1           7                3       -0.0736
     1           7                4       -0.0119
     1           7                5        0.0464
     1           7                6        0.0558
     1           8                1       -0.1177
     1           8                2        0.0169
     1           8                3        0.0480
     1           8                4       -0.0173
     1           8                5       -0.0223
     1           8                6        0.0178
     1           8                7       -0.0047
```

1	9	1	-0.0185
1	9	2	0.0373
1	9	3	0.0872
1	9	4	0.0618
1	9	5	0.0335
1	9	6	-0.0913
1	9	7	0.0202
1	9	8	-0.0671
1	10	1	-0.0415
1	10	2	-0.0276
1	10	3	0.0869
1	10	4	0.1342
1	10	5	-0.1565
1	10	6	-0.0522
1	10	7	0.0179
1	10	8	0.0701
1	10	9	-0.0191





Example 2

The second example involves three subjects' assessment of the dissimilarity between rectangles that vary in height and width. An analysis is performed in $k = 2$ dimensions using the individual-differences scaling model. The estimated subject weights, w_{mk}, indicate how each subject weight the dimensions. The raw data are given as follows:

$$\begin{pmatrix} - & 1.00 & 1.41 & 2.24 & 2.00 & 2.24 & 1.41 & 1.00 & 1.00 \\ & - & 1.00 & 2.00 & 2.24 & 2.83 & 2.24 & 2.00 & 1.41 \\ & & - & 1.00 & 1.41 & 2.24 & 2.00 & 2.24 & 1.00 \\ & & & - & 1.00 & 2.00 & 2.24 & 2.83 & 1.41 \\ & & & & - & 1.00 & 1.41 & 2.24 & 1.00 \\ & & & & & - & 1.00 & 2.00 & 1.41 \\ & & & & & & - & 1.00 & 1.00 \\ & & & & & & & - & 1.41 \\ & & & & & & & & - \end{pmatrix}$$

$$\begin{pmatrix} - & 1.50 & 1.68 & 2.12 & 1.50 & 2.12 & 1.68 & 1.50 & 0.75 \\ & - & 0.75 & 1.50 & 2.12 & 3.35 & 3.09 & 3.00 & 1.68 \\ & & - & 0.75 & 1.68 & 3.09 & 3.00 & 3.09 & 1.50 \\ & & & - & 1.50 & 3.00 & 3.09 & 3.35 & 1.68 \\ & & & & - & 1.50 & 1.68 & 2.12 & 0.75 \\ & & & & & - & 0.75 & 1.50 & 1.68 \\ & & & & & & - & 0.75 & 1.50 \\ & & & & & & & - & 1.68 \\ & & & & & & & & - \end{pmatrix}$$

$$\begin{pmatrix} - & 0.50 & 2.06 & 4.03 & 4.00 & 4.03 & 2.06 & 0.50 & 2.00 \\ & - & 2.00 & 4.00 & 4.03 & 4.12 & 2.24 & 1.00 & 2.06 \\ & & - & 2.00 & 2.06 & 2.24 & 1.00 & 2.24 & 0.50 \\ & & & - & 0.50 & 1.00 & 2.24 & 4.12 & 2.06 \\ & & & & - & 0.50 & 2.06 & 4.03 & 2.00 \\ & & & & & - & 2.00 & 4.00 & 2.06 \\ & & & & & & - & 2.00 & 0.50 \\ & & & & & & & - & 2.06 \\ & & & & & & & & - \end{pmatrix}$$

```
      USE MSIDV_INT
      INTEGER    ICNVT, IPRINT, ISTRS, ITRANS, LDCFL, LDW, LNX, MODEL, &
                 NDIM, NSTIM, NSUB
      PARAMETER  (IPRINT=1, LDCFL=9, LDW=3, LNX=108, MODEL=1, NDIM=2, &
                  NSTIM=9, NSUB=3)
!
      REAL       A(NSUB), B(NSUB), CFL(LDCFL,NDIM), DIST(LNX), &
                 RESID(LNX), STRS(NSUB), STRSS, W(LDW,NDIM), WT(NSUB), &
                 X(36,NSUB)
!
      DATA X/1.00, 1.41, 1.00, 2.24, 2.00, 1.00, 2.00, 2.24, 1.41, &
          1.00, 2.24, 2.83, 2.24, 2.00, 1.00, 1.41, 2.24, 2.00, 2.24, &
          1.41, 1.00, 1.00, 2.00, 2.24, 2.83, 2.24, 2.00, 1.00, 1.00, &
          1.41, 1.00, 1.41, 1.00, 1.41, 1.00, 1.41, 1.50, 1.68, 0.75, &
          2.12, 1.50, 0.75, 1.50, 2.12, 1.68, 1.50, 2.12, 3.35, 3.09, &
          3.00, 1.50, 1.68, 3.09, 3.00, 3.09, 1.68, 0.75, 1.50, 3.00, &
          3.09, 3.35, 2.12, 1.50, 0.75, 0.75, 1.68, 1.50, 1.68, 0.75, &
          1.68, 1.50, 1.68, 0.50, 2.06, 2.00, 4.03, 4.00, 2.00, 4.00, &
          4.03, 2.06, 0.50, 4.03, 4.12, 2.24, 1.00, 0.50, 2.06, 2.24, &
          1.00, 2.24, 2.06, 2.00, 0.50, 1.00, 2.24, 4.12, 4.03, 4.00, &
          2.00, 2.00, 2.06, 0.50, 2.06, 2.00, 2.06, 0.50, 2.06/
!
      CALL MSIDV (NSTIM, X, NDIM, DIST, CFL, A, B, WT, STRS, &
                  STRSS, RESID, MODEL=MODEL, IPRINT=IPRINT, W=W)
!
      END
```

Output

```
                  Iteration history
Iter      Source        Stress   Stress change  Maximum Change
  1     INIT   STRSS -0.3590E+03
  1     CONFIG STRSS -0.3590E+03    0.0000E+00      0.5708E-03
  1     SUB WT STRSS -0.3590E+03    0.0000E+00      0.1581E-02
  1     LINES  STRSS -0.3590E+03    0.0000E+00      0.2727E-02
  2     CONFIG STRSS -0.3590E+03    0.0000E+00      0.1442E-06
  2     SUB WT STRSS -0.3590E+03    0.0000E+00      0.7165E-04
  2     LINES  STRSS -0.3590E+03    0.0000E+00      0.7165E-04
Final parameter estimates.

    NCOM
```

```
 1    2    3
36   36   36

CFL
1       2
1   1.225   0.000
2   1.225  -1.225
3   0.000  -1.225
4  -1.225  -1.225
5  -1.225   0.000
6  -1.225   1.225
7   0.000   1.225
8   1.225   1.225
9   0.000   0.000

         W
         1       2
1   1.000   1.000
2   0.342   1.372
3   1.411   0.089

               A
         1           2           3
-0.002773   0.001941   0.000055

            B
     1        2        3
0.2229   0.2587   0.2963

           WT
     1        2        3
1000.0   1000.0   1000.0

          STRS
     1       2       3
-119.7  -119.7  -119.7

STRESS =    -359.018
```

Comments

1. Workspace may be explicitly provided, if desired, by use of M2IDV/DM2IDV. The reference is:

   ```
   CALL M2IDV (NSTIM, NSUB, X, ICNVT, MODEL, ISTRS, ITRANS, NDIM,
   IPRINT, DIST, CFL, LDCFL, W, LDW, A, B, WT, STRS, STRSS, RESID,
   WK1, WK2, WK3, WK4, WK5, WK6, WK7, IWK8, WK10, WK11, WK12, WK13,
   ID, WKDER, DWKHES, DWKGRA, WKDDP, NCOM, DISP)
   ```

   The additional arguments are as follows:

   WK1 — Work vector of length equal to max(NSUB, NDIM * NSTIM, ND + 1)

   WK2 — Work vector of length equal to NDIM * NDIM

WK3 — Work vector of length equal to NSTIM * NSTIM

WK4 — Work vector of length equal to NSTIM * NSTIM

WK5 — Work vector of length equal to NDSS * NDSS

WK6 — Work vector of length equal to 3 * NDSS

WK7 — Work vector of length equal to 5 * NDSS

IWK8 — Integer work vector of length equal to NDSS

WK10 — Work vector of length equal to NDIM * NDIM

WK11 — Work vector of length equal to NSUB * NSUB

WK12 — Work vector of length equal to NDIM * NDIM * max(NSUB, NSTIM)

WK13 — Work vector of length equal to NSTIM * NDIM

ID — Integer work vector of length equal to 4 * NDIM + 2

WKDER — Work vector of length equal to NPAR

DWKHES — Double precision work vector of length equal to NDIM * NDIM * NSTIM * NSTIM

DWKGRA — Double precision work vector of length equal to NPAR

WKDDP — Work vector of length equal to NC2

NCOM — Work vector of length equal to NSUB

DISP — Work vector of length equal to NSUB * NC2

where ND = NDIM * (NDIM + 1)/2, NC2 = NSTIM * (NSTIM – 1)/2, NDSS = max(NDIM, NSTIM, NSUB), and where NPAR = NDIM * NSTIM + 2 * NSUB when MODEL = 0; otherwise NPAR = NDIM * NSTIM + (NDIM + 2) * NSUB.

2. Informational errors

Type	Code	
3	1	At some point during the iterations there were too many step halvings. This is usually not serious.
4	1	The program cannot continue because a Hessian matrix is ill defined. A different model may be required. This error should only occur when there is serious numerical imprecision.
4	2	A dissimilarity matrix has every element missing.

Description

Routine MSIDV performs multidimensional scaling analysis according to an alternating optimization algorithm. Input to MSIDV consists of symmetric dissimilarity matrices measuring distances between the row and column objects. Optionally, similarities can be input, and these can be converted to dissimilarities by use of the ICNVT option. In MSIDV, the row and column objects (stimuli) must be identical. Dissimilarities in multidimensional scaling are used to position the objects within a d = NDIM dimensional space, where d is specified by the user. Optionally, in the individual differences scaling model (MODEL ≠ 0), the weight assigned to each dimension for each subject may be changed.

The Input Data

The data input in X must be in a special symmetric storage form. For this storage mode, the input array X contains only the upper triangular part of each dissimilarity matrix and does not contain the diagonal elements (which should all be zero anyway). Storage of symmetric data in X is as follows: X(1) corresponds to the (1, 2) element in the first matrix (which is a measure of the distance between objects 1 and 2), X(2) corresponds to the (1, 3) element, X(3) corresponds to the (2, 3) element, etc., until all $t = q(q - 1)/2$ off-diagonal elements in the first matrix are stored, where q = NSTIM. The $t + 1$ element in X contains the (1, 2) element in the second matrix, and so on.

Missing values are indicated in either of two ways: 1) The standard missing value indicator NaN (not a number), specified via routine AMACH(6) (Reference Material) may be used to indicate missing values, or 2) Negative elements of X may be used to indicate missing observations. In either case, missing values are estimated as the mean dissimilarity for the subject and used as such when computing initial estimates, and they are omitted from the criterion function when optimal estimates are computed.

Routine MSIDV assumes a metric scaling model. When no transformation is specified (ITRANS = 1), then each datum (after transforming to dissimilarities) is a measure of distance plus a constant, α_m. In this case, the constant (which is always called the "intercept") is assumed to vary with subject and must first be added to the observed dissimilarities in order to obtain a metric. When a transformation is specified (ITRANS ≠ 1), the meaning of α_m changes (with respect to metrics). Thus, when ITRANS = 1, the data is assumed to be interval (see the chapter introduction) while when ITRANS ≠ 1 ratio data is assumed. A scaling factor, the "slope", is also always estimated for each subject.

The Criterion Function

When ISTRS = 1 or 2, the criterion function in MSIDV is given as

$$\phi = \sum_m v_m \sum_{i,j} \left(f\left(\delta^*_{ijm}\right) - \alpha_m - \beta_m f\left(\delta_{ijm}\right) \right)^2$$

where δ_{ijm} denotes the predicted distance between objects i and j on subject m,

$$\delta^*_{ijm}$$

denotes the corresponding dissimilarity (the observed distance), ν_m is the subject weight, f is one of the transformations $f(x) = x^2$, $f(x) = x$, or $f(x) = \ln(x)$ specified by parameter ITRANS, α_m is the intercept added to the transformed observation within each subject, and β_m is the slope for the subject. For ISTRS = 0, the criterion function is given as

$$\phi = \sum_m n_m \ln\left(\sum_{i,j} \left(f\left(\delta^*_{ijm}\right) - \alpha_m - \beta_m f\left(\delta_{ijm}\right)\right)^2 \right)$$

where n_m is the number of nonmissing observations on the m-th subject. Assuming fixed weights, the first derivatives of the criterion for ISTRS = 0 are identical to the first derivatives of the criterion when ISTRS = 1 or 2, but with weights

$$v_m^{-1} = \sum_{i,j} \left(f\left(\delta^*_{ijm}\right) - \alpha_m - \beta_m f\left(\delta_{ijm}\right)\right)^2 / n_m$$

ISTRS can, thus, be thought of as changing the weighting to be used in the criterion function.

The transformation $f(x)$ specified by parameter ITRANS is used to obtain constant within-subject variance of the subject dissimilarities. If the variance of the log of the observed dissimilarities (about the predicted dissimilarities) is constant within subject, then the log transformation should be used. In this case, the variance of a dissimilarity should be proportional to its magnitude. Alternatively, the within-subject variance may be constant when distances (or squared distances) are used.

The Distance Models

The distance models for δ_{ijm} available in MSIDV are given by:

1. The Euclidean model:

$$\delta^2_{ijm} = \sum_{k=1}^{d} \left(\lambda_{ik} - \lambda_{jk} \right)^2$$

2. The individual-differences model:

$$\delta^2_{ijm} = \sum_{k=1}^{d} w_{mk} \left(\lambda_{ik} - \lambda_{jk} \right)^2$$

where Λ denotes the configuration (CFL) so that λ_{ik} is the location of the i-th stimulus in the k-th dimension, where d is the number of dimensions, and where w_{mk} is the weight assigned by the m-th subject to the k-th dimension (W).

The Subject Weights

Weights that are inversely proportional to the estimated variance of the dissimilarities (about their predicted values) within each subject may be preferred because such weights lead to normal distribution theory maximum likelihood estimates (when it is assumed that the dissimilarities are independently normally distributed with constant residual variance). The estimated (conditional) variance used as the inverse of the weight ν_m for the m-th subject in MSIDV (when ISTRS = 0) is computed as

$$v_m^{-1} = \sum_{i,j} \frac{\left(f\left(\delta_{ijm}^*\right) - \alpha_m - \beta_m f\left(\delta_{ijm}\right)\right)^2}{n_m}$$

where the sum is over the observations for the subject, and where n_m is the number of observed nonmissing dissimilarities for the subject. These weights are used in the first derivatives of the criterion function.

When ISTRS = 1, the within-subject average sum of squared dissimilarities are used for the weights. They are computed as

$$v_m^{-1} = \frac{\sum_{i,j} f\left(\delta_{ijm}^*\right)^2}{n_m}$$

Finally, when ISTRS = 2, the within-subject variance of the dissimilarities is used for the weights. These are computed as follows

$$v_m^{-1} = \frac{\sum_{i,j} (f(\delta_{ijm}^*) - \overline{f(\delta_{ijk}^*)})^2}{n_m}$$

where

$$\overline{f(\delta_{ijk}^*)}$$

denotes the average of the transformed dissimilarities in the stratum.

The Optimization Procedure

Initial estimates of all parameters are obtained through methods discussed in routine MSINI (page 1159). After obtaining initial estimates, a modified Gauss-Newton algorithm is used to obtain estimates for the parameters that optimize the criterion function. The parameters are optimized sequentially as follows:

1. Optimize the configuration estimates, Λ = CFL.
2. If required, estimate the optimal subject weights, w_{mk} = W(m, k), one subject at a time.
3. Optimize the parameters α_m = A(m) and β_m = B(m), one subject at a time.
4. If convergence has not been reached, continue at Step 1.

An iteration is defined to be all of the Steps 1, 2, and 3. Convergence is assumed when the maximum absolute change in any parameter during an iteration is less than 10^{-4} or if there is no change in the criterion function during an iteration.

The L_p Gauss-Newton Description

A modified Gauss-Newton algorithm is used in the estimation of all parameters. This algorithm, which is discussed in detail by Merle and Spath (1974), uses iteratively reweighted least squares

on a Taylor series linearization of the parameters in δ_{ijm}. During each iteration, the subject weights, which may depend upon the parameters in the model, are assumed to be fixed.

Standardization

All models available are overparameterized so the resulting parameter estimates are not uniquely defined. For example, in the Euclidean model, the columns of X can be translated or "rotated" (multiplied by an orthonormal matrix), and the resulting stress will not be changed. To eliminate lack of uniqueness due to translation, model estimates for the configuration are centered in all models. No attempt at eliminating the rotation problem is made, but note that rotation invariance is not a problem in many of the models given. With more general models than the Euclidean model, other kinds of overparameterization occur. Further restrictions on the parameters to eliminate this overparameterization are given below by the model transformation type specified by `ITRANS`. In the following, $w_{lk} \in W$, where W is the matrix of subject weights. The restrictions to be applied by model transformation type are

1. For all models:

(*a*) $\sum_{i=1}^{q} x_{ik} = 0$

where q = NSTIM. i.e., center the columns of X.

(*b*) If W is in the model, scale the columns of W so that

$$\sum_{i=1}^{q} x_{ik}^2 = 1$$

2. For $f(x) = x$ and $f(x) = x^2$:

(*a*) Set $b_h = 1$ if the data are matrix conditional and W is in the model or if the data are unconditional. (Matrix conditional with one matrix is considered to be unconditional data.)

(*b*) If W is not in the model, scale all elements in X so that

$$\sum_{h=1}^{\eta} b_h^2 = \eta$$

where η = NSUB is the number of matrices observed.

3. For $f(x) = \ln(x)$, substitute a_h for b_h (but set a_h to 0 instead of 1) in all restrictions in Item 2.

MSDST

Computes distances in a multidimensional scaling model.

Required Arguments

CFL — `NSTIM` by `NDIM` matrix containing the stimulus configuration. (Input)

NSUB — Number of subjects. (Input)

DIST — Vector of length nv * NSUB, where nv = NSTIM * (NSTIM – 1)/2 if IFORM = 0, and nv = NSTIM * NSTIM otherwise. (Output)
DIST may be treated as NSUB distance matrices. Storage in DIST is such that the elements of each column of a subject's distance matrix are adjacent. Each column in the matrix is immediately followed by the elements in the next column. If IFORM = 0, then only the elements in each column above the diagonal are stored. Otherwise, all elements are stored.

Optional Arguments

NSTIM — Number of stimuli. (Input)
Default: NSTIM = size (CFL,1).

NDIM — Number of dimensions in the model. (Input)
Default: NDIM = size (CFL,2).

LDCFL — Leading dimension of CFL exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCFL = size (CFL,1).

IMOD — Vector of length 3 describing the weighting to be used. (Input)
Default: IMOD = 0.

I	WEIGHT
1	Not used. Reserved for other scaling subroutines.
2	Subject weights (in W).
3	Stimulus weights (in WS).

If IMOD(*i*) is zero, then the *i*-th set of weights is not used. Otherwise, the weights are used. For the Euclidean model, set IMOD(2) = IMOD(3) = 0. For the individual differences model, IMOD(2) should not be zero. For the stimulus weighted individual differences model, both IMOD(2) and IMOD(3) are not zero.

IFORM — Form option. (Input)
If IFORM = 0, the computed distances are stored as the upper triangle of square matrices stored columnwise without the diagonal elements. Otherwise, the distances are stored as square matrices and include the diagonal elements. See argument DIST.
Default: IFORM = 0.

ITRANS — Transformation option. (Input)
ITRANS determines the output returned in DIST.
Default: ITRANS = 0.

ITRANS	Output in DIST

0	Squared distances
1	Distances
2	Log of the distances

W — NSUB by NDIM matrix of individual weights. (Input)
If IMOD(2) is zero, then W is not referenced and can be a 1x1 array.
Default: W is a 1x1 array and not used.

LDW — Leading dimension of W exactly as specified in the dimension statement in the calling program. (Input)
Default: LDW =size(W,1).

WS — NSTIM by NDIM matrix of stimulus weights. (Input)
If IMOD(3) is zero, then W is not referenced and can be a 1x1array.
Default: WS is a 1x1 array and not used.

LDWS — Leading dimension of WS exactly as specified in the dimension statement in the calling program. (Input)
Default: LDWS =size(WS,1)

FORTRAN 90 Interface

Generic: `CALL MSDST (CFL, NSUB, DIST [,…])`

Specific: The specific interface names are S_MSDST and D_MSDST.

FORTRAN 77 Interface

Single:
```
CALL MSDST (NSTIM, NDIM, CFL, LDCFL, NSUB, IMOD, IFORM,
ITRANS, W, LDW, WS, LDWS, DIST)
```

Double: The double precision name is DMSDST.

Example

The following small example illustrates the distance computations in symmetric matrices. The data are fictional.

```
      USE MSDST_INT
      USE UMACH_INT
      INTEGER    IFORM, ITRANS, LDCFL, LDW, LDWS, NDIM, NSTIM, NSUB
      PARAMETER  (LDCFL=4, LDW=2, LDWS=4, NDIM=2, NSTIM=4, NSUB=2)
!
      INTEGER    IMOD(3), NOUT
      REAL       CFL(NSTIM,NDIM), DIST(12), W(NSUB,NDIM), WS(1,1)
!
      DATA IMOD/0, 1, 0/
```

```
!
      DATA CFL/1.0, -1.0, 1.0, -1.0, &
              1.0, 1.0, -1.0, -1.0/
!
      DATA W/1.0, 2.0, 1.0, 2.0/
!
      CALL MSDST (CFL, NSUB, DIST, IMOD=IMOD, W=W)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) DIST
      END
```

Output

```
4.00000    4.00000    8.00000    8.00000    4.00000    4.00000    8.00000
8.00000    16.0000    16.0000    8.00000    8.00000
```

Description

Routine MSDST computes squared distances, distances, or log distances for various metrics in multidimensional scaling. The "distances" are computed and stored as either square matrices or as upper triangular symmetric matrices stored columnwise without the diagonal. In both cases, the distances are output in a vector of the required length. The terminology and metrics used here are the same as those used in the ALSCAL program of Takane, Young, De Leeuw (1977).

Suppose that there are q stimuli, M subjects, and d dimensions. Let λ_{ik} denote the location of the i-th stimulus in the k-th dimension. If w_{ik} denotes the weight of the i-th subject on the k-th dimension (matrix W) and pi_{ik} denotes the weight for the i-th stimulus on the k-th dimension (matrix WS), then the distance models computed are the same as the distance models in MSIDV. They are given by:

Euclidean Model

$$\delta_{ijm}^2 = \sum_{k=1}^{d} \left(\lambda_{ik} - \lambda_{jk} \right)^2$$

Individual Differences Model

$$\delta_{ijm}^2 = \sum_{k=1}^{d} w_{mk} \left(\lambda_{ik} - \lambda_{jk} \right)^2$$

Stimulus-Weighted Model

$$\delta_{ijm}^2 = \sum_{k=1}^{d} \pi_{ik} \left(\lambda_{ik} - \lambda_{jk} \right)^2$$

Stimulus-Weighted Individual Differences Model

$$\delta_{ijm}^2 = \sum_{k=1}^{d} \pi_{ik} w_{mk} \left(\lambda_{ik} - \lambda_{jk} \right)^2$$

where δ_{ijm} is the distance between the i-th and j-th stimuli on the m-th subject.

MSSTN

Transforms dissimilarity/similarity matrices and replace missing values by estimates to obtain standardized dissimilarity matrices.

Required Arguments

NROW — Number of row stimuli in each dissimilarity/similarity matrix. (Input)

NCOL — Number of column stimuli in each dissimilarity/similarity matrix. (Input)
If IFORM = 0 or 1, NCOL must equal NROW, and the stimuli in the rows and columns must correspond to one another.

IFORM — Storage option indicating the storage mode for the input data in each column of X. (Input)
Array X contains NSUB columns, and each column of X contains a dissimilarity/similarity matrix stored as specified by option IFORM.

IFORM	Data Storage Mode
0	Symmetric storage mode without the diagonal elements. (Upper triangular matrix stored columnwise.) In this storage mode, consecutive elements of each column of X contain the (1, 2), (1, 3), (2, 3), (1, 4), (2, 4), (3, 4), …, (NROW − 1, NROW) elements of the corresponding dissimilarity/similarity matrix.
1	Square matrix in full storage mode. Consecutive elements of each column of X contain the (1, 1), (2, 1), (3, 1), …, (NROW, 1), (1, 2), (2, 2), …, (NROW, NROW) elements of the corresponding dissimilarity/similarity matrix.
2	Rectangular matrix in full storage mode. In this storage mode, the row and column stimuli input in X do not correspond to each other. Let *m* = NROW. Consecutive elements of each column of X contain the (1, *m* + 1), (2, *m* + 1), …, (NROW, *m* + 1), (1, *m* + 2), …, (NROW, *m* + 2), …, (NROW, *m* + NCOL) elements of the corresponding dissimilarity/similarity matrix.

X — NSUB similarity or dissimilarity matrices in storage mode as determined by IFORM. (Input)
X must be dimensioned as:

```
DIMENSION X(LDX,NSUB)
```

where LDX ≥ NROW * NCOL in full storage mode and LDX ≥ NROW * (NROW − 1)/2 in symmetric storage mode. See argument IFORM for the method of storage used for each storage mode. Negative elements of X, or elements equal to NaN (“not a number”) are

presumed to be missing values and will be estimated as an appropriate average in MSSTN.

ICNVT — Option for converting from similarity to dissimilarity matrices. (Input)

ICNVT	Conversion
0	No conversion performed.
1	Subtracting each similarity from the largest similarity in the strata (see ISTRAT).
2	Take the reciprocal of each similarity (elements of X equal to zero are assumed to be missing).

ISTRAT — Option giving the level of stratification to be used. (Input)
If ISTRAT = 1, each dissimilarity/similarity matrix in X is considered to be in a different stratum. The data are said to be matrix conditional. If ISTRAT = 2, each column of each dissimilarity matrix is considered to be in a different stratum. (Thus, each column of array X contains NCOL strata.) For ISTRAT to be 2, IFORM must be 1 or 2. The data are said to be column conditional. If ISTRAT = 3, all of the dissimilarity/similarity matrices in X are considered to be in the same stratum. The data are said to be unconditional.

NCOM — Vector containing the number of nonmissing observations in each stratum. (Output)
The diagonal elements of each dissimilarity/similarity matrix are not counted.

ISTRAT	Length of NCOM
1	NSUB
2	NSUB * NSTIM, where NSTIM = NROW when IFORM = 0 or 1, and NSTIM = NROW + NCOL when IFORM = 2
3	1

XOUT — Vector of length NV * NSUB containing the standardized dissimilarity matrices where NV = NROW * (NROW – 1)/2 if IFORM = 0 and NV = NSTIM * NSTIM otherwise. (Output)
The value of NSTIM is as described in parameter NCOM. XOUT contains the standardized dissimilarity matrices in the same storage mode as X if IFORM = 0 or 1 and stored as square matrices when IFORM = 2. Missing values are replaced by an appropriate average dissimilarity and changed in sign. Scaling is performed as requested.

Optional Arguments

NSUB — Number of dissimilarity/similarity matrices. (Input)
Default: NSUB = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

ISCALE — Scaling option. (Input)
Default: ISCALE = 1.

ISCALE	Scaling
0	No scaling is performed.
1	The data in each stratum are scaled such that the sum of the squared dissimilarities equals the number of elements in the stratum.

FORTRAN 90 Interface

Generic: `CALL MSSTN (NROW, NCOL, IFORM, X, ICNVT, ISTRAT, NCOM, XOUT [,…])`

Specific: The specific interface names are S_MSSTN and D_MSSTN.

FORTRAN 77 Interface

Single: `CALL MSSTN (NROW, NCOL, NSUB, IFORM, X, LDX, ICNVT, ISTRAT, ISCALE, NCOM, XOUT)`

Double: The double precision name is DMSSTN.

Example

The following example illustrates the use of MSSTN on similarity data that are converted to dissimilarity data with the ICNVT = 1 option. Standardization within each matrix is used. The input data is such that IFORM = 0. Since ICNVT = 1 and all elements of the input data are nonnegative, no missing values are estimated. The input data is given by the following two similarity matrices:

$$\begin{pmatrix} - & 4 & 0 & 3 & 1 \\ & - & 1 & 1 & 3 \\ & & - & 0 & 2 \\ & & & - & 4 \\ & & & & - \end{pmatrix} \quad \begin{pmatrix} - & 1 & 2 & 3 & 1 \\ & - & 1 & 2 & 0 \\ & & - & 1 & 3 \\ & & & - & 4 \\ & & & & - \end{pmatrix}$$

```
      USE MSSTN_INT
      USE WRIRN_INT
      USE UMACH_INT
      INTEGER    ICNVT, IFORM, ISTRAT, LDX, NCOL, NROW, NSUB
      PARAMETER  (ICNVT=1, IFORM=0, ISTRAT=1, LDX=10, NCOL=5, &
                  NROW=5, NSUB=2)
!
      INTEGER    I, J, K, N, NCOM(NSUB), NOUT
      REAL       X(LDX,NSUB), XOUT(NROW*(NROW-1))
!
      DATA X/4.0, 0.0, 1.0, 3.0, 1.0, 0.0, 1.0, 3.0, 2.0, 4.0, 1.0, &
          2.0, 1.0, 3.0, 2.0, 1.0, 1.0, 0.0, 3.0, 4.0/
!
      CALL MSSTN (NROW, NCOL, IFORM, X, ICNVT, ISTRAT, &
                  NCOM, XOUT)
!
      CALL WRIRN ('NCOM', NCOM, 1, NSUB, 1)
      CALL UMACH (2, NOUT)
!
      N = 1
      DO 20  I=1, 2
         WRITE (NOUT,99998) I
         DO 10  J=1, 4
            WRITE (NOUT,99999) (XOUT(K),K=N,N+J-1)
            N = N + J
   10    CONTINUE
   20 CONTINUE
!
99998 FORMAT (///' Output matrix (in XOUT)', I2)
99999 FORMAT (1X, 4F8.3)
!
      END
```

Output

```
NCOM
 1    2
10   10

Output matrix (in XOUT) 1
 0.000
 1.569   1.177
 0.392   1.177   1.569
 1.177   0.392   0.784   0.000

Output matrix (in XOUT) 2
 1.205
 0.803   1.205
 0.402   0.803   1.205
 1.205   1.606   0.402   0.000
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `M2STN/DM2STN`. The reference is:

```
CALL M2STN (NROW, NCOL, IFORM, NSUB, X, LDX, ICNVT, ISTRAT,
ISCALE, NCOM, XOUT, NSTIM, XX, XMIS)
```

The additional arguments are as follows:

NSTIM — Integer scalar. NSTIM = NROW when IFORM = 0 or 1, and NSTIM = NROW + NCOL when IFORM = 2.

XX — Work vector of length NSTIM * NSTIM.

XMIS — Work vector of length NSTIM * NSTIM.

2. Informational errors

Type	Code	
3	1	At least one column in column conditional data has all elements missing.
4	2	A dissimilarity matrix has every element missing.

Description

Routine MSSTN standardizes dissimilarity/similarity data to be usable by other routines in the multidimensional scaling chapter. Routine MSSTN converts similarity to dissimilarity data, estimates missing values within specified strata ("conditionality groups"), scales the data, computes the number of nonmissing data elements within each stratum, and stores the data in a standard form.

The computations proceed as follows:

1. Routine MSSTN begins by expanding rectangular or symmetric storage-form data into square storage mode (the form when IFORM = 1).

2. Missing values are replaced by the average nonmissing value within the stratum, or when there is only one stratum, the average within each matrix is used. If all elements in a stratum are missing and the stratum is a column of the dissimilarity/similarity matrix, then the average of the nonmissing elements in the matrix is used as the missing value estimate. (Missing values are estimated primarily for use in routines computing estimates via "double-centering", routines MSINI, page 1159, and MSDBL, page 1154.) Missing values are denoted in the output by changing the signs of the estimated missing elements to be negative.

3. The data are converted to dissimilarity data from similarity data according to the method specified by the parameter ICNVT.

4. The data are scaled according to the method specified by the ISCALE parameter.

MSDBL

Obtains normalized product-moment (double centered) matrices from dissimilarity matrices.

Required Arguments

NSTIM — Number of stimuli in each dissimilarity matrix. (Input)

IFORM — Storage option for the data in each dissimilarity matrix. (Input) Each column of X contains one of the NSUB dissimilarity matrices in the storage mode specified by IFORM.

IFORM	Data Storage Mode
0	Symmetric storage mode without the diagonal elements. (Upper triangular matrix stored columnwise.) In this storage mode, consecutive elements of each column of X contain the (1, 2), (1, 3), (2, 3), (1, 4), (2, 4), (3, 4), …, (NSTIM − 1, NSTIM) elements of the corresponding dissimilarity matrix.
1	Square matrix in full storage mode. Consecutive elements of each column of X contain the (1, 1), (2, 1), (3, 1), …, (NROW, 1), (1, 2), (2, 2), …, (NSTIM, NSTIM) elements of the corresponding dissimilarity matrix.

X — NV by NSUB matrix containing the NSUB dissimilarity matrices, where NV = NSTIM * (NSTIM − 1)/2 if IFORM = 0, and NV = NSTIM * NSTIM if IFORM = 1. (Input)
Missing values (NaN, "not a number") are not allowed in X, but the position of a missing element may be indicated as a negative dissimilarity. Since MSDBL uses the absolute value of each element in X in the estimation procedure, the signs of elements in X have no effect. See Comments.

DISP — NSTIM by NSTIM by NSUB array containing the NSUB dissimilarity matrices in full storage mode. (Output)
In DISP, missing value estimates are positive, and all elements represent the square of distances.

P — NSTIM by NSTIM by NSUB array containing the standardized product-moment matrices in full storage mode. (Output)
P contains NSUB matrices, each of size NSTIM by NSTIM. If DISP is not needed, DISP and P can occupy the same storage locations.

DS — NSTIM by NSTIM array containing the sum of the NSUB matrices in P. (Output)

Optional Arguments

NSUB — Number of dissimilarity matrices. (Input)
Default: NSUB = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

ISCALE — Scaling option. (Input)
Default: ISCALE = 1.

ISCALE	Type of Scaling
0	No scaling
1	Scaling within each matrix
2	Scaling over all matrices

Scaling is such that the Euclidean norm of the vector of scaled data is equal to the number of elements in vector.

LDDISP — Leading and second dimension of DISP exactly as specified in the dimension statement in the calling program. (Input)
Default: LDDISP = size (DISP,1).

LDP — Leading and second dimension of P exactly as specified in the dimension statement in the calling program. (Input)
Default: LDP = size (P,1).

LDDS — Leading dimension of DS exactly as specified in the dimension statement in the calling program. (Input)
Default: LDDS = size (DS,1).

FORTRAN 90 Interface

Generic: `CALL MSDBL (NSTIM, IFORM, X, DISP, P, DS [,…])`

Specific: The specific interface names are S_MSDBL and D_MSDBL.

FORTRAN 77 Interface

Single:
```
CALL MSDBL (NSTIM, NSUB, IFORM, X, LDX, ISCALE, DISP,
LDDISP, P, LDP, DS, LDDS)
```

Double: The double precision name is DMSDBL.

Example

The following example illustrates the use of MSDBL in computing product-moment matrices for two input dissimilarity matrices. The input matrices are given as:

$$\begin{pmatrix} - & 4 & 1 & 3 & 1 \\ & - & 1 & 1 & 3 \\ & & - & 2 & 2 \\ & & & - & 4 \\ & & & & - \end{pmatrix} \begin{pmatrix} - & 1 & 2 & 3 & 1 \\ & - & 1 & 2 & 2 \\ & & - & 1 & 3 \\ & & & - & 4 \\ & & & & - \end{pmatrix}$$

```
      USE MSDBL_INT
      USE WRRRN_INT
      INTEGER    IFORM, LDDISP, LDDS, LDP, LDX, NSTIM, NSUB
      PARAMETER  (IFORM=0, LDDISP=5, LDDS=5, LDP=5, LDX=10, &
                  NSTIM=5, NSUB=2)
!
      REAL       DISP(LDDISP,LDDISP,NSUB), DS(LDDS,NSTIM), &
                 P(LDP,LDP,NSUB), X(LDX,NSUB)
!
      DATA X/4.0, 1.0, 1.0, 3.0, 1.0, 2.0, 1.0, 3.0, 2.0, 4.0, 1.0, &
          2.0, 1.0, 3.0, 2.0, 1.0, 1.0, 2.0, 3.0, 4.0/
!
      CALL MSDBL (NSTIM, IFORM, X, DISP, P, DS)
!
      CALL WRRRN ('The first matrix in DISP', DISP(1:,1:,1))
      CALL WRRRN ('The second matrix in DISP', DISP(1:,1:,2))
      CALL WRRRN ('The first matrix in P', P(1:,1:,1))
      CALL WRRRN ('The second matrix in P', P(1:,1:,2))
      CALL WRRRN ('DS', DS)
!
      END
```

Output

```
        The first matrix in DISP
          1       2       3       4       5
1    0.00   16.00    1.00    9.00    1.00
2   16.00    0.00    1.00    1.00    9.00
3    1.00    1.00    0.00    4.00    4.00
4    9.00    1.00    4.00    0.00   16.00
5    1.00    9.00    4.00   16.00    0.00
        The second matrix in DISP
          1       2       3       4       5
1    0.00    1.00    4.00    9.00    1.00
2    1.00    0.00    1.00    4.00    4.00
3    4.00    1.00    0.00    1.00    9.00
4    9.00    4.00    1.00    0.00   16.00
5    1.00    4.00    9.00   16.00    0.00
        The first matrix in P
          1       2       3       4       5
1   1.110  -1.931   0.274  -0.487   1.034
2  -1.931   1.110   0.274   1.034  -0.487
3   0.274   0.274  -0.182  -0.182  -0.182
4  -0.487   1.034  -0.182   1.338  -1.703
5   1.034  -0.487  -0.182  -1.703   1.338
        The second matrix in P
          1       2       3       4       5
```

```
1    0.500    0.000  -0.500  -1.000   1.000
2    0.000    0.000   0.000   0.000   0.000
3   -0.500    0.000   0.500   1.000  -1.000
4   -1.000    0.000   1.000   2.000  -2.000
5    1.000    0.000  -1.000  -2.000   2.000
                    DS
         1        2       3       4       5
1    0.805   -0.966  -0.113  -0.743   1.017
2   -0.966    0.555   0.137   0.517  -0.243
3   -0.113    0.137   0.159   0.409  -0.591
4   -0.743    0.517   0.409   1.669  -1.852
5    1.017   -0.243  -0.591  -1.852   1.669
```

Comments

Routine MSSTN (page 1150) may be used to obtain the matrix X with missing values estimated and changed in sign so that all estimates of missing values are negative. Routine MSSTN will also convert similarities to dissimilarities. Unless a ratio distance measure is observed, the user will usually call MSSTN prior to calling MSDBL.

Description

Routine MSDBL computes product-moment (double-centered) matrices from input dissimilarity matrices. The product-moment matrices output from MSDBL may be scaled either within each matrix, over all matrices input, or not at all.

The interest in product-moment matrices can be explained as follows: Let Λ denote a configuration of points in an d-dimensional Euclidean space with center at the origin. When the data is measured without error, the matrix $P = \Lambda\Lambda^T$ can also be written as the "double-centered" matrix (defined below) obtained from the matrix of squares of distances between the rows of

$$\Lambda\left(\delta_{ij}^2 = \sum\nolimits_k \left(\lambda_{ik} - \lambda_{jk}\right)^2\right)$$

These distances are input, approximately, in the dissimilarities. Thus, an estimate for Λ can be obtained, approximately, by computing the double-centered matrix P from the squared dissimilarities and then computing Λ from the scaled eigenvectors of P (such that $P = \Lambda\Lambda^T$).

The computation in MSDBL proceeds as follows:

1. Each input dissimilarity matrix is transformed into a square symmetric matrix of distances. Asymmetric matrices are made symmetric by averaging the matrix of dissimilarities with its transpose.

2. Estimates for the square of the distances,

   $$\overline{\tilde{\delta}^2}$$

   are computed as the square of the estimated distances.

3. Let

$$\overline{\tilde{\delta}^2}_{m\bullet\bullet}$$

denote the average squared distance in a matrix m of squared distances, let

$$\overline{\tilde{\delta}^2}_{mi\bullet}$$

denote the average of the i-th row of estimated squared distances in matrix m and let

$$\overline{\tilde{\delta}^2}_{m\bullet j}$$

denote the average of the j-th column. The m-th product-moment matrix is computed from the m-th estimated squared distance matrix as

$$p_{mij} = -\left(\overline{\tilde{\delta}^2}_{mij} - \overline{\tilde{\delta}^2}_{mi\bullet} - \overline{\tilde{\delta}^2}_{m\bullet j} + \overline{\tilde{\delta}^2}_{m\bullet\bullet}\right)/2$$

The resulting matrix is said to be double-centered.

4. If the elements of P_m are to be scaled within matrix m, then the elements of P_m are divided by

$$\sqrt{\sum_{i,j} p_{mij}^2 / q^2}$$

where q = NSTIM so that q^2 is the total number of elements in the matrix. If the elements of P are to be scaled over all matrices, then the elements of each matrix are divided by

$$\sqrt{\sum_{m,i,j} p_{mij}^2 / \left(sq^2\right)}$$

where s = NSUB.

5. The matrix DS is computed as the sum over all subjects of the product-moment matrices, P_m.

MSINI

Computes initial estimates in multidimensional scaling models.

Required Arguments

NSTIM — Number of stimuli in each dissimilarity matrix. (Input)

IFORM — Storage option for the data in each dissimilarity matrix. (Input)
Each column of X contains one of the NSUB dissimilarity matrices in the storage mode specified by IFORM.

IFORM	Data Storage Mode
0	Symmetric storage mode without the diagonal elements. (Upper triangular matrix stored columnwise.) Consecutive elements of each column of X

contain the (1, 2), (1, 3), (2, 3), (1, 4), (2, 4), (3, 4), …, (NSTIM – 1, NSTIM) elements of the dissimilarity matrix.

1 Square matrix in full storage mode. Consecutive elements of each column of X contain the (1, 1), (2, 1), (3, 1), …, (NSTIM, 1), (1, 2), (2, 2), …, (NSTIM, NSTIM) elements of the dissimilarity matrix.

X — NV by NSUB matrix containing the NSUB dissimilarity matrices, where NV = NSTIM * (NSTIM – 1)/2 if IFORM = 0, and NV = NSTIM * NSTIM if IFORM = 1. (Input)
If IFORM = 0, then the input data is assumed to be symmetric, and the elements below and on the diagonal are not input. If IFORM = 1, all elements of each column of X are input, and the data for the column need not form a symmetric matrix. Missing values (NaN, "not a number") are not allowed in X, but the position of a missing element may be indicated as a negative dissimilarity. Since MSINI uses the absolute value of each element in X as the dissimilarity to be used in the estimation procedure, the sign of an element in X has no effect.
See Comment 3.

IMOD — Vector of length 3 giving the model parameters to be estimated. (Input)
IMOD also gives the method of initialization to be used for each set of parameters. Each element of IMOD corresponds to a different parameter matrix. The correspondence is given as:

Element	Parameter Matrix
1	CFL–The configuration
2	W–The subject weights
3	WS–The stimulus weights

The value used for each element of IMOD tells how the parameter matrix is to be initialized.

Value	Effect on Parameter Matrix
0	The parameter matrix is not used.
1	The parameter matrix is input and its values are fixed. The parameter matrix may be standardized.
2	Initial estimates are input, but they may be changed by MSINI.
3	MSINI calculates the initial estimates.

IMOD(1) must be nonzero. IFORM must not be 0 if IMOD(3) is not zero. If IMOD(2) or IMOD(3) is 1, IMOD(1) must be 1. If IMOD(3) is 1, IMOD(2) must not be 2 or 3.

CFL — NSTIM by NDIM matrix containing the estimated stimulus coordinates. (Input/Output, if IMOD(1) = 1 or 2; output, otherwise)

W — NSUB by NDIM matrix of subject weights. (Input/Output, if IMOD(2) = 1 or 2, output, if IMOD(2) = 3, not referenced if IMOD(2) = 0)
W is not referenced and can be dimensioned as a 1 by 1 matrix if IMOD(2) = 0.

WS — NSTIM by NDIM matrix of stimulus weights. (Input/Output, if IMOD(3) = 1 or 2; output, if IMOD(3) = 3, not referenced if IMOD(3) = 0)
WS is not referenced and can be dimensioned as a 1 by 1 matrix if IMOD(3) = 0.

WMIN — Minimum weight in W prior to adjustment. (Output, if IMOD(2) = 2 or 3; not referenced if IMOD(2) = 0 or 1)
If WMIN is negative, the weights in W are adjusted such that all weights are positive by subtracting WMIN from each element in W.

WSMIN — Minimum weight in WS prior to adjustment. (Output, if IMOD(3) = 2 or 3; not referenced if IMOD(3) = 0 or 1)
If WSMIN is negative, the weights in WS are adjusted such that all weights are positive by subtracting WSMIN from each element in WS.

Optional Arguments

NSUB — Number of dissimilarity matrices to be used in the analysis. (Input)
Default: NSUB = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

NDIM — Number of dimensions in the solution. (Input)
Default: NDIM = size (CFL,2).

LDCFL — Leading dimension of CFL exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCFL = size (CFL,1).

LDW — Leading dimension of W exactly as specified in the dimension statement in the calling program. (Input)
Default: LDW = size (W,1).

LDWS — Leading dimension of WS exactly as specified in the dimension statement in the calling program. (Input)
Default: LDWS = size (WS,1).

FORTRAN 90 Interface

Generic: `CALL MSINI (NSTIM, IFORM, X, IMOD, CFL, W, WS, WMIN, WSMIN [,…])`

Specific: The specific interface names are `S_MSINI` and `D_MSINI`.

FORTRAN 77 Interface

Single: `CALL MSINI (NSTIM, NSUB, IFORM, X, LDX, IMOD, NDIM, CFL, LDCFL, W, LDW, WS, LDWS, WMIN, WSMIN)`

Double: The double precision name is `DMSINI`.

Example

The following example illustrates the use of `MSINI` to obtain initial estimates for an individual differences model when symmetric dissimilarities matrices obtained from two subjects are input. The input matrices are given as:

$$\begin{pmatrix} - & 4 & 1 & 3 & 1 \\ & - & 1 & 1 & 3 \\ & & - & 2 & 2 \\ & & & - & 4 \\ & & & & - \end{pmatrix} \quad \begin{pmatrix} - & 1 & 2 & 3 & 1 \\ & - & 1 & 2 & 2 \\ & & - & 1 & 3 \\ & & & - & 4 \\ & & & & - \end{pmatrix}$$

Estimates obtained from `MSINI` are not optimal. Usually an optimizing multidimensional scaling routine will be called with the initial estimates computed in `MSINI`.

```
      USE UMACH_INT
      USE MSINI_INT
      USE WRRRN_INT
      INTEGER    IFORM, LDCFL, LDW, LDWS, LDX, NDIM, NSTIM, NSUB
      PARAMETER  (IFORM=0, LDCFL=5, LDW=2, LDWS=5, LDX=10, NDIM=2, &
                 NSTIM=5, NSUB=2)
!
      INTEGER    IMOD(3), NOUT
      REAL       CFL(LDCFL,NDIM), W(LDW,NDIM), WMIN, WS(LDWS,NDIM), &
                 WSMIN, X(LDX,NSUB)
!
      DATA X/4.0, 1.0, 1.0, 3.0, 1.0, 2.0, 1.0, 3.0, 2.0, 4.0, 1.0, &
          2.0, 1.0, 3.0, 2.0, 1.0, 1.0, 2.0, 3.0, 4.0/
      DATA IMOD/3, 3, 0/
!
      CALL UMACH (2, NOUT)
!
      CALL MSINI (NSTIM, IFORM, X, IMOD, CFL, W, WS, WMIN, WSMIN)
!
      CALL WRRRN ('The Configuration', CFL)
      CALL WRRRN ('Subject weights', W)
      WRITE (NOUT,99999) WMIN
```

```
!
99999 FORMAT (/, ' WMIN = ', F12.4)
!
      END
```

Output

```
The Configuration
         1        2
1   0.2279   0.6854
2  -0.0808  -0.6584
3  -0.1728  -0.0090
4  -0.6621  -0.2287
5   0.6879   0.2107
Subject weights
        1       2
1   7.078   8.533
2   9.615   0.000
WMIN =       0.0000
```

Comments

1. Workspace may be explicitly provided, if desired, by use of M2INI/DM2INI. The reference is:

   ```
   CALL M2INI (NSTIM, NSUB, IFORM, X, LDX, IMOD, NDIM, CFL,
   LDCFL, W, LDW, WS, LDWS, WMIN, WSMIN, TR, XX, DISP, DS, EWK1,
   EWK2, IEWK, C)
   ```

   The additional arguments are as follows:

   TR — Real work vector of length max(NDIM + 1, NSUB, NSTIM).

   XX — Real work vector of length NSTIM * NSTIM.

   DISP — Real work vector of length NSTIM * NSTIM * NSUB.

   DS — Real work vector of length NSTIM * NSTIM.

   EWK1 — Real work vector of length 3 * NSTIM.

   EWK2 — Real work vector of length max(5 * NSTIM, 4 * NSUB).

   IEWK — Integer work vector of length NSTIM.

   C — Real work vector of length NDIM * NDIM * NSUB.

2. Informational error

Type	Code	
4	1	The sum of the product moment matrices for the data input in X has less than NDIM positive eigenvalues. Rerun with NDIM = number of

positive eigenvalues or less or provide initial estimates for the configuration matrix CFL.

3. Routine MSSTN (page 1150) may be used to obtain the matrix X with missing values estimated and changed in sign so that all estimates of missing values are negative. Routine MSSTN will also convert similarities to dissimilarities. Unless a ratio distance measure is observed, the user will usually call MSSTN prior to calling MSINI.

Description

Routine MSINI computes initial estimates for the stimulus configuration (Λ = CFL), subject weights (W = W), and stimulus weights (Π = WS) in multidimensional scaling models. The number of dimensions in the solution must also be input. Routine MSINI requires complete (i.e., no missing values) dissimilarity matrices as input. Consequently, missing data must be replaced by an estimate (often an average of other dissimilarities). Because the absolute values of dissimilarities are used, missing dissimilarities may be denoted by changing their sign to be negative. Estimation of missing values, and further standardization, can be performed through the use of routine MSSTN (page 1150).

In some cases, MSINI can use values input in parameter matrices CFL, W, or WS in order to compute initial estimates for other parameter matrices. For example, values input in matrix CFL may be used in the estimation of initial estimates for W or WS. Because of the method of estimation, values input for some parameter matrices will not effect the estimate computed for other matrices. In particular, values input in W will not effect the estimation of CFL, and values input in WS will not effect the estimation of either CFL or W. Note that some combinations of input and estimated matrices are not even allowed (see the option parameter IMOD). Also, note that when the configuration matrix CFL is input and fixed (except for standardization), computed estimates for all weights W and WS are arbitrarily taken as 1.

Let

$$\tilde{\delta}_{ijl}^2$$

denote the squared distance between stimulus i and stimulus j for matrix (subject) l, let

$$\bar{\delta}_{i\bullet l}^2$$

denote the average of the squared distances in the i-th row for the l-th subject, let

$$\bar{\delta}_{\bullet jl}^2$$

be similarly defined, and let

$$\bar{\delta}_{\bullet\bullet l}^2$$

denote the average of all squared distances for the l-th subject. If each dissimilarity input in X is measured without error, then the dissimilarities and the distances are identical. In MSINI, the errors observed in the dissimilarities,

$$\tilde{\delta}_{ijl}^2$$

are assumed to be small so that good estimates for the squared distances may be computed by squaring each dissimilarity (after first subtracting the constant obtained in Step 1 below). The computations proceed as follows:

1. The squared distance matrices are double-centered using the product moment transformation

$$p_{ijl} = -\left(\tilde{\delta}^2_{ijl} - \overline{\tilde{\delta}^2}_{i\bullet l} - \overline{\tilde{\delta}^2}_{\bullet jl} + \overline{\tilde{\delta}^2}_{\bullet\bullet l}\right)/2$$

   The matrix formed by averaging the product moment matrices P_l (over subjects) is computed as $\overline{P}$

2. If the configuration has been input and cannot be modified (i.e., if `IMOD`(1) is 1), then all weights to be estimated are taken as 1, and the computations continue in Step 8 below.

3. If the configuration matrix has not been input, then a preliminary estimate is obtained by first computing the eigenvectors (Γ) corresponding to the *d*-largest eigenvalues of $\overline{P}$.

   The configuration is then estimated as $\Gamma\Delta^{1/2,}$ where Δ is the square matrix containing the eigenvalues along the diagonal and zeros off the diagonal.

4. If the subject weights `W` are to be estimated, or if they can be modified (i.e., if `IMOD`(2) is 2 or greater), then a SUMSCALE procedure (De Leeuw and Pruzansky, 1978) is used to estimate the weights (regardless of the values input) and to "rotate" the configuration estimates. This is done as follows:

   A. The matrices

$$C_l = \Phi^{-1}\Lambda^T P_l \Lambda \Phi^{-1}$$

   are computed, where $\Phi = \Delta$ if Δ has been computed, and where the diagonal elements of Φ are the diagonal elements of $\Lambda^T\Lambda$ otherwise (the off-diagonal elements of Φ are always zero).

   B. An orthogonal matrix Q is found such that the sum of the squared off-diagonal elements of $Q^T C_l Q$ is minimized over all matrices C. (See IMSL routine `KPRIN`,Chapter 9, Covariance Structures and Factor Analysis.)

   C. A new configuration estimate is obtained by "rotating" the current estimate, i.e., $\Lambda_n = \Lambda$.

   D. The subject weights for subject l are taken as the diagonal elements of $Q^T C_l Q$.

5. If the subject weights have been computed and the minimum weight in `W` is negative, add its absolute value to all elements in `W` to ensure that all estimated stimulus weights are nonnegative.

6. If the stimulus weights are to be estimated (i.e., if (IMOD(3) is 2 or 3), then least-squares estimates are used. The least-squares model is obtained by substituting predicted distance for actual distance in the multidimensional scaling model specified by IMOD (see the chapter introduction for a discussion of the models available). Least-squares fitting is then performed over the NSUB subjects.

7. If the stimulus weights have been computed and the minimum weight in WS is negative, its absolute value is added to all elements in WS to ensure that all estimated stimulus weights are nonnegative.

8. The estimates are standardized (even when IMOD(i) = 2) as follows:

   A. If IMOD(2) is not zero, then let

   $$r_i = \lambda_i^T \lambda_i$$

   where i-th is the ithcolumn of the configuration matrix. Let w_i denote the i-th column of the subject weight matrix. Standardize Λ such that the diagonal elements of $\Lambda^T\Lambda$ are 1. Multiply w_i by r_i.

   B. If IMOD(2) = 0 but IMOD(3) is not zero, then compute r_i and standardize the configuration matrix as above. Multiply the i-th column of WS by r_i.

   C. If both IMOD(2) and IMOD(3) are nonzero, then compute

   $$s_i = \sqrt{w_i^T w_i}$$

   and standardize W such that $W^T W$ is an identity matrix. Multiply the i-th column of WS by c_i.

MSTRS

Computes various stress criteria in multidimensional scaling.

Required Arguments

DIST — Vector of length N containing the distances. (Input)
Missing values are not allowed in DIST.

DISP — Vector of length N containing the disparities. (Input)

A — The intercept. (Input)
If INTCEP = 0, A is not used.

B — The slope. (Input)
If ISLOPE = 0, B is not used.

POWER — Power to use in the stress function. (Input)
POWER must be greater than or equal to 1.

STRSS — The computed stress criterion. (Output)

WT — The weight used in computing the stress. (Output)
If the weight is too large, a maximum weight is used. See the algorithm section of the manual document.

Optional Arguments

N — Number of distances and disparities. (Input)
Default: N = size (DIST,1).

INTCEP — Intercept option parameter. (Input)
If INTCEP = 0, the intercept is not used in the model. If INTCEP = 1, the intercept is used in the model.
Default: INTCEP = 1.

ISLOPE — Slope option parameter. (Input)
If ISLOPE = 0, the slope B is not used. If ISLOPE = 1, the slope is used.
Default: ISLOPE = 1.

ISTRS — Stress option parameter. (Input)
Default: ISTRS = 1.

ISTRS	Stress Criterion Used
0	Log stress
1	Stress weighted by the inverse of the sum of the squared disparities
2	Stress weighted by the inverse of the sum of the centered squared disparities

FORTRAN 90 Interface

Generic: `CALL MSTRS (DIST, DISP, A, B, POWER, STRSS, WT [,…])`

Specific: The specific interface names are S_MSTRS and D_MSTRS.

FORTRAN 77 Interface

Single:
```
CALL MSTRS (N, DIST, DISP, INTCEP, A, ISLOPE, B, POWER,
ISTRS, STRSS, WT)
```

Double: The double precision name is DMSTRS.

Example

The following example illustrates the computation of stress when the log of the distances and disparities are input. For this example, ISTRS is 1 and POWER is 2.

```
      USE MSTRS_INT
      USE UMACH_INT
      USE SDOT_INT
      INTEGER    INTCEP, ISLOPE, ISTRS, N
      REAL       A, POWER
      PARAMETER  (A=0.0, INTCEP=0, N=10, POWER=2.0)
!
      INTEGER    I, NOUT
      REAL       ALOG, B, DISP(N), DIST(N), STRSS, WT
      INTRINSIC  ALOG
!
      DATA DIST/4.0, 1.5, 1.25, 3.0, 1.75, 2.0, 1.0, 3.5, 2.5, 3.75/
      DATA DISP/4.0, 1.0, 1.0, 3.0, 1.0, 2.0, 1.0, 3.0, 2.0, 4.0/
!                                Transform the data
      DO 10  I=1, N
         DIST(I) = ALOG(DIST(I))
         DISP(I) = ALOG(DISP(I))
   10 CONTINUE
!                                Compute a slope
      B = SDOT(N,DISP,1,DIST,1)/SDOT(N,DIST,1,DIST,1)
!                                Compute the stress
      CALL MSTRS (DIST, DISP, A, B, POWER, STRSS, WT, INTCEP=INTCEP)
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STRSS, WT
!
99999 FORMAT (' STRSS = ', F12.4, '    WT = ', F12.4)
      END
```

Output

```
STRSS =       0.0720    WT =       0.1385
```

Description

Routine MSTRS computes the value of stress criteria commonly used in multidimensional scaling. Routine MSTRS allows transformed values of the disparities and distances to be input and will compute the stress on the transformed values. Additionally, the user can input a slope and/or an intercept to be used in the stress computations, and the stress can be computed using an arbitrary L_p norm as well as the squared error norm in which $p = 2$.

Let

$$\delta_i^*$$

denote a disparity, δ_i denote the corresponding distance, α denote the intercept, and let β denote the slope. If INTCEP = 0, then set $\alpha = 0$. If ISLOPE = 0, then set $\beta = 1$.

Set $\varepsilon = 0.001$, and let

$$\tau = \sum_{i=1}^{n} \left| \delta_i^* - \alpha - \beta \delta_i \right|^p$$

When ISTRS = 0, the stress is computed as

$$\phi_0 = n \ln \left[\max \left(n\varepsilon, \tau \right) \right]$$

where n is the number of nonmissing disparities, and p = POWER is the power to be used. This stress formula, when optimized, can lead to to normal distribution theory maximum likelihood estimation. It can not be used in nonmetric scaling. The weight is computed as $n/\max(n\varepsilon, \tau)$.

When ISTRS is 1, the stress is computed as

$$\phi_1 = \frac{\tau}{\max \left(\varepsilon\tau, \sum_{i=1}^{n} | \delta_i^* |^p \right)}$$

and the weight returned is given as

$$1 / \max \left(\varepsilon\tau, \sum_{i=1}^{n} \left| \delta_i^* \right|^p \right)$$

Takane, Young, and de Leeuw (1977) recommend using this formula when the data is not column conditional (i.e., whenever the stress is computed over one or more dissimilarity matrices rather than over one column in a single matrix). When ISTRS = 2, the stress is given by

$$\phi_2 = \frac{\tau}{\max \left(\varepsilon\tau, \sum_{i=1}^{n} | \delta_i^* - \delta_\bullet^* |^p \right)}$$

where

$$\bar{\delta}_\bullet^* = \sum_{i=1}^{n} \delta_i^*$$

is the average of the nonmissing disparities. The weight is computed as

$$1 / \max \left(\varepsilon\tau, \sum_{i=1}^{n} \left| \delta_i^* - \bar{\delta}_\bullet^* \right|^p \right)$$

Takane, Young, and de Leeuw (1977) recommend this stress for column conditional data.

Missing values (NaN) are not allowed in DIST while missing disparities in DISP are not used in the computations. If all disparities are missing, the stress criteria is set to 0, and the weight (WT) is set to missing (NaN).

In general, a single call to MSTRS would be made for each strata ("conditionality group") in the data.

Chapter 15: Density and Hazard Estimation

Routines

Usage Notes

The routines described in this chapter compute estimates for smoothing parameters and estimates in models for estimating density and hazard functions. For density estimation, the penalized likelihood method of Scott (1976) may be used to obtain smooth estimates for arbitrary (smooth) densities. Alternatively, the routines DESKN (page 1176) and DNFFT (page 1180) obtain density estimates by the kernel method for a given window width and kernel function. Routine DNFFT uses a Gaussian kernel, while for routine DESKN, the kernel is provided by the user. Finally, routine DESPT (page 1185) finds linear or quasi-cubic interpolated estimates of a density. Tapia and Thompson (1978) discuss all of these methods.

For hazard estimation, the methods of Tanner and Wong (1984) are used to obtain estimates of the smoothing parameters in a modified likelihood. These methods are implemented in routines HAZRD (page 1188) and HAZEZ (page 1195), the difference between the routines is in the ease of use and the options offered. For given smoothing parameters, the routine HAZST (page 1203) may be used to obtain estimates for the hazard function.

DESPL

Performs nonparametric probability density function estimation by the penalized likelihood method.

Required Arguments

X — Vector of length NOBS containing the random sample of responses. (Input)

NODE — Number of mesh nodes for the discrete probability density estimate. (Input)
NODE must be an odd integer greater than 4.

BNDS — Vector of length 2 containing the upper and lower endpoints for the interval of support of the density. (Input)
The node values are taken as BNDS(1), BNDS(1) + h, …, BNDS(2), where h = (BNDS(2) − BNDS(1))/(NODE − 1). All observations in vector X should be in the support interval.

DENS — Vector of length NODE containing the estimated values of the discrete pdf at the NODE equally spaced mesh nodes. (Input/Output, if INIT ≠ 0; output, otherwise)
If INIT is not zero, then DENS(1) through DENS(NODE) contain the (positive) initial estimates on input. The sum of these estimates times the window width h (see BNDS) must equal 1.0, i.e., the integral of the density must be 1.

STAT — Vector of length 4 containing output statistics. (Output)
STAT(1) and STAT(2) contain the log-likelihood and the log-penalty terms, respectively. STAT(3) and STAT(4) contain the estimated mean and variance for the estimated density.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

INIT — Initialization option. (Input)
INIT = 0 means that a bootstrap procedure is used to obtain initial estimates for the density. Otherwise, user-supplied initial estimates are contained in DENS on entry into DESPL.
Default: INIT = 0.

ALPHA — Penalty-weighting factor that controls the smoothness of the estimate. (Input)
For standard normal data, ALPHA = 10.0 works well. Other values that might be tried are 1.0 and 100.0. ALPHA must be greater than 0.0.
Default: ALPHA = 10.0.

MAXIT — Maximum number of iterations allowed in the iterative procedure. (Input)
MAXIT = 30 is typical.
Default: MAXIT = 30.

EPS — Convergence criterion. (Input)
When the Euclidean norm of the changes to DENS is less than EPS, convergence is assumed. EPS = 0.0001 is typical.
Default: EPS = 0.0001.

NMISS — Number of missing values in X. (Output)

FORTRAN 90 Interface

Generic: `CALL DESPL (X, NODE, BNDS, DENS, STAT [,…])`

Specific: The specific interface names are S_DESPL and D_DESPL.

FORTRAN 77 Interface

Single:
```
CALL DESPL (NOBS, X, NODE, BNDS, INIT, ALPHA, MAXIT, EPS,
DENS, STAT, NMISS)
```

Double: The double precision name is DDESPL.

Example

An estimate for a density function of unknown form using a random sample of size 10 and 13 mesh points with $\alpha = 10$ is estimated as follows:

```
      USE DESPL_INT
      USE UMACH_INT
      INTEGER    NOBS, NODE
      PARAMETER  (NOBS=10, NODE=13)
!
      INTEGER    NOUT
      REAL       BNDS(2), DENS(NODE), STAT(4), X(NOBS)
!
      DATA BNDS/-3., 3./
      DATA X/-.9471, -.7065, -.2933, -.1169, .2217, .4425, .4919, &
          .5752, 1.1439, 1.3589/
!
      CALL DESPL (X, NODE, BNDS, DENS, STAT)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,'('' DENS = '',9F7.4, /, 9X, 4F7.4)') DENS
      WRITE (NOUT,'('' Log-likelihood term = '', F7.3, /, &
      &             '' Log-penalty term    = '', F7.3, /, &
      &             '' Mean                = '', F7.3, /, &
      &             '' Variance            = '', F7.3)') STAT
      END
```

Output

```
DENS =  0.0000 0.0014 0.0356 0.1111 0.2132 0.3040 0.3575 0.3565 0.2947
        0.1986 0.0986 0.0288 0.0000
Log-likelihood term = -11.968
Log-penalty term    =  -1.303
Mean                =   0.217
Variance            =   1.042
```

The following figure shows the affect of various choices of α. For larger α, the density estimate is smoother.

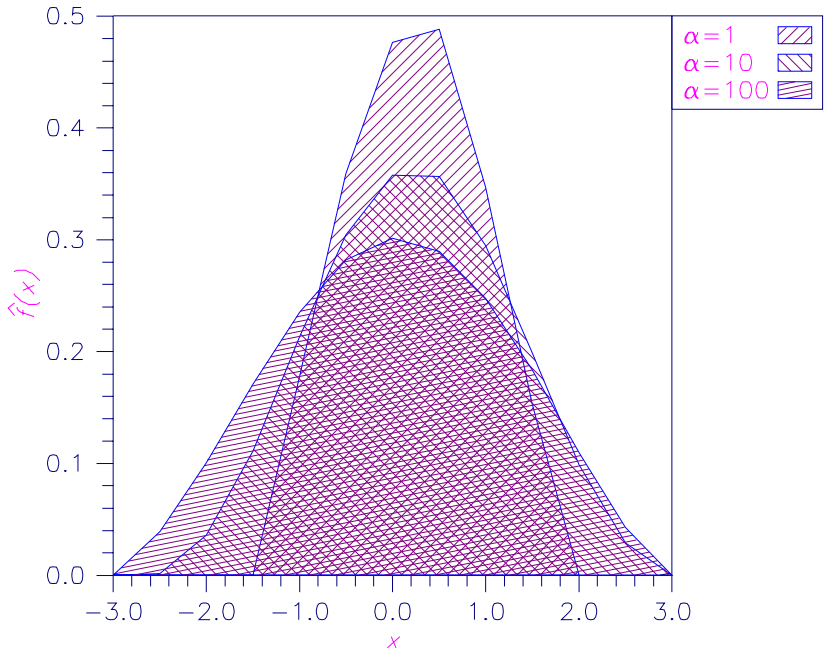


Figure 15-1 Density Estimates Using α = 1, 10, 100

Comments

1. Workspace may be explicitly provided, if desired, by use of D2SPL/DD2SPL. The reference is:

   ```
   CALL D2SPL (NOBS, X, NODE, BNDS, INIT, ALPHA, MAXIT,
   EPS, DENS, STAT, NMISS, HESS, LDHESS, ILOHI, DENEST, B, IPVT,
   WK2, XWK)
   ```

   The additional arguments are as follows:

   HESS — Work vector of length 7 * (NODE − 2).

LDHESS — Leading dimension of HESS exactly as specified in the dimension statement in the calling program. (Input)
The leading dimension must be set to 7.

ILOHI — Integer work vector of length 2 * NODE.

DENEST — Work vector of length 3 * NODE.

B — Work vector of length NODE.

IPVT — Integer work vector of length NODE − 2.

WK2 — Work vector of length NODE − 2.

XWK — Work vector of length NOBS. If X is sorted with all missing (NaN, not a number) values at the end, then XWK is not needed. If X is not needed, X and XWK can share the same storage location.

2. Informational error

Type	Code	
3	1	The maximum number of iterations is exceeded.

3 Routine DESPT (page 1185) may be used after the estimates DENS have been obtained in order to obtain an interpolated estimate of the density at new points. Use AMESH = BNDS in calling DESPT.

Description

Routine DESPL computes piecewise linear estimates of a one-dimensional density function for a given random sample of observations. These estimates are discussed in detail in Scott et al. (1980), and in Tapia and Thompson (1978, Chapter 5). The estimator of the density function is piecewise linear over the finite interval (BNDS(1) to BNDS(2)), is nonnegative, and integrates to one. A penalty method is used to ensure "smooth" behavior of the estimator. The criterion function to be maximized is a discrete approximation to

$$\Phi = \prod_{i=1}^{n} f(x_i) \exp\left(-\alpha \int \left| \frac{d^2 f(t)}{dt^2} \right|^2 dt \right)$$

where n = NOBS and $f(t)$ is a density function. Let m = NODE. The discrete approximation is as follows: The density f is estimated at each of the equally spaced grid points t_j, for $j = 1, \ldots, m$, with restriction $f(t_1) = f(t_m) = 0.0$; the density at each data point x_i is then estimated using linear interpolation. The integral of the second derivative of the square of f is approximated using the piecewise linear function defined by the estimates of f at the grid points t_j.

Because ln Φ is actually maximized, the criterion can be separated into a likelihood term (returned in STAT(1)) and a penalty term (returned in STAT(2)).

The parameter α (= ALPHA) determines the amount of "smoothness" in the estimate. The larger the value of α, the smoother the resulting estimator for *f*. In practice, the user should pick α as small as possible such that there is not excessive bumpiness in the estimator. One way of doing this is to try several values of α that differ by factors of 10. The resulting estimators can then be graphically displayed and examined for bumpiness. α could then be chosen from the displayed density estimates. IMSL routines can be used to produce line printer plots (PLOTP, page 1232) of the estimated density. For a random sample from the standard normal distribution, $\alpha = 10.0$ works well. Note that α changes with scale. If *x* is multiplied by a factor β, α should be multiplied by a factor β^5.

The second choice to be made in using DESPL is the mesh for the estimator. The mesh interval (BNDS(1), BNDS(2)) should be picked as narrow as possible since a narrow mesh will speed algorithm convergence. Note, however, that points outside the interval (BNDS(1), BNDS(2)) are not included in the likelihood. Because of this fact, DESPL actually estimates a density that is conditional on the mesh interval (BNDS(1), BNDS(2)). The number of mesh nodes, NODE, should be as small as possible, but large enough to exhibit the "fine" structure of the density. One possible method for determining NODE is to use NODE = 21 initially. With NODE = 21, find an acceptable value for α. When an acceptable value for α has been found, increase or decrease NODE as required.

STAT(3) and STAT(4) contain "exact" estimates of the mean and variance when the estimated piecewise linear density is used in the required integrals. Routine DESPT (page 1185) may be used to find interpolated estimates for the density at any point *x* given the NODE estimates of the density returned in DENS.

DESKN

Performs nonparametric probability density function estimation by the kernel method.

Required Arguments

XKER — User-supplied FUNCTION to compute the kernel at any point on the real line. The form is XKER(Y), where:

Y — Point at which the kernel is to be evaluated.

XKER — Value of the kernel at point Y.

X — Vector of length NOBS containing the random sample of observations. (Input)

WINDOW — Window width for the kernel function. (Input)
Generally, several different values of WINDOW should be tried.

XMAX — Cutoff value such that XKER(Y) = 0.0 for all |Y| greater than XMAX. (Input)
If XMAX exists, then the kernel function is 0.0 for all Y greater in absolute value than XMAX, and the efficiency of the computations is enhanced. If no such XMAX exists or the user does not wish to make use of XMAX, then XMAX should be assigned any nonpositive value.

XPT — Vector of length NXPT containing the values at which a density estimate is desired. (Input)
If XMAX is greater than zero, then XPT must be sorted from smallest to largest.

DENS — Vector of length NXPT containing the density estimates at the points specified in XPT. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NXPT — Number of points at which a density estimate is desired. (Input)
Default: NXPT = size (XPT,1).

NMISS — Number of missing (NaN, not a number) values in X. (Output)

FORTRAN 90 Interface

Generic: `CALL DESKN (XKER, X, WINDOW, XMAX, XPT, DENS [,…])`

Specific: The specific interface names are S_DESKN and D_DESKN.

FORTRAN 77 Interface

Single: `CALL DESKN (XKER, NOBS, X, WINDOW, XMAX, NXPT, XPT, DENS, NMISS)`

Double: The double precision name is DDESKN.

Example

In this example, the standard normal density function is estimated at 13 points using a random sample of 10 points from a standard normal distribution. The biweight kernel function is used. The actual density for the standard normal density is also reported in the output for comparison. The random sample is generated using routines RNSET and RNNOR in Chapter 18, Random Number Generation.

```
      USE RNSET_INT
      USE RNNOR_INT
      USE DESKN_INT
      USE UMACH_INT
      INTEGER    NOBS, NXPT
      REAL       C1, WINDOW, XMAX
      PARAMETER  (C1=0.3989423, NOBS=10, NXPT=13, WINDOW=2.0, XMAX=1.0)
!
      INTEGER    I, NMISS, NOUT
      REAL       DENS(NXPT), EXP, X(NOBS), XKER, XPT(NXPT)
      INTRINSIC  EXP
      EXTERNAL   XKER
```

```
!
      DATA XPT/-3.0, -2.5, -2.0, -1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5, &
          2.0, 2.5, 3.0/
!
      CALL RNSET (1234457)

      CALL RNNOR (X)
      CALL DESKN (XKER, X, WINDOW, XMAX, XPT, DENS, NMISS=NMISS)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,'('' NMISS = '', I1)') NMISS
      WRITE (NOUT,'('' DENS Estimate = '', 10F6.4,/,8X,3F6.4)') DENS
      WRITE (NOUT,'('' DENS Exact    = '',10F6.4,/,8X,3F6.4)') &
                            (C1*EXP(-XPT(I)*XPT(I)/2.0),I=1,NXPT)
      END
      REAL FUNCTION XKER (Y)
      REAL       Y
!
      REAL       ABS
      INTRINSIC  ABS
!
      IF (ABS(Y) .LT. 1.0) THEN
         XKER = 15.0*(1.0-Y*Y)*(1.0-Y*Y)/16.0
      ELSE
         XKER = 0.0
      END IF
      RETURN
      END
```

Output

```
NMISS = 0
DENS Estimate = 0.00000.01180.07900.16980.26780.34670.36870.31840.22340.1391
        0.06120.01350.0005
DENS Exact    = 0.00440.01750.05400.12950.24200.35210.39890.35210.24200.1295
        0.05400.01750.0044
```

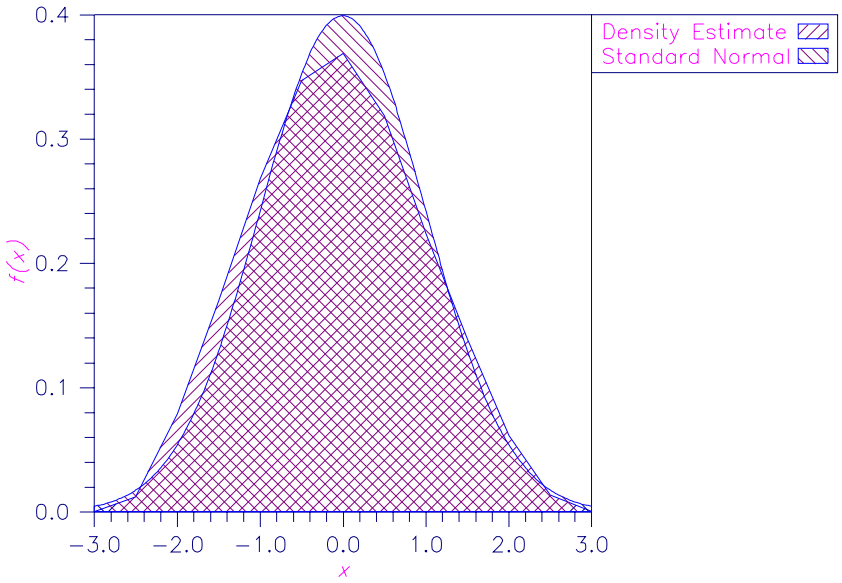


Figure 15-2 Density Estimate and Standard Normal Density

Comments

1. Informational error

Type	Code	
4	7	Negative kernel functions are not allowed.

2. Routine DESPT (page 1185) may be used to obtain interpolated density estimates from the NXPT density estimates returned in DENS. Array AMESH in DESPT corresponds to array XPT in DESKN.

Description

Routine DESKN computes kernel estimates of the density function for a random sample of (scalar-valued) observations. The kernel estimate of the density at the point y is given by.

$$\hat{f}(y) = \frac{1}{nh}\sum_{i=1}^{n} K[(y - x_i)/h]$$

where

$$\hat{f}(y)$$

is the estimated density at y, K is the kernel function, x_i denotes the i-th observation, n is the number of observations, and h is a fixed constant (called the "window width") supplied by the user.

One is usually interested in computing the density estimates using several values of the window width h. Tapia and Thompson (1978), Chapter 2, give some considerations relevant to the

choice of h. Some common kernel functions (see Tapia and Thompson 1978, page 60) are given as follows.

Name	Function
Uniform	$K(y) = \begin{cases} 0.5 & \text{for} \lvert y \rvert < 1 \\ 0 & \text{elsewhere} \end{cases}$
Triangular	$K(y) = \begin{cases} 1 - \lvert y \rvert & \text{for} \lvert y \rvert < 1 \\ 0 & \text{elsewhere} \end{cases}$
Biweight	$K(y) = \begin{cases} 15\left(1 - y^2\right)^2 / 16 & \text{for} \lvert y \rvert < 1 \\ 0 & \text{elsewhere} \end{cases}$
Normal	$K(y) = \frac{1}{\sqrt{2\pi}} e^{-y^2/2} \quad -\infty < y < \infty$

The computation can be made much more efficient when the kernel is nonzero over a finite range since observations outside this range can be ignored in the computation of the density. In this case, the array XPT is assumed to be sorted.

DNFFT

Computes Gaussian kernel estimates of a univariate density via the fast Fourier transform over a fixed interval.

Required Arguments

X — Vector of length NOBS containing the data for which a univariate density estimate is desired. (Input)
X is not referenced and may be dimensioned of length 1 in the calling program if IFFT = 1.

BNDS — Vector of length 2 containing the minimum and maximum values of X at which the density is to be estimated. (Input)
Observations less than BNDS(1) or greater than BNDS(2) are ignored. If either range of the hypothesized density is infinite, a value equal to the smallest observation minus 3 * WINDOW is a good choice for BNDS(1), and a value equal to the largest observation plus 3 * WINDOW is a good choice for BNDS(2). Let STEP = (BNDS(2) − BNDS(1))/(NXPT − 1), and note that the density is estimated at the points BNDS(1) + *i* STEP where *i* = 0, 1, …, NXPT − 1. The density is assumed constant over the interval from BNDS(1) + *i* * STEP to BNDS(1) + (*i* + 1) * STEP.

WINDOW — Window width for the kernel function. (Input)
Generally, several different values for WINDOW should be tried. When several different values are tried, use the IFFT option.

COEF — Vector of length NXPT containing the Fourier coefficients. (Input, if IFFT= 1; output, otherwise)

DENS — Vector of length NXPT containing the density estimates. (Output)
The density is estimated at the points BNDS(1) + *i* * STEP, *i* = 0, 1, …, NXPT − 1, where STEP = (BNDS (2) − BNDS(1))/(NXPT − 1).

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

FRQ — Vector of length NOBS containing the frequency of the corresponding element of X. (Input)
If FRQ(1) is −1.0, then the vector FRQ is not used and all frequencies are taken to be one. FRQ is also not used if IFFT = 1. In either case, FRQ may be dimensioned of length 1 in the calling program.
Default: FRQ(1) = −1.0.

IFFT — Fourier transform option parameter. (Input)
If IFFT = 1, then COEF contains the Fourier coefficients on input, and the coefficients are not computed. Otherwise, the coefficients are computed. This option is used when several different values for WINDOW are to be tried. On the first call to DNFFT, IFFT = 0 and the coefficients COEF are computed. On subsequent calls, these coefficients do not need to be recomputed (but only if NXPT also remains fixed).
Default: IFFT = 0.

NXPT — Number of equally-spaced points points at which the density is to be estimated. (Input)
Routine DNFFT is most efficient when NXPT is a power of 2. Little efficiency is lost if NXPT is a product of small primes. Because of the method of estimation, NXPT should be large, say greater than 64.
Default: NXPT = 128.

NRMISS — Number of rows of data that contain missing values in X or FRQ. (Output)
NRMISS is not referenced if IFFT = 1.

FORTRAN 90 Interface

Generic: `CALL DNFFT (X, BNDS, WINDOW, COEF, DENS [,…])`

Specific: The specific interface names are S_DNFFT and D_DNFFT.

FORTRAN 77 Interface

Single:
```
CALL DNFFT (NOBS, X, FRQ, BNDS, WINDOW, IFFT, NXPT, COEF,
DENS, NRMISS)
```

Double: The double precision name is DDNFFT.

Example

In this example, the density function is estimated at 64 points using a random sample of 150 points from a standard normal distribution. The actual density for the standard normal density is also reported in the output for comparison. The random sample is generated using routines RNSET and RNNOR in Chapter 18, Random Number Generation.

```
      USE RNSET_INT
      USE RNNOR_INT
      USE DNFFT_INT
      USE UMACH_INT
      INTEGER    NOBS, NXPT
      REAL       CONS, WINDOW
      PARAMETER  (CONS=0.39894228, NOBS=150, NXPT=64, WINDOW=0.25)
!
      INTEGER    I, NOUT
      REAL       BNDS(2), COEF(NXPT), DENS(NXPT), EXP, STEP, X(NOBS), XX
      INTRINSIC  EXP
!
      DATA BNDS/-4.0, 3.875/
!
      CALL RNSET (123457)
      CALL RNNOR (X)
!
      CALL DNFFT (X, BNDS, WINDOW, COEF, DENS, NXPT=NXPT)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99998)
99998 FORMAT ('   X  DENSITY POPULATION')
      STEP = (BNDS(2)-BNDS(1))/(NXPT-1)
      XX   = BNDS(1)
      DO 10  I=1, NXPT, 2
         WRITE (NOUT,99999) XX, DENS(I), CONS*EXP(-XX*XX/2.0)
99999    FORMAT (F6.2, 2F8.4)
         XX = XX + STEP*2.0
   10 CONTINUE
!
      END
```

Output

```
X  DENSITY POPULATION
-4.00  0.0000  0.0001
-3.75  0.0000  0.0004
-3.50  0.0000  0.0009
-3.25  0.0000  0.0020
-3.00  0.0001  0.0044
-2.75  0.0011  0.0091
-2.50  0.0089  0.0175
-2.25  0.0345  0.0317
-2.00  0.0772  0.0540
-1.75  0.1204  0.0863
```

```
-1.50  0.1573  0.1295
-1.25  0.2076  0.1826
-1.00  0.2682  0.2420
-0.75  0.2987  0.3011
-0.50  0.2976  0.3521
-0.25  0.3072  0.3867
 0.00  0.3336  0.3989
 0.25  0.3458  0.3867
 0.50  0.3169  0.3521
 0.75  0.2834  0.3011
 1.00  0.2683  0.2420
 1.25  0.2242  0.1826
 1.50  0.1557  0.1295
 1.75  0.1182  0.0863
 2.00  0.0946  0.0540
 2.25  0.0569  0.0317
 2.50  0.0199  0.0175
 2.75  0.0033  0.0091
 3.00  0.0002  0.0044
 3.25  0.0000  0.0020
 3.50  0.0000  0.0009
 3.75  0.0000  0.0004
```

Comments

1. Workspace may be explicitly provided, if desired, by use of D2FFT/DD2FFT. The reference is:

   ```
   CALL D2FFT (NOBS, X, FRQ, BNDS, WINDOW, IFFT, NXPT,COEF,DENS,
   NRMISS, WFFTR)
   ```

   The additional argument is:

   WFFTR – Work vector of length 2 * NXPT + 15. See Comment 3. (Input)

2. Informational errors

Type	Code	
4	1	The sum of the frequencies must be positive.
4	2	Each frequency must be nonnegative.
4	3	There are no valid observations remaining after all missing values are eliminated.

3. WFFTR is computed in DNFFT. If D2FFT is to be called, WFFTR must first be computed via the following FORTRAN statement:

   ```
   CALL FFTRI (NXPT, WFFTR)
   ```

   If DD2FFT is used, call DFFTRI instead of FFTRI. WFFTR need not be recomputed between successive calls to D2FFT if NXPT does not change.

Description

Routine DNFFT computes Gaussian kernel estimates of the density function for a random sample of (scalar-valued) observations using a Gaussian kernel (normal density). The computations are comparatively fast because they are performed through the use of the fast Fourier transform. Routine DESKN (page 1176) should be used in place of DNFFT if a kernel other than the Gaussian kernel is to be used, if a irregular grid is desired, or if the approximations in DNFFT are not acceptable. Because of its speed, DNFFT will usually be preferred to DESKN.

A Gaussian kernel estimate of the density at the point y is given by:

$$\hat{f}(y) = \frac{1}{nh}\sum_{i=1}^{n}\frac{1}{\sqrt{2\pi}}\exp\left[-\frac{1}{2}\left(\frac{y-x_i}{h}\right)^2\right]$$

where

$$\hat{f}(y)$$

is the estimated density at y, x_i denotes the i-th observation, n is the number of observations, and h is a fixed constant called the *window width* supplied by the user. If density estimates for several different window sizes are to be computed, then DNFFT performs a fast Fourier transform on the data only during the first call (when IFFT is zero). On subsequent calls (with IFFT set at 1), the Fourier transform of the data need not be recomputed.

If the same value of NXPT is to be used with several different input vectors X, then the computations can be made faster by the use of D2FFT. In D2FFT, it is assumed that some constants required by the Fourier transform and its inverse have already been computed via routine FFTRI (IMSL MATH/LIBRARY) in work array WFFTR. If D2FFT is called repeatedly with the same value of NXPT, WFTTR need only be computed once.

Routine DNFFT is an implementation of *Applied Statistics* algorithm AS 176 (Silverman 1982) using IMSL routines for the fast Fourier transforms. Modification to algorithm AS 176, as discussed in Silverman (1986, pages 61–66), gives the details of the computational method. The basic idea is to partition the support of the density into NXPT equally-sized nonoverlapping intervals. The frequency of the observations within each interval is then computed, and the Fourier transform of the frequencies obtained. Since the kernel density estimate is the convolution of the frequencies with the Gaussian kernel (for given window size), the Fourier coefficients for the Gaussian kernel density estimates are computed as the product of the coefficients obtained for the frequencies, times the Fourier coefficients for the Gaussian kernel function. The discrete Fourier coefficients for the Gaussian kernel may be estimated from the continuous transform. The inverse transform is then used to to obtain the density estimates.

Because the fast Fourier transform is used in computing

$$\hat{f}(y)$$

the computations are relatively fast (providing that NXPT is a product of small primes). To maintain precision, a large number of intervals, say 256, is usually recommended. Tapia and Thompson (1978), Chapter 2, give some considerations relevant to the choice of the window size parameter WINDOW. Generally, several different window sizes should be tried in order to obtain the best value for this parameter.

DESPT

Estimates a probability density function at specified points using linear or cubic interpolation.

Required Arguments

XPT — Vector of length NODE containing the points at which an estimate of the probability density is desired. (Input)

AMESH — Vector of length NORD for IOPT = 2 or 4, and of length 2 for IOPT = 1 or 3. (Input)
If IOPT = 2 or 4, AMESH(I) contains the abscissas corresponding to each density estimate in DENS(I). In this case, the abscissas must be specified in increasing order. If IOPT = 1 or 3 (i.e., for an equally spaced mesh), then the lower and upper ends of the mesh are specified by AMESH(1) and AMESH(2), respectively, with the increment between mesh points given by (AMESH(2) – AMESH(1))/(NORD – 1).

DENS — Vector of length NORD containing the density function values corresponding to each of the NORD abscissa values. (Input)

DENEST — Vector of length NODE containing the density function estimates for the points in XPT. (Output)

Optional Arguments

NODE — Number of points at which the density is desired. (Input)
Default: NODE = size (XPT,1).

IOPT — Interpolation option parameter. (Input)
Default: IOPT = 1.

IOPT	Method of interpolation
1	Linear on equally spaced points
2	Linear with unequal spacing
3	Cubic on equally spaced points
4	Cubic with unequal spacing

NORD — Number of ordinates supplied. (Input)
NORD must be greater than one for linear interpolation, and greater than three for cubic interpolation.
Default: NORD = size (DENS,1).

FORTRAN 90 Interface

Generic: `CALL DESPT (XPT, AMESH, DENS, DENEST [,…])`

Specific: The specific interface names are `S_DESPT` and `D_DESPT`.

FORTRAN 77 Interface

Single: `CALL DESPT (NODE, XPT, IOPT, NORD, AMESH, DENS, DENEST)`

Double: The double precision name is `DDESPT`.

Example

The standard normal density is to be estimated via a grid of points over which the density is provided. Grid points are given by (0.0, 0.5, 1.0, 1.5, 2.0) while the density is to be estimated (via linear interpolation) at the four points (0.25, 0.75, 1.25, 1.75). For comparison, both the exact and the estimated density values at each of the four points are printed.

```
      USE DESPT_INT
      USE UMACH_INT
      INTEGER    NODE, NORD
      PARAMETER  (NODE=4, NORD=5)
!
      INTEGER    I, NOUT
      REAL       AMESH(2), DENEST(NODE), DENS(NORD), EXP, F, H, X, X0, &
                 XPT(NODE)
      INTRINSIC  EXP
!
      DATA XPT/0.25, 0.75, 1.25, 1.75/
      DATA AMESH/0, 2/
!
      F(X) = 0.3989423*EXP(-X*X/2.0)
!                                Get the grid values
      H  = (AMESH(2)-AMESH(1))/(NORD-1)
      X0 = AMESH(1)
      DO 10  I=1, NORD
         DENS(I) = F(X0)
         X0      = X0 + H
   10 CONTINUE
!                                Get the density estimates
      CALL DESPT (XPT, AMESH, DENS, DENEST)
!                                Print the results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,'(''    X       DENEST      EXACT'')')
      DO 20  I=1, NODE
         WRITE (NOUT,'(F5.2, 2F12.5)') XPT(I), DENEST(I), F(XPT(I))
   20 CONTINUE
      END
```

Output

```
X       DENEST      EXACT
```

```
0.25      0.37550      0.38667
0.75      0.29702      0.30114
1.25      0.18574      0.18265
1.75      0.09175      0.08628
```

Comments

1. Workspace may be explicitly provided, if desired, by use of D2SPT/DD2SPT . The reference is:

   ```
   CALL D2SPT (NODE, XPT, IOPT, NORD, AMESH, DENS,
   DENEST, CF, X, BREAK)
   ```

   The additional arguments are as follows:

   CF — Work vector of length 4 * NORD for IOPT = 3 or 4. CF is not used for other values of IOPT and may be dimensioned of length 1.

   X — Work vector of length NORD for IOPT = 3 or 4. X is not used for other values of IOPT and may be dimensioned of length 1.

   BREAK — Work vector of length NORD for IOPT = 3 or 4. BREAK is not used for other values of IOPT and may be dimensioned of length 1.

2. Array AMESH is the same as array BNDS in DESPL (page 1172) when IOPT is 1 or 3, and the same as array XPT in DESKN (page 1176) when IOPT is 2 or 4.

Description

Routine DESPT computes an estimate of a density function using either linear or cubic spline interpolation on a set $\{(X_i, F_i)$, for $i = 1, \ldots, N\}$, where F_i = DENS(i), N = NODE, and where the values of the the grid points X_i can be obtained from the vector AMESH. The value of IOPT indicates the type of interpolation (linear or cubic) to be performed and whether the mesh values are equally spaced. When IOPT is 1 or 3, then an equally spaced mesh is used with mesh values given by

$$\text{AMESH}\,(1) + i * \text{DELTA}$$

for $i = 0, 1, \ldots, N - 1$, where

$$\text{DELTA} = (\text{AMESH}(2) - \text{AMESH}(1))/(\text{NORD} - 1)$$

IOPT = 2 or 4 yields an unequally spaced mesh with all mesh values contained in the vector AMESH.

The Akima cubic spline method of interpolation (Akima 1970) is used for the cubic interpolation.

HAZRD

Performs nonparametric hazard rate estimation using kernel functions and quasi-likelihoods.

Required Arguments

X — NOBS by *m* matrix containing the raw data, where *m* = 1 if ICEN = 0, and *m* = 2 otherwise. (Input)

IRT — Column number in X of the event times. (Input)

KMIN — Minimum number for parameter *k*. (Input)
Parameter *k* is the number of nearest neighbors to be used in computing the *k*-th nearest neighbor distance.

INK — Increment between successive values of parameter *k*. (Input)

NK — Number of values of *k* to be considered. (Input)
HAZRD finds the optimal value of *k* over the grid given by: KMIN + (*j* – 1) * INK, for *j* = 1, …, NK.

ST — Vector of length NOBS containing the times of occurrence of the events, sorted from smallest to largest. (Output)
Vector ST is obtained from the matrix X and should be used as input to routine HAZST (page 1203).

JCEN — Vector of length NOBS containing the sorted censor codes. (Output) Censor codes are sorted corresponding to the events ST(*i*), with censored observations preceding tied failures. Vector JCEN is obtained from the censor codes in X, if present, and is used as input to routine HAZST (page 1203).

ALPHA — Optimal estimate for the parameter α. (Output)

BTA — Optimal estimate for the parameter β. (Output)

K — Optimal estimate for the parameter *k*. (Output)

VML — Optimum value of the criterion function. (Output)

H — Vector of length NOBS * 5 containing constants needed to compute the *k*-th nearest failure distances, and the observation weights. (Output)
H is used as input to routine HAZST (page 1203).

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

ICEN — Censoring option. (Input)
If ICEN = 0, then all of the data is treated as exact data with no censoring. For ICEN > 0, column ICEN of X contains the censoring codes. A censoring code of 0 means an exact event (failure). A censoring code of 1 means that the observation was right censored at the event time.
Default: ICEN = 0.

IWTO — Weight option. (Input)
If IWTO = 1, then weight $\ln(1 + 1/(\text{NOBS} - i + 1))$ is used for the i-th smallest observation. Otherwise, weight $1/(\text{NOBS} - i + 1)$ is used.
Default: IWTO = 0.

NGRID — Grid option. (Input)
If NGRID = 0, a default grid is used to locate an initial starting value for parameter BTA. For NGRID > 0, a user-defined grid is used. This grid is defined as BSTART + (j – 1) * GINC, for j = 1, …, NGRID, where BSTART, GINC, and NGRID are input.
Default: NGRID = 0.

BSTART — First value to be used in the user-defined grid. (Input)
Not used if NGRID = 0.

GINC — For a user-defined grid, the increment between successive grid values of BTA. (Input)
Not used if NGRID = 0.

IPRINT — Printing option. (Input)
If IPRINT = 1, the grid estimates and the optimized estimates are printed for each value of k. Otherwise, no printing is performed.
Default: IPRINT = 0.

ISORT — Sorting option. (Input)
If ISORT = 1, then the event times are not automatically sorted by HAZRD. Otherwise, sorting is performed with exact failure times following tied right-censored times.
Default: ISORT = 0.

NMISS — Number of missing (NaN, not a number) values in X. (Output)

FORTRAN 90 Interface

Generic: `CALL HAZRD (X, IRT, KMIN, INK, NK, ST, JCEN, ALPHA, BTA, K, VML, H [,…])`

Specific: The specific interface names are S_HAZRD and D_HAZRD.

FORTRAN 77 Interface

Single: `CALL HAZRD (NOBS, X, LDX, IRT, ICEN, IWTO, NGRID, BSTART, GINC, KMIN, INK, NK, IPRINT, ISORT, ST, JCEN,ALPHA, BTA, K, VML, H, NMISS)`

Double: The double precision name is `DHAZRD`.

Example

The following example is taken from Tanner and Wong (1984). The data are from Stablein, Carter, and Novak (1981) and involve the survival times of individuals with nonresectable gastric carcinoma. Individuals treated with radiation and chemotherapy are used. For each value of k from 18 to 22 with increment of 2, the default grid search for β is performed. Using the optimal value of β in the grid, the optimal parameter estimates of α and β are computed for each value of k. The final solution is the parameter estimates for the value of k which optimizes the modified likelihood (`VML`). Because the `IPRINT` = 1 option is in effect, `HAZRD` prints all of the results in the output.

```
      USE HAZRD_INT
      USE UMACH_INT
      USE WRRRN_INT
      USE WRIRN_INT
      INTEGER    ICEN, INK, IPRINT, IRT, ISORT, KMIN, LDX, &
                 NK, NOBS
      PARAMETER  (ICEN=2, INK=2, IPRINT=1, IRT=1, ISORT=1, &
                  KMIN=18, LDX=45, NK=3, NOBS=45)
!
      INTEGER    JCEN(NOBS), K, NMISS, NOUT
      REAL       ALPHA, BTA, H(5*NOBS), ST(NOBS), VML, X(NOBS,2)
!
      DATA X/17, 42, 44, 48, 60, 72, 74, 95, 103, 108, 122, 144, 167, &
          170, 183, 185, 193, 195, 197, 208, 234, 235, 254, 307, 315, &
          401, 445, 464, 484, 528, 542, 567, 577, 580, 795, 855, 882, &
          892, 1031, 1033, 1306, 1335, 1366, 1452, 1472, 36*0, 9*1/
!
      CALL HAZRD (X, IRT, KMIN, INK, NK, ST, JCEN, ALPHA, BTA, &
                  K, VML, H, ICEN=ICEN, IPRINT=IPRINT, ISORT=ISORT, &
                  NMISS=NMISS)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) NMISS
99999 FORMAT (/' NMISS = ', I4/)
      CALL WRRRN ('ST', ST, 1, NOBS, 1)
      CALL WRIRN ('JCEN', JCEN, 1, NOBS, 1)
      CALL WRRRN ('H', H, NOBS, 5, NOBS)
      END
```

Output

```
                *** GRID SEARCH FOR K =    18 ***
          ALPHA                   BETA                   VML
          4.578322            2980.958008            -266.804504
          4.543117              54.598148            -266.619690
```

```
          4.336464             20.085537            -265.541168
          4.019334             12.182494            -264.001404
          3.542742              7.389056            -262.540100
          2.990577              4.481689            -262.511810
          2.351537              2.718282            -262.633911
          1.584173              1.648721            -262.158264
          0.966332              1.000000            -262.868408

                *** OPTIMAL PARAMETER ESTIMATES ***
          ALPHA                  BETA                   VML
          1.695147              1.769263            -262.118530

                *** GRID SEARCH FOR K =    20 ***
          ALPHA                  BETA                   VML
          4.053934           2980.958008            -266.525970
          4.032835             54.598148            -266.401428
          3.905046             20.085537            -265.648315
          3.687815             12.182494            -264.401672
          3.304344              7.389056            -262.665924
          2.822716              4.481689            -262.080078
          2.252759              2.718282            -262.445251
          1.555777              1.648721            -261.772278
          0.955586              1.000000            -262.617645

                *** OPTIMAL PARAMETER ESTIMATES ***
          ALPHA                  BETA                   VML
          1.540533              1.631551            -261.771484

                *** GRID SEARCH FOR K =    22 ***
          ALPHA                  BETA                   VML
          3.656405           2980.958008            -267.595337
          3.641593             54.598148            -267.498596
          3.550560             20.085537            -266.903870
          3.388752             12.182494            -265.859131
          3.071474              7.389056            -264.066040
          2.645036              4.481689            -263.038696
          2.137399              2.718282            -263.334717
          1.512606              1.648721            -262.639740
          0.936368              1.000000            -262.682739

                *** OPTIMAL PARAMETER ESTIMATES ***
          ALPHA                  BETA                   VML
          1.342176              1.450016            -262.561188

             *** THE FINAL SOLUTION     (K =    20) ***
          ALPHA                  BETA                   VML
          1.540533              1.631551            -261.771484
NMISS =    0

                                  ST
      1        2        3        4        5        6        7        8
   17.0     42.0     44.0     48.0     60.0     72.0     74.0     95.0

      9       10       11       12       13       14       15       16
  103.0    108.0    122.0    144.0    167.0    170.0    183.0    185.0
```

17	18	19	20	21	22	23	24
193.0	195.0	197.0	208.0	234.0	235.0	254.0	307.0

25	26	27	28	29	30	31	32
315.0	401.0	445.0	464.0	484.0	528.0	542.0	567.0

33	34	35	36	37	38	39	40
577.0	580.0	795.0	855.0	882.0	892.0	1031.0	1033.0

41	42	43	44	45
1306.0	1335.0	1366.0	1452.0	1472.0

JCEN

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

41	42	43	44	45
1	1	1	1	1

H

	1	2	3	4	5
1	217.0	218.0	1.0	21.0	1.0
2	192.0	193.0	1.0	21.0	0.5
3	190.0	191.0	1.0	21.0	0.3
4	186.0	187.0	1.0	21.0	0.2
5	174.0	175.0	1.0	21.0	0.2
6	162.0	163.0	1.0	21.0	0.2
7	160.0	161.0	1.0	21.0	0.1
8	139.0	140.0	1.0	21.0	0.1
9	131.0	132.0	1.0	21.0	0.1
10	126.0	127.0	1.0	21.0	0.1
11	112.0	113.0	1.0	21.0	0.1
12	102.0	110.0	2.0	22.0	0.1
13	123.0	125.0	3.0	23.0	0.1
14	126.0	128.0	3.0	23.0	0.1
15	132.0	135.0	5.0	25.0	0.1
16	130.0	137.0	5.0	25.0	0.1
17	133.0	145.0	5.0	25.0	0.1
18	135.0	147.0	5.0	25.0	0.1
19	137.0	149.0	5.0	25.0	0.1
20	148.0	160.0	5.0	25.0	0.1
21	167.0	174.0	6.0	26.0	0.0
22	166.0	175.0	6.0	26.0	0.0
23	182.0	191.0	6.0	26.0	0.0
24	204.0	212.0	9.0	29.0	0.0
25	212.0	213.0	9.0	29.0	0.0
26	231.0	234.0	14.0	34.0	0.0
27	275.0	278.0	14.0	34.0	0.0
28	294.0	297.0	14.0	34.0	0.0
29	311.0	314.0	15.0	35.0	0.0
30	343.0	345.0	16.0	36.0	0.0

```
31    357.0    359.0     16.0     38.0      0.0
32    382.0    384.0     16.0     38.0      0.0
33    392.0    394.0     16.0     38.0      0.0
34    395.0    397.0     16.0     38.0      0.0
35    610.0    612.0     16.0     43.0      0.0
36    670.0    672.0     16.0     45.0      0.0
37    689.0    697.0     17.0     45.0      0.0
38    699.0    707.0     17.0     45.0      0.0
39    838.0    846.0     17.0     45.0      0.0
40    840.0    848.0     17.0     45.0      0.0
41   1113.0   1121.0     17.0     45.0      0.0
42   1142.0   1150.0     17.0     45.0      0.0
43   1173.0   1181.0     17.0     45.0      0.0
44   1259.0   1267.0     17.0     45.0      0.0
45   1279.0   1287.0     17.0     45.0      0.0
```

Comments

1. Informational Errors

Type	Code	
4	18	All observations are missing (NaN, not a number) values.

2. In the optimization routines, the parameterization is changed to β^* and α^*, where $\beta^* = -\ln(\beta)$ and $\alpha^* = -\ln(\alpha)$. The default grid uses –8, –4, –3, –2.5, –2, –1.5, –1, –0.5, and 0.5 for β^*. This corresponds to a grid in β of 2981, 54.6, 20.08, 12.18, 7.39, 4.48, 2.72, 1.64, and .61. The grid β that maximizes the modified "likelihood" is used as the starting point for the iterations.

3. If the initial estimate of β as determined from the grid or as given by the user is greater than 400 (actually e^6), then infinite β is assumed, and an analytic estimate of α based upon infinite β is used. In the optimization, if it is determined that β must be greater than 1000, then an infinite β is assumed. Infinite β corresponds to a "flat" hazard rate.

Description

Routine HAZRD is an implementation of the methods discussed by Tanner and Wong (1984) for estimating the hazard rate in survival or reliability data with right censoring. It uses the biweight kernel,

$$K(x) = \begin{cases} \frac{15}{16}(1-x^2)^2 & \text{for } |x| < 1 \\ 0 & \text{elsewhere} \end{cases}$$

and a modified likelihood to obtain data-based estimates of the smoothing parameters α, β, and k needed in the estimation of the hazard rate. For kernel $K(x)$, define the "smoothed" kernel $K_s(x - x_{(j)})$ as follows:

$$K_S(x - x_{(j)}) = \frac{1}{\alpha d_{jk}} K\left(\frac{x - x_{(j)}}{\beta d_{jk}}\right)$$

where d_{jk} is the distance to the k-th nearest failure from $x(j)$, and $x(j)$ is the j-th ordered observation (from smallest to largest). For given α and β, the hazard at point x is then

$$h(x) = \sum_{i=1}^{N} \{(1-\delta_i) w_i K_s(x - x_{(i)})\}$$

where N = NOBS, δ_i is the i-th observation's censor code (1 = censored, 0 = failed), and w_i is the i-th ordered observation's weight, which may be chosen as either $1/(N - i + 1)$, or $\ln(1 + 1/(N - i + 1))$. After the smoothing parameters have been obtained, the hazard may be estimated via HAZST (page 1203).

Let

$$H(x) = \int_o^x h(s)ds$$

The likelihood is given by

$$L = \prod_{i=1}^{N} \{h(x_i)^{(1-\delta_i)} \exp(-H(x_{(i)}))\},$$

where Π denotes product. Since the likelihood leads to degenerate estimates, Tanner and Wong (1984) suggest the use of a modified likelihood. The modification consists of deleting observation x_i in the calculation of $h(x_i)$ and $H(x_i)$ when the likelihood term for x_i is computed using the usual optimization techniques. α and β for given k can then be estimated.

Estimates for α and β are computed as follows: for given β, a closed form solution is available for α. The problem is thus reduced to the estimation of β. A grid search for β is first performed. Experience indicates that if the initial estimate of β from this grid search is greater than, say, e^6, then the modified likelihood is degenerate because the hazard rate does not change with time. In this situation, β should be taken to be infinite, and an estimate of α corresponding to infinite β should be directly computed. When the estimate of β from the grid search is less than e^6, a secant algorithm is used to optimize the modified likelihood. The secant algorithm iteration stops when the change in β from one iteration to the next is less than 10^{-5}. Alternatively, the iterations may cease when the value of β becomes greater than e^6, at which point an infinite β with a degenerate likelihood is assumed.

To find the optimum value of the likelihood with respect to k, a user-specified grid of k-values is used. For each grid value, the modified likelihood is optimized with respect to $\alpha$and β. That grid point, which leads to the smallest likelihood, is taken to be the optimal k.

Programming Notes

1. The routine HAZST (page 1203) may be used to estimate the hazard on a grid of points once the optimal values for α, β and k have been found. The user should also consider using the "easy-to-use" version of HAZRD, routine HAZEZ (page 1195).

2. If sorting of the data is performed by HAZRD, then the sorted array will be such that all censored observations at a given time precede all failures at that time. To specify an arbitrary pattern of censored/failed observations at a given time point, the ISORT = 1 option must be used. In this case, it is assumed that the times have already been sorted from smallest to largest.

3. The smallest value of k must be greater than the largest number of tied failures since d_{jk} must be positive for all j. (Censored observations are not counted.) Similarly, the largest value of k must be less than the total number of failures. If the grid specified for k includes values outside the allowable range, then a warning error is issued; but k is still optimized over the allowable grid values.

4. The secant algorithm iterates on the transformed parameter $\beta^* = \exp(-\beta)$. This assures a positive β, and it also seems to lead to a more desirable grid search. All results returned to the user are in the original parameterization, however.

5. Since local minimums have been observed in the modified likelihood, it is recommended that more than one grid of initial values for α and β be used.

HAZEZ

Performs nonparametric hazard rate estimation using kernel functions. Easy-to-use version of HAZRD.

Required Arguments

X — NOBS by *m* matrix containing the raw data, where *m* = 1 if ICEN = 0, and *m* = 2 otherwise. (Input)

IRT — Column number in X containing the times of occurrence of the events. (Input)

ST — Vector of length NOBS containing the times of occurrence of the events, sorted from smallest to largest. (Output)
Vector ST is obtained from matrix X and is used as input to routine HAZST (page 1203).

JCEN — Vector of length NOBS containing the sorted censor codes. (Output)
Censor codes are sorted corresponding to the events ST(*i*), with censored observations preceding tied failures. Vector JCEN is obtained from the censor codes in X and is used as input to routine HAZST (page 1203).

ALPHA — Optimal estimate for the parameter α. (Output)

BTA — Optimal estimate for the parameter β. (Output)

K — Optimal estimate for the parameter *k*. (Output)

VML — Optimal value of the criterion function. (Output)
VML is the "modified likelihood".

H — Vector of length 5 * NOBS containing the constants needed to compute the *k*-th nearest failure distance and the observation weights. (Output)
H is used as input to routine HAZST (page 1203).

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

ICEN — Censoring option. (Input)
If ICEN = 0, then all of the data is treated as exact data with no censoring. For ICEN > 0, column ICEN of X contains the censoring codes. A censoring code of 0 means an exact event (failure). A censoring code of 1 means that the observation was right censored at the event time.
Default: ICEN = 0.

IPRINT — Printing option. (Input)
If IPRINT = 1, the grid estimates and the optimized estimates are printed for each value of *k*. Otherwise, no printing is performed.
Default : IPRINT = 0.

NMISS — Number of missing (NaN, not a number) values in X. (Output)

FORTRAN 90 Interface

Generic: `CALL HAZEZ (X, IRT, ST, JCEN, ALPHA, BTA, K, VML, H [,…])`

Specific: The specific interface names are S_HAZEZ and D_HAZEZ.

FORTRAN 77 Interface

Single:
```
CALL HAZEZ (NOBS, X, LDX, IRT, ICEN, IPRINT, ST, JCEN,
ALPHA, BTA, K, VML, H, NMISS)
```

Double: The double precision name is DHAZEZ.

Example

The following example is illustrated in Tanner and Wong (1984), and the data are taken from Stablein, Carter, and Novak (1981). It involves the survival times of individuals with nonresectable gastric carcinoma. Only those individuals treated with radiation and chemotherapy are used.

```
USE HAZEZ_INT
USE UMACH_INT
USE WRRRN_INT
USE WRIRN_INT
INTEGER    ICEN, IPRINT, IRT, LDX, NOBS
PARAMETER  (ICEN=2, IPRINT=1, IRT=1, LDX=45, NOBS=45)
```

```
!
      INTEGER    JCEN(NOBS), K, NMISS, NOUT
      REAL       ALPHA, BTA, H(5*NOBS), ST(NOBS), VML, X(NOBS,2)
!
      DATA X/17, 42, 44, 48, 60, 72, 74, 95, 103, 108, 122, 144, 167, &
          170, 183, 185, 193, 195, 197, 208, 234, 235, 254, 307, 315, &
          401, 445, 464, 484, 528, 542, 567, 577, 580, 795, 855, 882, &
          892, 1031, 1033, 1306, 1335, 1366, 1452, 1472, 36*0, 9*1/
!
      CALL HAZEZ (X, IRT, ST, JCEN, ALPHA, BTA, K, VML, H, ICEN=ICEN, &
                  IPRINT=IPRINT, NMISS=NMISS)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) NMISS
99999 FORMAT (/' NMISS = ', I4/)
      CALL WRRRN ('ST', ST, 1, NOBS, 1)
      CALL WRIRN ('JCEN', JCEN, 1, NOBS, 1)
      CALL WRRRN ('H', H, NOBS, 5, NOBS)
      END
```

Output

```
              *** GRID SEARCH FOR K =     2 ***
          ALPHA                   BETA                    VML
       65.157967            2980.958008            -266.543945
       32.434208              54.598148            -262.551147
       17.100269              20.085537            -263.100769
       11.402525              12.182494            -264.410187
        7.263529               7.389056            -267.502014
        4.452315               4.481689            -270.548523
        2.689497               2.718282            -335.407288
        1.633702               1.648721            -338.566162
        0.995799               1.000000            -519.939514

              *** OPTIMAL PARAMETER ESTIMATES ***
          ALPHA                   BETA                    VML
       32.219337              53.968315            -262.550781

              *** GRID SEARCH FOR K =     4 ***
          ALPHA                   BETA                    VML
       25.596716            2980.958008            -266.471558
       20.476425              54.598148            -262.893860
       13.995192              20.085537            -262.792755
       10.109113              12.182494            -262.573212
        6.883837               7.389056            -263.030121
        4.407142               4.481689            -265.238647
        2.690131               2.718282            -265.606293
        1.633339               1.648721            -266.822693
        0.993371               1.000000            -271.831390

              *** OPTIMAL PARAMETER ESTIMATES ***
          ALPHA                   BETA                    VML
        8.530729               9.683726            -262.545593

              *** GRID SEARCH FOR K =     6 ***
          ALPHA                   BETA                    VML
```

```
16.828691           2980.958008           -266.729248
14.840095             54.598148           -264.019409
11.215133             20.085537           -262.844360
 9.013870             12.182494           -263.663391
 6.557755              7.389056           -263.283752
 4.330785              4.481689           -263.732697
 2.691744              2.718282           -264.613800
 1.633932              1.648721           -265.381866
 0.990891              1.000000           -266.242767

       *** OPTIMAL PARAMETER ESTIMATES ***
 ALPHA                  BETA                   VML
12.553377             28.178671           -262.529877

       *** GRID SEARCH FOR K =     8 ***
 ALPHA                  BETA                   VML
11.377748           2980.958008           -266.746185
10.773529             54.598148           -265.469299
 8.766835             20.085537           -262.476807
 7.427887             12.182494           -263.109009
 5.916299              7.389056           -264.492432
 4.184323              4.481689           -264.289886
 2.656351              2.718282           -264.807617
 1.623750              1.648721           -265.270691
 0.989442              1.000000           -264.738403

       *** OPTIMAL PARAMETER ESTIMATES ***
 ALPHA                  BETA                   VML
 8.522110             18.281288           -262.438568

       *** GRID SEARCH FOR K =    10 ***
 ALPHA                  BETA                   VML
 8.689023           2980.958008           -267.026093
 8.412854             54.598148           -266.250366
 7.196551             20.085537           -263.192688
 6.207793             12.182494           -262.648376
 5.143391              7.389056           -264.274384
 3.934601              4.481689           -264.523193
 2.630993              2.718282           -264.877869
 1.611710              1.648721           -264.332581
 0.984530              1.000000           -263.920013

       *** OPTIMAL PARAMETER ESTIMATES ***
 ALPHA                  BETA                   VML
 6.483376             13.956067           -262.589661

       *** GRID SEARCH FOR K =    12 ***
 ALPHA                  BETA                   VML
 6.669007           2980.958008           -266.778259
 6.551508             54.598148           -266.347595
 5.933167             20.085537           -264.141174
 5.252526             12.182494           -262.516205
 4.471936              7.389056           -262.691589
 3.598284              4.481689           -263.914032
 2.557817              2.718282           -263.390106
```

```
1.588307              1.648721           -263.879578
0.973723              1.000000           -263.361908

       *** OPTIMAL PARAMETER ESTIMATES ***
ALPHA                  BETA                   VML
4.923379              9.819798           -262.336670

       *** GRID SEARCH FOR K =    14 ***
ALPHA                  BETA                   VML
5.668086           2980.958008           -266.747559
5.595870             54.598148           -266.436584
5.195685             20.085537           -264.737946
4.685275             12.182494           -262.971497
4.044650              7.389056           -262.288147
3.335586              4.481689           -263.126434
2.496436              2.718282           -262.891663
1.585763              1.648721           -263.418976
0.969140              1.000000           -263.164032

       *** OPTIMAL PARAMETER ESTIMATES ***
ALPHA                  BETA                   VML
4.145060              7.966486           -262.260559

       *** GRID SEARCH FOR K =    16 ***
ALPHA                  BETA                   VML
4.970138           2980.958008           -266.419281
4.924928             54.598148           -266.199646
4.663393             20.085537           -264.938660
4.280633             12.182494           -263.266602
3.741570              7.389056           -262.020355
3.132969              4.481689           -262.401733
2.421248              2.718282           -262.555817
1.586469              1.648721           -262.426025
0.967658              1.000000           -263.135101

       *** OPTIMAL PARAMETER ESTIMATES ***
ALPHA                  BETA                   VML
3.639074              6.767537           -261.987305

       *** GRID SEARCH FOR K =    18 ***
ALPHA                  BETA                   VML
4.578322           2980.958008           -266.804504
4.543117             54.598148           -266.619690
4.336464             20.085537           -265.541168
4.019334             12.182494           -264.001404
3.542742              7.389056           -262.540100
2.990577              4.481689           -262.511810
2.351537              2.718282           -262.633911
1.584173              1.648721           -262.158264
0.966332              1.000000           -262.868408

       *** OPTIMAL PARAMETER ESTIMATES ***
ALPHA                  BETA                   VML
1.695147              1.769263           -262.118530
```

```
          *** GRID SEARCH FOR K =    20 ***
      ALPHA                 BETA                  VML
      4.053934          2980.958008          -266.525970
      4.032835            54.598148          -266.401428
      3.905046            20.085537          -265.648315
      3.687815            12.182494          -264.401672
      3.304344             7.389056          -262.665924
      2.822716             4.481689          -262.080078
      2.252759             2.718282          -262.445251
      1.555777             1.648721          -261.772278
      0.955586             1.000000          -262.617645

          *** OPTIMAL PARAMETER ESTIMATES ***
      ALPHA                 BETA                  VML
      1.540533             1.631551          -261.771484

         *** THE FINAL SOLUTION     (K =    20) ***
      ALPHA                 BETA                  VML
      1.540533             1.631551          -261.771484

NMISS =    0

                              ST
     1        2        3        4        5        6        7        8
  17.0     42.0     44.0     48.0     60.0     72.0     74.0     95.0

     9       10       11       12       13       14       15       16
 103.0    108.0    122.0    144.0    167.0    170.0    183.0    185.0

    17       18       19       20       21       22       23       24
 193.0    195.0    197.0    208.0    234.0    235.0    254.0    307.0

    25       26       27       28       29       30       31       32
 315.0    401.0    445.0    464.0    484.0    528.0    542.0    567.0

    33       34       35       36       37       38       39       40
 577.0    580.0    795.0    855.0    882.0    892.0   1031.0   1033.0

    41       42       43       44       45
1306.0   1335.0   1366.0   1452.0   1472.0

                                     JCEN
 1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   1   1

41  42  43  44  45
 1   1   1   1   1

                         H
          1        2        3        4        5
 1    217.0    218.0      1.0     21.0      1.0
```

```
 2    192.0    193.0      1.0     21.0      0.5
 3    190.0    191.0      1.0     21.0      0.3
                      .
                      .
                      .
43   1173.0   1181.0     17.0     45.0      0.0
44   1259.0   1267.0     17.0     45.0      0.0
45   1279.0   1287.0     17.0     45.0      0.0
```

Comments

1. Informational errors

Type	Code	
4	6	All observations are missing (NaN, not a number) values.
4	7	There are not enough failing observations in X to continue.

2. The grid values in the initial grid search are given as follows: Let $\beta^* = -8, -4, -2, -1, -0.5, 0.5, 1$, and 2, and

$$\beta = e^{-\beta^*}$$

For each value of β, VML is computed at the optimizing β. The maximizing β is used to initiate the iterations.

3. If the initial β^* is determined from the grid search to be less than −6, then it is presumed that β is infinite, and an analytic estimate of α based upon infinite β is used. Infinite β corresponds to a flat hazard rate.

Description

Routine HAZEZ is an implementation of the methods discussed by Tanner and Wong (1984) for estimating the hazard rate in survival or reliability data with right censoring. It uses the biweight kernel,

$$K(x) = \begin{cases} \frac{15}{16}(1-x^2)^2 & \text{for} \left|x\right| < 1 \\ 0 & \text{elsewhere} \end{cases}$$

and a modified likelihood to obtain data-based estimates of the smoothing parameters α, β, and k needed in the estimation of the hazard rate. For kernel $K(x)$, define the "smoothed" kernel $K_s(x - x(j))$ as follows:

$$K_s(x - x_{(j)}) = \frac{1}{\alpha d_{jk}} K\left(\frac{x - x_{(j)}}{\beta d_{jk}}\right)$$

where d_{jk} is the distance to the k-th nearest failure from $x(j)$, and $x(j)$is the j-th ordered observation (from smallest to largest). For given α and β, the hazard at point x is given by:

$$h(x) = \sum_{i=1}^{N} \{(1-\delta_i) w_i K_s(x - x_{(i)})\}$$

where N = NOBS, δ_i is the censor code (0 = failed, 1 = censored) for the i-th ordered observation, and w_i is the weight of the i-th ordered observation (given by $1/(N - i + 1)$). The hazard may be estimated via routine HAZST (page 1203) after the smoothing parameters have been obtained

Let

$$H(x) = \int_0^x h(s)ds$$

The likelihood is given by:

$$L = \prod_{i=1}^{N} \{h(x_i)^{(1-\delta_i)} \exp(-H(x_i))\}$$

where Π denotes product. Since the likelihood, as specified, will lead to degenerate estimates, Tanner and Wong (1984) suggest the use of a modified likelihood. The modification consists of deleting the observation x_i in the calculation of $h(x_i)$ and $H(x_i)$ when the likelihood term for x_i is computed. For a given k, α and β can then be estimated via the usual optimization techniques.

Estimates for α and β are computed as follows. For a given β, a closed form solution is available for α. The problem is thus reduced to the estimation of β. To estimate α and β, a grid search is first performed. Experience indicates that if the initial estimate of β from this grid search is greater than exp(6), then the modified likelihood is degenerate because the hazard rate does not change with time. In this situation, β should be taken to be infinite, and an estimate of α corresponding to infinite β is computed directly. When the estimate of β from the grid search is less than exp(6) (approximately 400), a secant algorithm is used to optimize the modified likelihood. The secant algorithm is said to have converged when the change in β from one iteration to the next is less than 0.00001. Additionally, convergence is assumed when the value of β becomes greater than exp(6). This corresponds to an infinite β with a degenerate likelihood.

A grid of k-values is used to find the optimum value of the likelihood with respect to k. The grid is determined by HAZEZ and consists of at most 10 points. The starting value in the grid is the smallest possible value of k. An increment of 2 is then used to obtain the remaining grid points.

For each grid value, the modified likelihood is optimized with respect to α and β. That grid point, which leads to the smallest likelihood, is taken to be the optimal k.

Programming Notes

1. Routine HAZST (page 1203) may be used to estimate the hazard on a grid of points once the optimal values for α, β and k have been found. (The user should also consider using routine HAZRD (page 1188), which allows for more options than HAZEZ.)

2. Routine HAZEZ assumes that censored observations precede failed observations at tied failure/censoring times.

3. The secant algorithm iterates on the transformed parameter $\beta^* = \exp(-\beta)$. This assures a positive β, and it also seems to lead to a more desirable grid search. All results returned to the user are in the original parameterization.

HAZST

Performs hazard rate estimation over a grid of points using a kernel function.

Required Arguments

ST — Vector of length NOBS containing the event times, sorted in ascending order. (Input)
ST may not contain missing values.

JCEN — Vector of length NOBS containing the censor codes. (Input)
JCEN(i) = 1 means that event i was (right) censored at time ST(i), i = 1, …, NOBS.
JCEN(i) = 0 means that event i was a failure at time ST(i).

NGRID — Number of grid points at which to compute the hazard. (Input)

GSTRT — First grid value. (Input)

GINC — Increment between grid values. (Input)

ALPHA — Value for parameter α. (Input)

BTA — Value for parameter β. (Input)

K — Value for parameter k. (Input)

H — Vector of length 5 * NOBS containing the constants used in computing the k-th failure distance. (Input, if IHCOMP = 1; Output, otherwise)

HAZ — Vector of length NGRID containing the estimated hazard rates. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
If HAZRD (page 1188) or HAZEZ (page 1195) is called prior to this routine and the original data contained missing values, then NOBS in HAZST must be adjusted for the number of missing values from the value used in HAZRD or HAZEZ. That is, NOBS in HAZST is NOBS minus NMISS from HAZRD or HAZEZ .
Default: NOBS = size (ST,1).

IWTO — Weighting option. (Input)
IWTO = 1 means use weights ln(1 + 1/(NOBS – i + 1)). IWTO = 0 means use weights 1/(NOBS – i + 1). Not used if IHCOMP = 1.
Default: IWTO = 0.

IHCOMP — Option parameter. (Input)
If IHCOMP = 0, H is computed. If IHCOMP = 1, H has already been computed (generally

by HAZRD or HAZEZ).
Default: IHCOMP = 0.

FORTRAN 90 Interface

Generic: CALL HAZST (ST, JCEN, NGRID, GSTRT, GINC, ALPHA, BTA, K, H, HAZ [,…])

Specific: The specific interface names are S_HAZST and D_HAZST.

FORTRAN 77 Interface

Single: CALL HAZST (NOBS, ST, JCEN, IWTO, NGRID, GSTRT, GINC, ALPHA, BTA, K, IHCOMP, H, HAZ)

Double: The double precision name is DHAZST.

Example

The following example is a continuation of the example from HAZRD. The data are from Stablein, Carter, and Novak (1981), and involve the survival times of individuals with nonresectable gastric carcinoma. Only those individuals treated with both radiation and chemotherapy are used.

```
      USE HAZST_INT
      USE WRRRN_INT
      INTEGER    K, NGRID, NOBS
      REAL       ALPHA, BTA, GINC, GSTRT
      PARAMETER  (ALPHA=1.540537, BTA=1.631553, GINC=10, GSTRT=0.0, &
                  K=20, NGRID=100, NOBS=45)
!
      INTEGER    JCEN(NOBS), NOUT
      REAL       H(5*NOBS), HAZ(NGRID), ST(NOBS)
!
      DATA ST/17, 42, 44, 48, 60, 72, 74, 95, 103, 108, 122, 144, 167, &
          170, 183, 185, 193, 195, 197, 208, 234, 235, 254, 307, 315, &
          401, 445, 464, 484, 528, 542, 567, 577, 580, 795, 855, 882, &
          892, 1031, 1033, 1306, 1335, 1366, 1452, 1472/
      DATA JCEN/36*0, 9*1/
!
      CALL HAZST (ST, JCEN, NGRID, GSTRT, GINC, ALPHA, &
                  BTA, K, H, HAZ)
!
      CALL WRRRN ('Ten elements of HAZ', HAZ, 1, 10, 1)
      CALL WRRRN ('The first 10 rows of H', H, 10, 5, NOBS)
      END
```

Output

```
                        Ten elements of HAZ
       1          2          3          4          5          6          7
0.000962   0.001111   0.001276   0.001451   0.001634   0.001819   0.002004
```

```
       8          9         10
0.002185   0.002359   0.002523

          The first 10 rows of H
          1        2        3        4        5
 1   217.0    218.0      1.0     21.0      1.0
 2   192.0    193.0      1.0     21.0      0.5
 3   190.0    191.0      1.0     21.0      0.3
 4   186.0    187.0      1.0     21.0      0.2
 5   174.0    175.0      1.0     21.0      0.2
 6   162.0    163.0      1.0     21.0      0.2
 7   160.0    161.0      1.0     21.0      0.1
 8   139.0    140.0      1.0     21.0      0.1
 9   131.0    132.0      1.0     21.0      0.1
10   126.0    127.0      1.0     21.0      0.1
```

Comments

1. Informational error

Type	Code	
4	13	At least one missing (NaN, not a number) value was found in ST. Missing values are not allowed in this routine.

2. The user-defined grid is given by GSTRT + j * GINC, j = 0, …, NGRID − 1.

3. Routine HAZST assumes that the grid points are new data points.

Description

Routine HAZST estimates the hazard function by use of the biweight kernel,

$$K(x) = \frac{15}{16}(1-x^2)^2$$

Because a "smoothed" estimate is computed, one generally would use either routine HAZRD (page 1188) or routine HAZEZ (page 1195) to obtain maximum (modified) likelihood estimates of the smoothing parameters α, β, and k. Maximum (modified) likelihood estimates of these parameters are not required, however. A user-specified grid of points is generated. For each point, the hazard estimate is computed as

$$h(x) = \sum_{i=1}^{n} (1-\delta_i) w_i K_s(x - x_{(i)})$$

where n = NOBS, δ_i is the i-th observation's censoring code (0 = failed, 1 = censored), w_i is the i-th observation's weight (either $1/(n-i+1)$ or $\ln(1 + 1/(n-i+1))$ depending upon IWTO), and $K_s(x - x_{(i)})$, the "smoothed kernel", is as follows:

$$K_s(x - x_{(i)}) = \frac{1}{\alpha d_{ik}} K\left(\frac{x - x_{(i)}}{\beta d_{ik}}\right)$$

Here, d_{ik} is the distance to the k-th nearest failure from the i-th observation. Because of the d_{ik}, HAZST requires the computation of matrix H, which contains constants needed to quickly compute d_{ik}. Often, H will have been computed in routine HAZRD or HAZEZ. In this case, the parameter IHCOMP should be set to zero and H should be input to HAZST. If H must be computed by HAZST, set IHCOMP = 1.

Chapter 16: Line Printer Graphics

Routines

Usage Notes

The routine names in this chapter end with the letter "P" to indicate line printer plotting and every routine starts printing at the beginning of a new page.

Depending on the nature of plots, some routines allow the user to change page width and/or length. This capability is specified in each routine and, if allowed, can be done by calling the routine PGOPT (see Chapter 19, Utilities) in advance. To change the page width, the user should make the following call to PGOPT:

```
CALL PGOPT(-1, IPAGEW)
```

where IPAGEW indicates the page width in columns. To change the page length, the user should make the following call to PGOPT:

```
CALL PGOPT(-2, IPAGEL)
```

where IPAGEL indicates the page length in rows. See the PGOPT document for more information.

VHSTP

Prints a vertical histogram.

Required Arguments

FRQ — Vector of length NBAR containing the frequencies or counts. (Input)
Elements of FRQ must be nonnegative.

TITLE — CHARACTER string containing main title. (Input)

Optional Arguments

NBAR – Number of bars. (Input)
If NBAR exceeds 100/(ISP + 1), then NBAR = 100/(ISP + 1) is used. NBAR must be positive.
Default: NBAR = size (FRQ,1).

ISP — Spacing between histogram bars. (Input)
ISP may be 0, 1, or 4.
Default: ISP= 4.

FORTRAN 90 Interface

Generic: `CALL VHSTP (FRQ, TITLE [,…])`

Specific: The specific interface names are S_VHSTP and D_VHSTP.

FORTRAN 77 Interface

Single: `CALL VHSTP (NBAR, FRQ, ISP, TITLE)`

Double: The double precision name is DVHSTP.

Example

Consider the data set in Example 1 of the routine OWFRQ (see Chapter 1, Basic Statistics). This data set consists of the measurements (in inches) of precipitaion in Minneapolis/St. Paul during the month of March for 30 consecutive years. We use the routine OWFRQ to create a one-way frequency table. A vertical histogram is then generated using VHSTP. A horizontal histogram for the same data set can be found in the document example for the routine HHSTP (see Chapter 16, Line Printer Graphics).

```
USE UMACH_INT
USE OWFRQ_INT
USE VHSTP_INT
```

```
      INTEGER    NBAR, NOBS
      PARAMETER  (NBAR=10, NOBS=30)
!
      INTEGER    IOPT, NOUT
      REAL       DIV(NBAR), TABLE(NBAR), X(NOBS), XHI, XLO
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
!                                  Get output unit number
      CALL UMACH (2, NOUT)
!                                  Create a one-way frequency table from
!                                  a given data set using intervals of
!                                  equal length and user-supplied values
!                                  of XLO and XHI
      IOPT = 1
      XLO  = 0.5
      XHI  = 4.5
      CALL OWFRQ (X, NBAR, TABLE, IOPT=IOPT, XLO=XLO, XHI=XHI, DIV=DIV)
      WRITE (NOUT,99999) DIV, TABLE
99999 FORMAT (' Midpoints:  ', 10F6.2, /, '    Counts:  ', 10F6.0)
!                                  Create the horizontal histogram
      CALL VHSTP (TABLE, 'Plot of VHSTP')
      END
```

Output

```
 Midpoints:    0.25  0.75  1.25  1.75  2.25  2.75  3.25  3.75  4.25  4.75
    Counts:      2.    7.    6.    6.    4.    2.    2.    0.    0.    1.

                          Plot of VHSTP
Frequency-------------------------------------------------------------
   7  *          I                                                  *
   6  *          I    I    I                                        *
   5  *          I    I    I                                        *
   4  *          I    I    I    I                                   *
   3  *          I    I    I    I                                   *
   2  *     I    I    I    I    I    I    I                         *
   1  *     I    I    I    I    I    I    I              I    I     *
-----------------------------------------------------------------------
Class       1    2    3    4    5    6    7    8    9   10
```

Comments

1. Informational errors

Type	Code	
3	1	`ISP` is out of range. `ISP` = 0 is used.
3	3	`NBAR` * (`ISP` + 1) is less than 1 or greater than 100. The width of the histogram is set to 100, and 100/(`ISP` + 1) bars are printed. The number of class intervals will be printed completely if `ISP` ≠ 0 and will always be printed up to and including 100/(`ISP` + 1) even though the histogram body is only 100 spaces wide.

3	5	The maximum value in the vector FRQ is less than 1; therefore, the body of the histogram is blank.
3	6	TITLE is too long. TITLE was truncated from the right side.

2. Output is written to the unit specified by the routine UMACH (see the Reference Material section of this manual).

3. TITLE is centered and placed at the top of the plot. The plot starts on a new page.

Description

VHSTP prints a vertical histogram on not more than one printer page using not more than 50 vertical and 100 horizontal print positions. Spacing control is allowed on the horizontal axis.

Given a vector containing positive counts, VHSTP determines the maximum count T_{max}. Vertical printing position depends on K defined by $K = 1 + (T_{max} - 1)/50$: If a frequency is greater than K, then a character is printed on the corresponding position of the first horizontal line from above. Henceforth, K is reduced by $K/50$ for each horizontal line, and frequencies are compared to the new K.

VHS2P

Prints a vertical histogram with every bar subdivided into two parts.

```
CALL VHS2P (NBAR, FRQX, FRQY, ISP, TITLE)
```

Required Arguments

FRQX — Vector of length NBAR. (Input)
FRQX contains the frequencies or counts, and the elements of FRQX must be nonnegative.

FRQY — Vector of length NBAR. (Input)
FRQY contains the second frequencies or counts, and the elements of FRQY must be nonnegative.

TITLE — CHARACTER string containing the title. (Input)

Optional Arguments

NBAR — Number of bars. (Input)
NBAR must be positive.
Default: NBAR = size (FRQX,1).

ISP —Spacing between histogram bars. (Input)
ISP = 0, 1 or 4 is allowed.
Default: ISP = 4.

FORTRAN 90 Interface

Generic: `CALL VHS2P (FRQX, FRQY, TITLE [,…])`

Specific: The specific interface names are `S_VHS2P` and `D_VHS2P`.

FORTRAN 77 Interface

Single: `CALL VHS2P (FRQX, FRQY, TITLE, NBAR, ISP)`

Double: The double precision name is `DVHS2P`.

Example

Let X = `FRQX` contain 12 months of projected income figures and let Y = `FRQY` contain the actual income figures for the same 12 months. `VHS2P` produces a histogram that allows projected versus actual figures to be graphically compared.

```
      USE VHS2P_INT
      USE UMACH_INT
      INTEGER    NBAR
      PARAMETER  (NBAR=12)
!
      INTEGER    ISP, NOUT
      REAL       FRQX(NBAR), FRQY(NBAR)
!
      DATA FRQX/11., 4., 4., 8., 4., 3., 10., 14., 4., 20., 4., 3./
      DATA FRQY/10., 6., 4., 12., 3., 4., 8., 18., 6., 18., 3., 7./
!
      CALL VHS2P (FRQX, FRQY, 'Plot of VHS2P')
!                                  Get output unit number
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999)
99999 FORMAT (/, 3X, 'Twelve months projected sales versus actual ', &
             'sales, in thousands of dollars.', /, 11X, 'A positive ', &
             'sign (+) implies projected exceeded actual.', /, 11X, &
             'A negative sign (-) implies actual exceeded projected.')
!
      END
```

Output

```
                              Plot of VHS2P
Frequency-------------------------------------------------------------
  20                                                  -
  19                                                  -
  18                                          +       I
  17                                          +       I
  16                                          +       I
  15                                          +       I
  14                                          I       I
  13                                          I       I
  12                      +                   I       I
```

```
  11       -               +                   I          I
  10       I               +               -   I          I
   9       I               +               -   I          I
   8       I               I               I   I          I
   7       I               I               I   I          I         +
   6       I    +          I               I   I    +     I         +
   5       I    +          I               I   I    +     I         +
   4       I    I    I     I    -    +     I   I    I     I    -    +
   3       I    I    I     I    I    I     I   I    I     I    I    I
   2       I    I    I     I    I    I     I   I    I     I    I    I
   1       I    I    I     I    I    I     I   I    I     I    I    I
-------------------------------------------------------------------------
Class      1    2    3     4    5    6     7   8    9    10   11   12

  Twelve months projected sales versus actual sales, in thousands of dollars.
           A positive sign (+) implies projected exceeded actual.
           A negative sign (-) implies actual exceeded projected.
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `V2S2P/DV2S2P`. The reference is:

   ```
   CALL V2S2P (NBAR, FRQX, FRQY, ISP, TITLE, WK)
   ```

   The additional argument is

   WK — Work vector of length 2 * `NBAR`.

2. Informational errors

Type	Code	
3	2	`NBAR` * (`ISP` + 1) is less than 1 or greater than 100. The width of the histogram is set to 100 and 100/(`ISP` + 1) bars are printed.
3	3	`ISP` as specified is not valid. The zero option is used.
3	4	`TITLE` is too long. `TITLE` was truncated from the right side.

3. If `NBAR` exceeds 100/(`ISP` + 1), then only 100/(`ISP` + 1) bars are printed.

4. If the maximum frequency is greater than 9999, the frequency column contains on some lines.

5. Output is written to the unit specified by the routine `UMACH` (see the Reference Material secrion of this manual).

6. `TITLE` is automatically centered and plot starts on a new page.

Description

The routine `VHS2P` prints a vertical histogram on one or more pages, using not more than 50 vertical and 100 horizontal print positions. Spacing control is allowed on the horizontal axis. Given two vectors containing positive counts, `VHS2P` determines the maximum count of the

combined vectors T_{max}. Vertical printing position depends on K defined by $K = 1 + (T_{max} - 1)/50$. If a frequency is greater than K, then a character is printed on the first line. Henceforth, K is reduced by $K/50$ for each position, and frequencies are compared to the new K.

HHSTP

Prints a horizontal histogram.

Required Arguments

FRQ —Vector of length NBAR containing the frequencies or counts. (Input)
Elements of FRQ must be nonnegative.

IBEG — Indicates the beginning setting of the plot. (Input)
If IBEG = 0, HHSTP skips to a new page before printing the first line. If IBEG ≠ 0, HHSTP skips two spaces and begins printing on the same page.

TITLE — CHARACTER string containing the title of the histogram. (Input)

Optional Arguments

NBAR — Number of bars. (Input)
NBAR must be positive.
Default: NBAR = size (FRQ,1).

ISPACE — Indicates spaces between horizontal histogram lines. (Input)
ISPACE = 0, 1, or 2 is allowed.
Default: ISPACE = 1.

LENGTH — Indicates the upper limit of the number of lines to print within the histogram per page. (Input)
After that number of lines is printed, the routine skips to a new page to continue printing. If LENGTH = 0; then the maximum number of lines coincides with the standard printer page, which is 60.
Default: LENGTH = 0.

IREP — Determines the repeating appearance for the class line (top) and frequency line (bottom) when multiple pages are required. (Input)
If IREP = 0, the class line and the frequency line are printed on the first and last page of the histogram, respectively. If IREP ≠ 0, both class and frequency line are printed on every page.
Default: IREP = 0.

IOPT — Page width option. (Input)
IOPT = 0 will cause a full (horizontal) page histogram. IOPT = 1 will limit the width to 80 columns.
Default: IOPT = 1.

FORTRAN 90 Interface

Generic: `CALL HHSTP (FRQ, IBEG, TITLE [,…])`

Specific: The specific interface names are `S_HHSTP` and `D_HHSTP`.

FORTRAN 77 Interface

Single: `CALL HHSTP (NBAR, FRQ, IBEG, ISPACE, LENGTH, IREP, IOPT, TITLE)`

Double: The double precision name is `DHHSTP`.

Example

Consider the data set in Example 1 of the routine `OWFRQ` (see Chapter 1, Basic Statistics). We use the routine `OWFRQ` to create a one-way frequency table. A horizontal histogram is then generated using `HHSTP`. The user may find a vertical histogram for the same data set in the routine `VHSTP` (see Chapter 16, Line Printer Graphics). Note that classes are listed from left to right in `VHSTP`.

```
      USE UMACH_INT
      USE OWFRQ_INT
      USE HHSTP_INT
      INTEGER    NBAR, NOBS
      PARAMETER  (NBAR=10, NOBS=30)
!
      INTEGER    IBEG, IOPT, NOUT
      REAL       CLHW, DIV(NBAR), TABLE(NBAR), X(NOBS), XHI, XLO
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
!                                Get output unit number
      CALL UMACH (2, NOUT)
!                                Create a one-way frequency table from
!                                a given data set with intervals of
!                                equal length and user-supplied values
!                                of XLO and XHI
      IOPT = 1
      XLO  = 0.5
      XHI  = 4.5
      CALL OWFRQ (X, NBAR, TABLE, IOPT=IOPT, XLO=XLO, XHI=XHI, DIV=DIV)
      WRITE (NOUT,99999) DIV, TABLE
99999 FORMAT (’  Midpoints:  ’, 10F6.2, /, ’     Counts:  ’, 10F6.0)
!                                Create the horizontal histogram
      IBEG   = 1
      IOPT   = 0
      CALL HHSTP (TABLE, IBEG, ’Histogram’, IOPT=IOPT)
      END
```

Output

```
Midpoints:     .25   .75  1.25  1.75  2.25  2.75  3.25  3.75  4.25  4.75
   Counts:      2.    7.    6.    6.    4.    2.    2.    0.    0.    1.

    Histogram
Class -----------
  10  *I       *
      *        *
   9  *        *
      *        *
   8  *        *
      *        *
   7  *II      *
      *        *
   6  *II      *
      *        *
   5  *IIII    *
      *        *
   4  *IIIIII  *
      *        *
   3  *IIIIII  *
      *        *
   2  *IIIIIII *
      *        *
   1  *II      *
-----------------
Frequency  5
           One frequency unit is equal to 1 count unit(s).
```

Comments

Informational errors

Type	Code	
3	3	`ISPACE` is not 0, 1, or 2. The zero option is used for `ISPACE`.
3	6	`IOPT` is not 0 or 1. The zero option is used for `IOPT`.
3	7	`TITLE` is too long and is truncated from the right side.

Description

The routine `HHSTP` prints a horizontal histogram on one or more pages. Given a vector containing frequencies or counts, `HHSTP` determines the maximum count $T_{\max}$. Horizontal printing position depends on K defined by

$K = 1 + (T_{\max} - 1)/60$ for 72 characters

$K = 1 + (T_{\max} - 1)/120$ for 132 characters

If a frequency is greater than K, then a character is printed in the first position. Henceforth, K is increased by $K/60$ or $K/120$ for each position, and frequencies are compared to the resulting K.

SCTP

Prints a scatter plot of several groups of data.

Required Arguments

A —NOBS by NVAR matrix containing the data. (Input)

ICOL — Vector of length NVAR representing the nature of each column of matrix A. (Input)
The I-th column of A is the independent variable vector if ICOL(I) = 1. The I-th column of A is a dependent variable vector if ICOL(I) = 2. The I-th column of A is ignored otherwise.

RANGE — Vector of length four specifying minimum *x*, maximum *x*, minimum *y* and maximum *y*. (Input)
SCTP will calculate the range of the axis if the minimum of that range is greater than or equal to the maximum of that range.

SYMBOL — CHARACTER string of length NVAR. (Input)
SYMBOL (I : I) is the character used to plot the data set represented by column I. SYMBOL(I : I) is ignored if ICOL(I) ≠ 2.

XTITLE — CHARACTER string containing the *x*-axis title. (Input)

YTITLE — CHARACTER string containing the *y*-axis title. (Input)

TITLE — CHARACTER string containing the plot title. (Input)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (A,1).

NVAR — Number of variables. (Input)
Default: NVAR = size (A,2).

LDA —Leading dimension of A exactly as specified in the dimension statement of the calling program. (Input)
Default: LDA = size (A, 1).

FORTRAN 90 Interface

Generic: `CALL SCTP (A, ICOL, RANGE, SYMBOL, XTITLE, YTITLE, TITLE [,…])`

Specific: The specific interface names are S_SCTP and D_SCTP.

FORTRAN 77 Interface

Single: `CALL SCTP (NOBS, NVAR, A, LDA, ICOL, RANGE, SYMBOL, XTITLE, YTITLE, TITLE)`

Double: The double precision name is `DSCTP`.

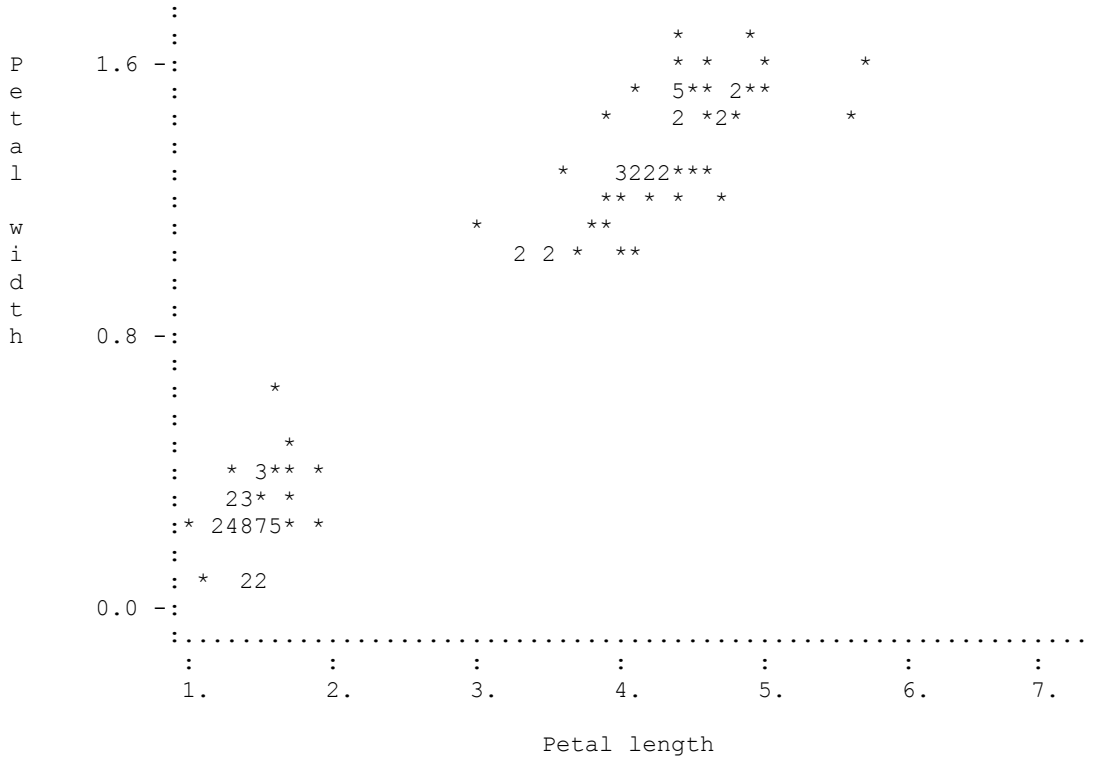
Example

This example prints a scatter plot of width against length for 150 iris petals. The routine `GDATA` (see Chapter 20, Mathematical Support) is used to retrieve the Fisher iris data.

```
      USE GDATA_INT
      USE SCTP_INT
      USE PGOPT_INT
      INTEGER    ICOL(5), IDATA, IPAGE, LDA, NDA, NOBS, NVAR
      REAL       A(150,5), RANGE(4)
      CHARACTER  SYMBOL*5
!
      DATA ICOL/5*0/
      DATA RANGE/4*0./
      DATA SYMBOL/'    *'/
!
      IDATA  = 3
!                                Get Fisher Iris Data
      CALL GDATA (IDATA, A, NOBS, NVAR)
!                                Plot petal width against
!                                petal length
      ICOL(4) = 1
      ICOL(5) = 2
!                                Set page width and length
      IPAGE = 78
      CALL PGOPT (-1, IPAGE)
      IPAGE = 40
      CALL PGOPT (-2, IPAGE)

      CALL SCTP (A, ICOL, RANGE, SYMBOL, 'Petal length', &
                 'Petal width', 'Fisher Iris Data')
!
      END
```

Output

```
                                Fisher Iris Data
          :
          :                                            *  **
      2.4 -:                                        *    2
          :                                         ****  * * *        *
          :                                              * *         *
          :
          :                                           **** *       *
          :                                       ****            *  *
          :                                        *2 *        *
          :                                      32 *   2* * *  *
```



Comments

1. Informational errors

Type	Code	
3	10	XTITLE is too long to fit into the page width determined by the routine PGOPT. XTITLE is truncated from the right side.
3	11	YTITLE is too long to fit into the page width determined by the routine PGOPT. YTITLE is truncated from the right side.
3	12	TITLE is too long to fit into the page width determined by the routine PGOPT. TITLE is truncated from the right side.

2. Integers 2, …, 9 indicate two through nine points occupying the same plot position, respectively, and the character "M" indicates 10 or more multiple points. Consequently, it is recommended not to use any one of the above characters for SYMBOL.

3. One and only one column of A can be the independent variable vector.

4. A point is ignored if either the independent or the dependent variable contains NaN (not a number).

5. Output is written to the unit number specified by the routine UMACH (see the Reference Material section in this manual).

6. Default page width and length are 78 and 60; respectively. The user may change them by calling the routine PGOPT (see Chapter 19, Utilities) in advance.

Description

Routine SCTP prints a scatter plot of one variable on the *x*-axis against several variables on the *y*-axis. For multiple points, 2, 3, …, 9 are used to denote the number of points at a location. The character "M" is used when the number of points is greater than 9. Any entry of the matrix A containing NaN (not a number) is ignored. See AMACH in "Machine-Dependent Constants.".

BOXP

Prints boxplots for one or more samples.

Required Arguments

NI — Vector of length NGROUP. (Input)
NI(I) is the number of observations in the I-th group.

X —Vector of length NI(1) + NI(2) + … + NI(NGROUP). (Input)
The first NI(1) positions contain the observations for the first group. The next NI(2) positions contain the observations for the second group, and so on.

TITLE — CHARACTER string containing the title of the plot. (Input)

Optional Arguments

NGROUP — The total number of groups of samples. (Input)
Default: NGROUP = size (NI,1).

FORTRAN 90 Interface

Generic: `CALL BOXP (NI, X, TITLE [,…])`

Specific: The specific interface names are S_BOXP and D_BOXP.

FORTRAN 77 Interface

Single: `CALL BOXP (NGROUP, NI, X, TITLE)`

Double: The double precision name is DBOXP.

Example

This example prints boxplots of three batches of data containing 5, 16 and 7 observations, respectively.

```
      USE PGOPT_INT
      USE BOXP_INT
      INTEGER    IPAGE, NGROUP
      PARAMETER  (NGROUP=3)
!
      INTEGER    NI(NGROUP)
      REAL       X(28)
!
      DATA (NI(I),I=1,3)/5, 16, 7/
      DATA (X(I),I=1,5)/7., 9., 3., 1., 1./
      DATA (X(I),I=6,21)/25., 0., 1., 0., 5., 4., 3., 5., 5., 5., 5., &
          5., 5., 25., 15., 9./
      DATA (X(I),I=22,28)/10., 15., 20., 25., 2., 9., 12./
!                                Set page width.
      IPAGE = 70
      CALL PGOPT (-1, IPAGE)
      CALL BOXP (NI, X, 'Plot of BOXP')
!
      END
```

Output

```
                          Plot of BOXP

  X    X          X    X
  2

         I--------I
+--------I   *    I-----+              O                          X
         I--------I                                              2

                       I--------------------I
     +-----------------I     *              I-------------------+
                       I--------------------I

+..............................+..............................+
0.0                           12.5                           25.0
```

Comments

1. Workspace may be explicitly provided, if desired, by use of B2XP/DB2XP. The reference is:

   ```
   CALL B2XP (NGROUP, NI, X, TITLE, WKSP)
   ```

   The additional argument is:

   WKSP — Workspace of length NI(1) + … + NI(NGROUP). (Input)
   The first NI(1) positions contain the sorted data from the first NI(1) positions of X. The next NI(2) positions contain sorted data from the next NI(2) positions of X, and so on.

2. Informational error

Type	Code	
3	5	TITLE is too long to fit into the page width determined by the routine PGOPT. TITLE is truncated from the right side.

3. TITLE is centered and placed at the top of the plot. The plot starts on a new page and the default page width is 78. The user may change the width by calling the routine PGOPT (see Chapter 19, Utilities) in advance.

Description

BOXP prints NGROUP boxplots. The minimum and maximum of X are printed. The median of each data group is marked by "*" and the upper and lower hinges by "I". The "H-spread" is the distance between the upper and lower hinges. The observation farthest from the median that still remains within one step (1.5 H-spread) from each hinge also is marked by "+". The values in the second step (between 1.5 and 3 H-spreads from the hinges) are marked by the letter "O" and the values beyond the second step are marked by "X". If there are fewer than five data points, each data point is plotted with an "X." If multiple data points occur at positions marked "X" or "O", the number of multiple points is noted. More information on boxplots can be found in Chapter 2 of Chambers et al. (1983).

STMLP

Prints a stem-and-leaf plot.

Required Arguments

X —Array of length NOBS containing the data. (Input)

UNITS — Size of the increment on the stem. (Input)
If UNITS is set so small that the length of the stem is more than 60 lines, STMLP will use a UNITS such that the stem will be no longer than 60 lines. However, if UNITS is a negative integer, STMLP will use the absolute value of UNITS, even if the stem would become very long. A common value for UNITS is 10.

TITLE — CHARACTER string containing the plot title. (Input)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOB = size (X,1).

FORTRAN 90 Interface

Generic: `CALL STMLP (X, UNITS, TITLE [,…])`

Specific: The specific interface names are S_STMLP and D_STMLP.

FORTRAN 77 Interface

Single: `CALL STMLP (NOBS, X, UNITS, TITLE)`

Double: The double precision name is `DSTMLP`.

Example

This example prints a stem-and-leaf plot consisting of 27 data points ranging from –21.8 to 106.5.

```
      USE STMLP_INT
      INTEGER    NOBS
      PARAMETER  (NOBS=27)
!
      REAL       UNITS, X(NOBS)
!
      DATA X/6.0, 106.5, 34.0, 88.1, 89.0, 0.3, 0.7, 4.0, 4.0, 5.0, &
          56.0, 62.8, 99.0, 4.0, 15.0, 76.0, 7.6, 101.5, 33.0, 91.0, &
          91.0, -6.3, -21.8, 0.0, 8.99, 5.5, 6.9/
!
      UNITS = 10.
      CALL STMLP (X, UNITS, 'Stem and leaf plot')
!
      END
```

Output

```
                          Stem and leaf plot

Each line on the stem represents 1.0  unit(s).
For example:  1 25
              2 2
represents the data 12., 15., and 22.
      -2.2
      -1
      -0 6
       0 001444566789
       1 5
       2
       3 34
       4
       5 6
       6 3
       7 6
       8 89
       9 119
      10 27
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `S2MLP/DS2MLP`. The reference is:

```
CALL S2MLP (NOBS, X, UNITS, TITLE, MAXWID, IWK, WK)
```

The additional arguments are as follows:

MAXWID — Page width. (Input)
MAXWID = 78 when STMLP is called.

IWK — Work vector of length MAXWID.

WK — Vector of length NOBS. (Output)
WK contains the sorted data from X.

2. Informational error

Type	Code	
3	4	TITLE is too long to fit into the page width determined by the routine PGOPT. TITLE is truncated from the right side.

3. Default page width is 78. The user may change it by calling the routine PGOPT (see Chapter 19, Utilities) in advance.

Description

Routine STMLP prints a stem-and-leaf display. The user can specify that the plot be longer than one page, but the default maximum is 60 lines. A plus sign (+) at the end of a line indicates that there are too many data points to fit within the width specifications. A scale marked in units of 10 is printed below the stemand-leaf display.

CDFP

Prints a sample cumulative distribution function (CDF), a theoretical CDF, and confidence band information.

Required Arguments

CDF — User-supplied FUNCTION to compute the cumulative distribution function. The form is CDF(P), where
P — Sample point. (Input)
CDF — Theoretical probability at the point P or integral of the probability density function at the point P. (Output)

X —Vector of length NOBS containing the sample. (Input)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

N12 —Confidence band option. (Input)
If N12 = 0, then no confidence bands are printed. If N12 = 1, then positive or upper one-sided confidence band information is printed. If N12 = –1, then negative or lower one-sided confidence band information is printed. If N12 = 2, then two-sided confidence band information is printed.
Default: N12 = 2.

N95 —Confidence band option. (Input)
If N95 = 95, the 95-percent band is desired. Otherwise, the 99-percent band is desired.
Default: N95 = 95.

IPRINT — Print option. (Input)
If IPRINT = 1, then CDFP prints the sample CDF, the theoretical CDF, and the confidence band on the CDF. If IPRINT = 0, then the above information will not be printed.
Default: IPRINT = 1.

FORTRAN 90 Interface

Generic: `CALL CDFP (CDF, X [,…])`

Specific: The specific interface names are S_CDFP and D_CDFP.

FORTRAN 77 Interface

Single: `CALL CDFP (CDF, NOBS, X, N12, N95, IPRINT)`

Double: The double precision name is DCDFP.

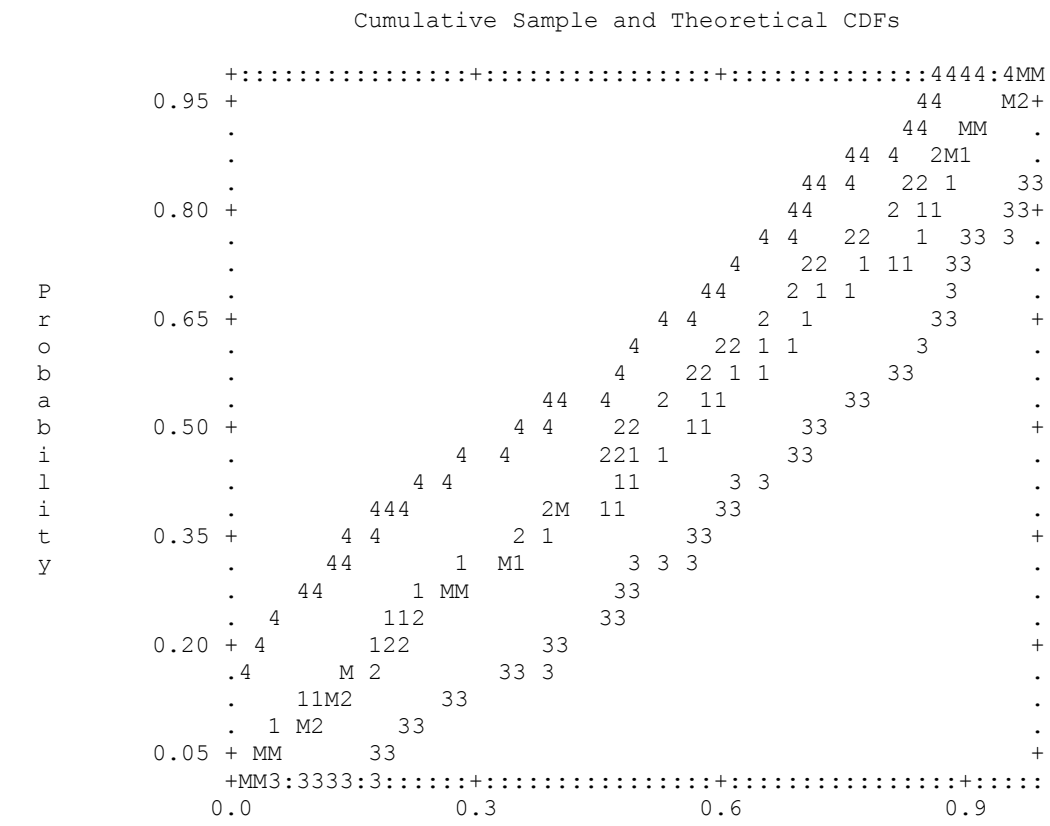
Example

This example prints and plots the sample CDF, the theoretical CDF, and the two-sided 95 percent band information using 70 observations. Routines RNSET (page 1166) and RNUN (page 1171) are called to generate these uniform (0, 1) random numbers.

```
      USE PGOPT_INT
      USE RNSET_INT
      USE RNUN_INT
      USE CDFP_INT
      INTEGER    IPAGE, ISEED, NOBS
      PARAMETER  (NOBS=70)
      REAL       CDF, X(NOBS)
      EXTERNAL   CDF
!
      ISEED = 123457
!                                Two-sided confidence band option.
!                                95-percent band option.
!                                Set page width and length.
      IPAGE =  78
```

```
      CALL PGOPT (-1, IPAGE)
      IPAGE=40
      CALL PGOPT (-2, IPAGE)
!                                Initialize the seed.
      CALL RNSET (ISEED)
!                                Generate pseudo-random numbers from
!                                a uniform (0,1) distribution.
      CALL RNUN (X)
!                                Plot
      CALL CDFP (CDF, X, IPRINT=0)
      END
!
      REAL FUNCTION CDF (X)
      REAL       X
!
      CDF = X
      RETURN
      END
```

Output



Comments

1. Workspace may be explicitly provided, if desired, by use of C2FP/DC2FP. The reference is:

   ```
   CALL C2FP (CDF, NOBS, X, N12, N95, IPRINT, WKX, WK)
   ```

   The additional arguments are as follows:

   WKX — Vector of length NOBS containing the sorted data X in ascending order. (Output)

   WK — Vector of length 4 * NOBS containing confidence band values. (Output)
   WK may be dimensioned 3 * NOBS instead of 4 * NOBS for a lower or upper confidence band.

2. Note that sample CDFs are step functions.

3. Confidence bands are plotted around the sample CDF.

4. Output is written to the unit specified by the routine UMACH (see the Reference Material section in this manual).

5. Printing starts on a new page with default page width 78 columns and default page length 60 rows. The user may change these values by calling the routine PGOPT (page 1263) in advance.

Description

When IPRINT = 1, CDFP prints the sample cumulative distribution function (CDF), the theoretical CDF, and confidence bands on the CDF. The theoretical CDF will be plotted with or without the confidence band information. The sample CDF is calculated. The theoretical CDF is calculated by calling the user supplied FUNCTION subprogram CDF. Asymptotic critical values are used (from the Smirnov tables) for confidence interval calculations.

CDF2P

Prints a plot of two sample cumulative distribution functions.

Required Arguments

NOBS1 — Size of sample one. (Input)

NOBS2 — Size of sample two. (Input)

X — Vector of length NOBS1 + NOBS2. (Input)
X contains sample one followed by sample two.

FORTRAN 90 Interface

Generic: `CALL CDF2P (NOBS1, NOBS2, X)`

Specific: The specific interface names are `S_CDF2P` and `D_CDF2P`.

FORTRAN 77 Interface

Single: `CALL CDF2P (NOBS1, NOBS2, X)`

Double: The double precision name is `DCDF2P`.

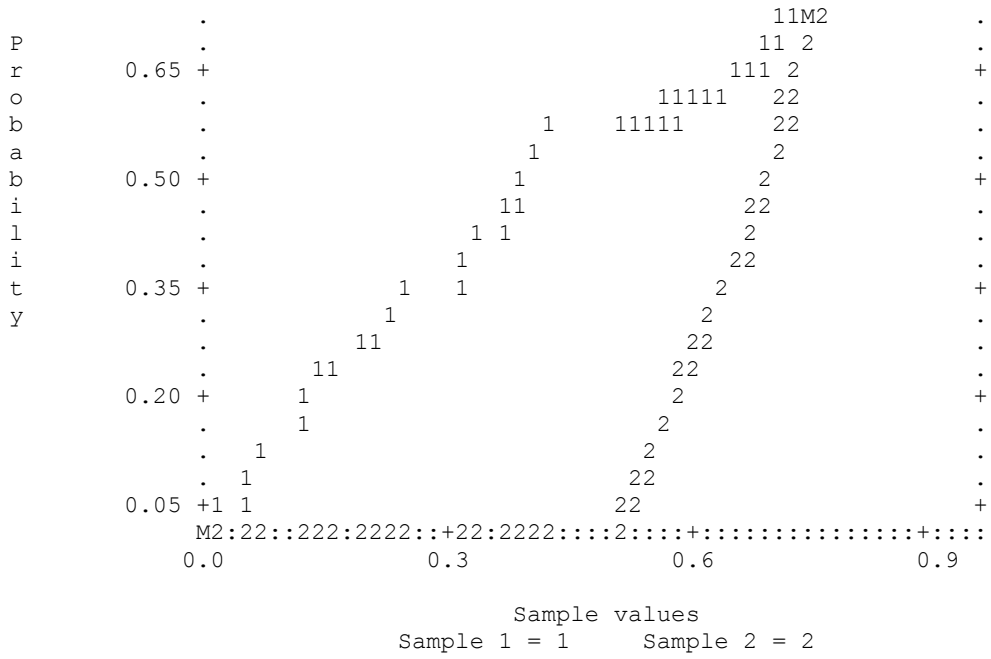
Example

The first sample consists of pseudo-random numbers from a uniform (0, 1) distribution. Routines `RNSET` and `RNUN` (see Chapter 18, Random Number Generation) are used to generate this sample. The second sample consists of points of the standard normal (Gaussian) distribution function generated by the routine `ANORDF` (see Chapter 17, Probability Distribution Functions and Inverses).

```
      USE IMSL_LIBRARIES
      INTEGER    I, IPAGE
      REAL       VAL, X(100)
!                                Initialize the seed.
      CALL RNSET (1234567)
!                                Generate pseudo-random numbers from
!                                a uniform (0,1) distribution.
      CALL RNUN (X)
!                                Second sample consists of 50 points of
!                                the std normal distribution function.
      VAL = 0.
      DO 10  I=1, 50
         VAL     = VAL + .02
         X(I+50) = ANORDF(VAL)
   10 CONTINUE
!                                Set page width and length.
      IPAGE =  78
      CALL PGOPT (-1, IPAGE)
      IPAGE =  40
      CALL PGOPT (-2, IPAGE)
      CALL CDF2P (50, 50, X)
      END
```

Output

```
                  Cumulative Sample Distribution Functions
          +:::::::::::::::::+:::::::::::::::::+:::::::::::::22:2+22:M
     0.95 +                                                  2     1 +
          .                                                 22  1 1  .
          .                                                 2 1 1    .
          .                                                2111      .
     0.80 +                                                1         +
          .                                               M2         .
```



Comments

1. Workspace may be explicitly provided, if desired, by use of C2F2P/DC2F2P. The reference is:

   ```
   CALL C2F2P (NOBS1, NOBS2, X, WK, IWK)
   ```

   The additional arguments are as follows:

   WK — Work vector of length 3 * (NOBS1 + NOBS2).

   IWK — Work vector of length NOBS1 + NOBS2.

2. Printing starts on a new page with default page width 78 and default page length 60. The user may change page width and length by calling the routine PGOPT (page 1263) in advance.

Description

Routine CDF2P plots two sample cumulative probability distribution functions (CDFs). Two samples are first merged and then sorted. The cumulative distribution functions are then calculated. On the plots, the characters "1" and "2" indicate the first and second samples, respectively, and the character "M" indicates multiple points.

PROBP

Prints a probability plot.

Required Arguments

NOBS — Total number of observations in uncensored sample. (Input)

N1 — The rank number of the smallest observation in the sample X, if ranked in the complete sample. (Input)
In other words, the number of observations that have been censored from below is N1 − 1.

N2 — The rank number of the largest observation in the sample X, if ranked in the complete sample. (Input)
In other words, the number of observations that have been censored from above is NOBS − N2.

X — {Vector of length N2 − N1 + 1. (Input)
X contains the data, possibly a censored data set from a complete sample of size NOBS.

IDIST — Distribution option. (Input)

IDIST = 1, normal distribution.

IDIST = 2, lognormal distribution.

IDIST = 3, half-normal distribution.

IDIST = 4, exponential distribution.

IDIST = 5, Weibull distribution.

IDIST = 6, extreme value distribution.

FORTRAN 90 Interface

Generic: `CALL PROBP (NOBS, N1, N2, X, IDIST)`

Specific: The specific interface names are `S_PROBP` and `D_PROBP`.

FORTRAN 77 Interface

Single: `CALL PROBP (NOBS, N1, N2, X, IDIST)`

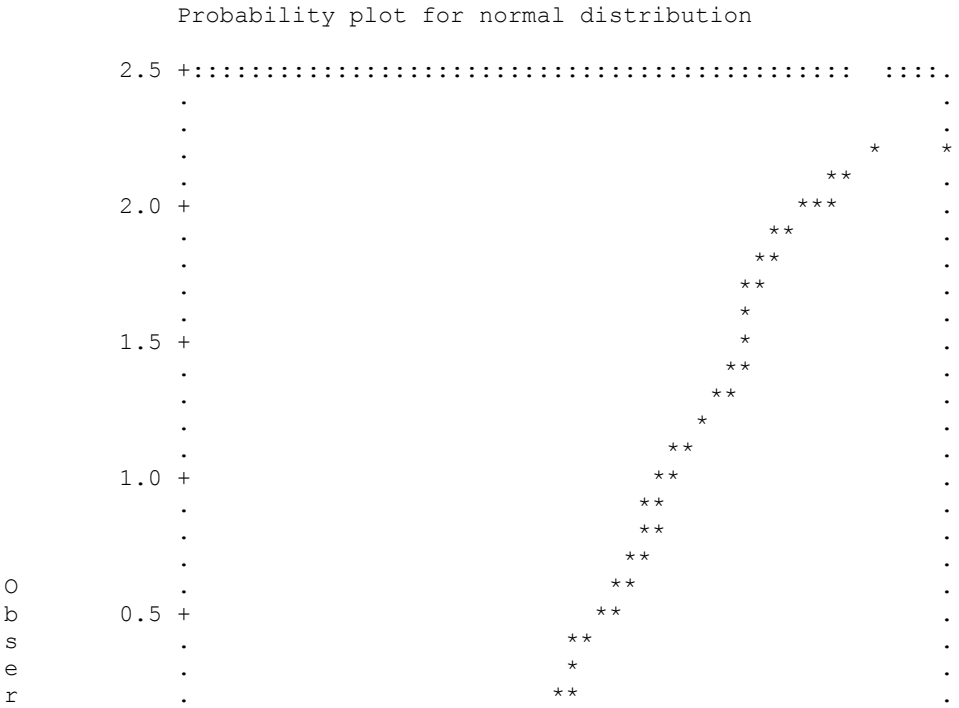
Double: The double precision name is `DPROBP`.

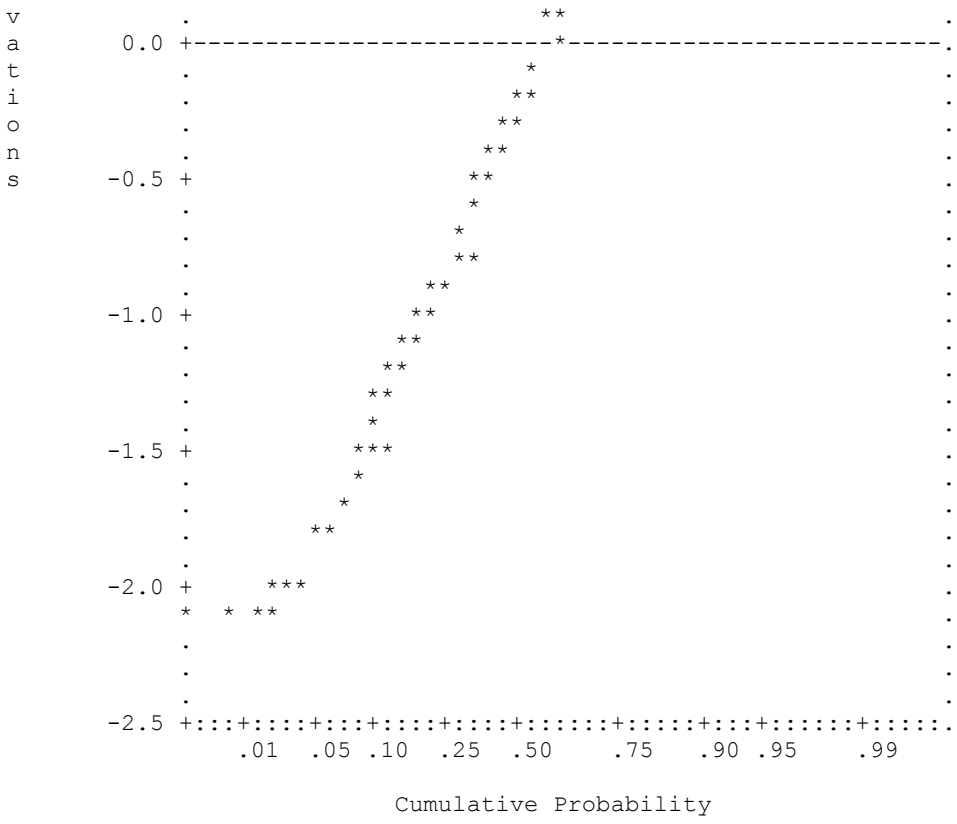
Example

In this example, a sample of size 250 (artificially generated from a normal distribution by routines RNSET and RNNOR, in Chapter 18, Random Number Generation) is plotted by PROBP against a normal distribution function. The generally straight line produced is an indication that the sample is from a normal distribution.

```
      USE RNSET_INT
      USE RNNOR_INT
      USE PROBP_INT
      INTEGER    NOBS
      PARAMETER  (NOBS=250)
!
      INTEGER    IDIST, N1, N2
      REAL       X(NOBS)
!
      IDIST = 1
!                                No censoring
      N1 = 1
      N2 = 250
!                                Initialize the seed
      CALL RNSET (123457)
      CALL RNNOR (X)
!
      CALL PROBP (NOBS, N1, N2, X, IDIST)
      END
```

Output





Comments

1. Workspace may be explicitly provided, if desired, by use of P2OBP/DP2OBP. The reference is:

   ```
   CALL P2OBP (NOBS, N1, N2, X, IDIST, M1, M2, WK)
   ```

   The additional arguments are as follows:

   M1 — Rank of the smallest observation actually used. (Output)

   M2 — Rank of the largest observation actually used. (output)

   WK — Work space of length 2 * NOBS.

2. Informational error

Type	Code	
3	7	It is necessary to delete some items from the plotting because those items do not satisfy properties of the distribution.

3. NOBS must be greater than or equal to N2 – N1 + 1. If there is no censoring, then N1 = 1 and N2 = NOBS.

4. Output is written to the unit specified by the routine UMACH (see the Reference Material section in this manual).

5. Printing starts on a new page with default page width 78. The user may change it by calling the routine PGOPT (see Chapter 19, Utilities) in advance.

Description

Routine PROBP sorts a data set and plots the observed values along the vertical axis and the ranks along the horizontal axis. In the case of the lognormal and Weibull distributions, the vertical axis has a log scale. The horizontal axis has the appropriate cumulative distribution function scale. Let M = NOBS denote the total number of observations in an uncensored sample. For normal and lognormal distributions, the horizontal plotting distance for the observation with rank I (out of M) is proportional to the inverse normal cumulative distribution function evaluated at $(3 * I - 1)/(3 * M + 1)$. For the half-normal plot, the corresponding horizontal distance is proportional to the inverse normal cumulative distribution function evaluated at $(3 * M + 3 * I - 1)/(6 * M + 1)$. For other plots, the horizontal distances are proportional to the respective inverse cumulative distribution functions evaluated at $(I - .5)/M$.

Let N_1 = N1 and N_2 = N2. In PROBP it is assumed that the $N_1 - 1$ smallest observations and the $M - N_2$ largest observations have been censored. If there has been no censoring, N_1 should be set to 1 and N_2 set to M. The smallest observation is plotted against the expected value (or the approximated expected value) of the N_1-th order statistic from a sample of size M; the next smallest observation is plotted as if it were the $(N_1 + 1)$-th sample order statistic, and so on.

PROBP does not do any shifting of location of the observation in the data set. If any observations fall outside of the range of the distribution (that is, if any observations are nonpositive when the distribution specified is lognormal or Weibull), those observations are censored and N_1 or N_2 is modified to reflect the number censored. In this case an error message of type 3 is generated. A plot which is a straight line provides evidence that the sample is from the distribution specified.

PLOTP

Prints a plot of up to 10 sets of points.

Required Arguments

X — Vector of length NDATA containing the values of the independent variable. (Input)

A — Matrix of dimension NDATA by NFUN containing the NFUN sets of dependent variable values. (Input)

SYMBOL — CHARACTER string of length NFUN. (Input)
SYMBOL (I : I) is the symbol used to plot function I.

XTITLE — CHARACTER string used to label the x-axis. (Input)

YTITLE — CHARACTER string used to label the *y*-axis. (Input)

TITLE — CHARACTER string used to label the plot. (Input)

Optional Arguments

NDATA — Number of independent variable data points. (Input)
Default: NDATA = size (X,1).

NFUN — Number of sets of points. (Input)
NFUN must be less than or equal to 10.
Default: NFUN = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement of the calling program. (Input)
Default: LDA = size (A,1).

INC — Increment between elements of the data to be used. (Input)
PLOTP plots X(1 + (I – 1) * INC) for I = 1, 2, …, NDATA.
Default: INC = 1.

RANGE — Vector of length four specifying minimum *x*, maximum *x*, minimum *y* and maximum *y*. (Input)
PLOTP will calculate the range of the axis if the minimum and maximum of that range are equal.
Default: RANGE = 1.0.

FORTRAN 90 Interface

Generic: `CALL PLOTP (X, A, SYMBOL, XTITLE, YTITLE, TITLE [,…])`

Specific: The specific interface names are S_PLOTP and D_PLOTP.

FORTRAN 77 Interface

Single:
```
CALL PLOTP (NDATA, NFUN, X, A, LDA, INC, RANGE, SYMBOL,
XTITLE, YTITLE, TITLE)
```

Double: The double precision name is DPLOTP.

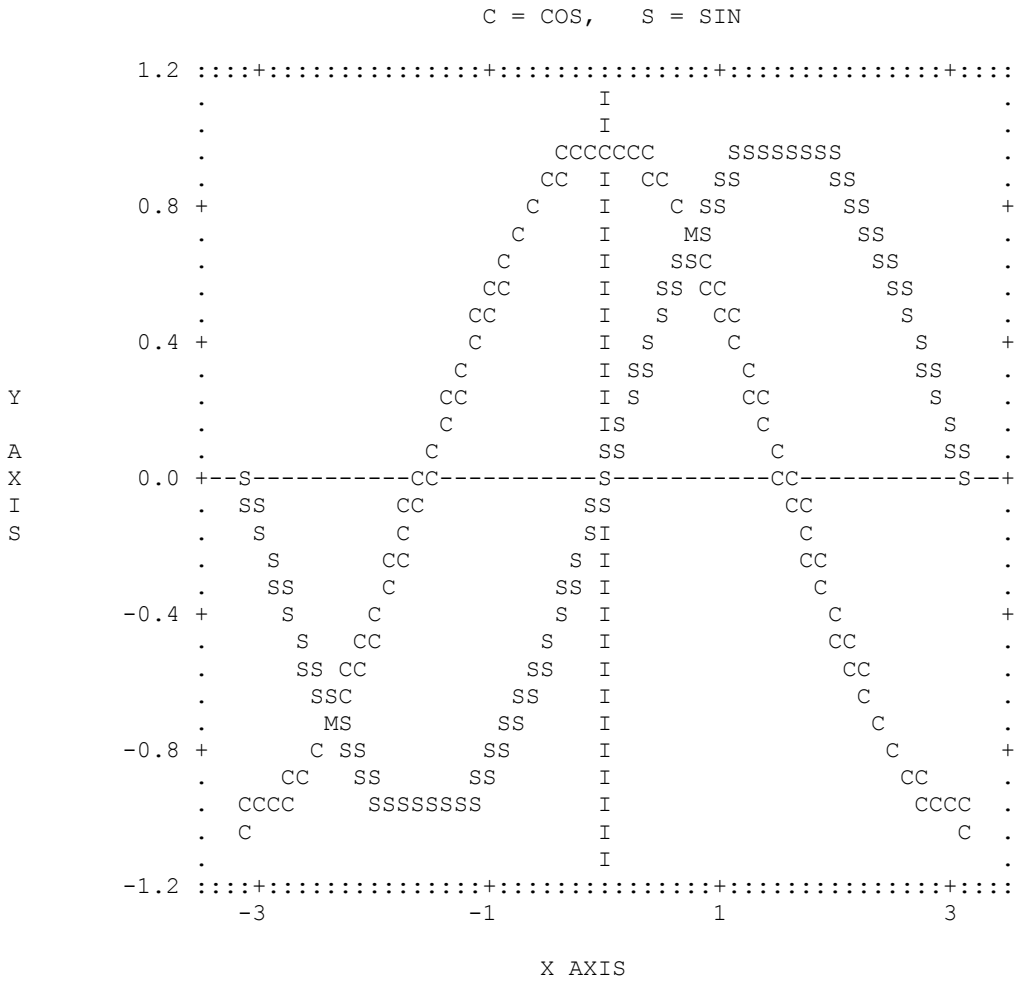
Example

This example plots the sine and cosine functions from – 3.5 to + 3.5 and sets page width and length to 78 and 40, respectively, by calling PGOPT (see Chapter 19, Utilities) in advance.

```
USE PGOPT_INT
USE PLOTP_INT
INTEGER    I, IPAGE
```

```
      REAL       A(200,2), DELX, PI, RANGE(4), X(200)
      CHARACTER  SYMBOL*2
      INTRINSIC  COS, SIN
!
      DATA SYMBOL/'SC'/
      DATA RANGE/-3.5, 3.5, -1.2, 1.2/
!
      PI     = 3.14159
      DELX   = 2.*PI/199.
      DO 10  I= 1, 200
         X(I)   = -PI + FLOAT(I-1) * DELX
         A(I,1) = SIN(X(I))
         A(I,2) = COS(X(I))
   10 CONTINUE
!                                 Set page width and length
      IPAGE =  78
      CALL PGOPT (-1, IPAGE)
      IPAGE =  40
      CALL PGOPT (-2, IPAGE)
      CALL PLOTP (X, A, SYMBOL,'X AXIS', 'Y AXIS', 'C = COS, S = SIN')
!
      END
```

Output



Comments

1. Informational errors

Type	Code	
3	7	NFUN is greater than 10. Only the first 10 functions are plotted.
3	8	TITLE is too long. TITLE is truncated from the right side.
3	9	YTITLE is too long. YTITLE is truncated from the right side.
3	10	XTITLE is too long. XTITLE is truncated from the right side. The maximum number of characters allowed depends on the page width and the page length. See Comment 5 below for more information.

2. YTITLE and TITLE are automatically centered.

3. For multiple plots, the character M is used if the same print position is shared by two or more data sets.

4 Output is written to the unit specified by UMACH (see the Reference Material section in this manual).

5. Default page width is 78 and default page length is 60. They may be changed by calling PGOPT in advance.

Description

Routine PLOTP produces a line printer plot of up to ten sets of points superimposed upon the same plot. A character "M" is printed to indicate multiple points. The user may specify the *x* and *y*-axis plot ranges and plotting symbols. Plot width and length may be reset in advance by calling PGOPT.

TREEP

Prints a binary tree.

Required Arguments

ICLSON — Vector of length NODE − 1 containing the left son nodes. (Input)
Node number NODE + K has left son given by ICLSON(K) for K = 1, …, NODE − 1.

ICRSON — Vector of length NODE − 1 containing the right son nodes. (Input)
Node number NODE + K has right son given by ICRSON(K) for K = 1, …, NODE − 1.

CLEVEL — Vector of length NODE − 1 containing the level used in merging or splitting the son nodes. (Input)
CLEVEL(K) specifies the scale to be used on the vertical (IMETH = 1 or 2) or horizontal (IMETH = 3) axis for node NODE + K, for K = 1, 2, …, NODE − 1.

NSCALE — Number of horizontal slices of tree. (Input)
NSCALE must be positive.

SCALE — Vector of length two giving the interval on the CLEVEL axis which should be used to plot the tree. (Input)
SCALE(1) is the location for printing the terminal nodes. The root node is printed at SCALE(2).

NODENM — CHARACTER*(*) vector of length NODE containing the terminal node labels. (Input)
If terminal node labels are to be 1, 2, 3, …, then NODENM(1) should be "DEFAULT" and

the remaining elements of NODENM are not used. The length of each label is M, where M is determined by the user.

Optional Arguments

NODE — Initial number of observations or nodes. (Input)
NODE must be greater than 2.
Default: NODE = size (ICLSON,1) + 1.

IMETH — Method to be used for printing the binary tree. (Input)
Default: IMETH = 1.

IMETH	Method
1	Horizontal tree
2	Horizontal *I*-tree
3	Vertical tree

IROOT — Subtree specification. (Input)
IROOT specifies the root node of the subtree to be printed. If IROOT = 2 * NODE – 1 (or zero for the default), the entire tree is printed. IROOT must be in the range NODE + 1 to 2 * NODE – 1.
Default: IROOT = 0.

NFILL — The number of filler lines printed between horizontal or vertical node lines. (Input)
NFILL = 1 is usually sufficient. NFILL must be nonnegative.
Default: NFILL = 1.

FORTRAN 90 Interface

Generic: `CALL TREEP (ICLSON, ICRSON, CLEVEL, NSCALE, SCALE, NODENM [,…])`

Specific: The specific interface names are S_TREEP and D_TREEP.

FORTRAN 77 Interface

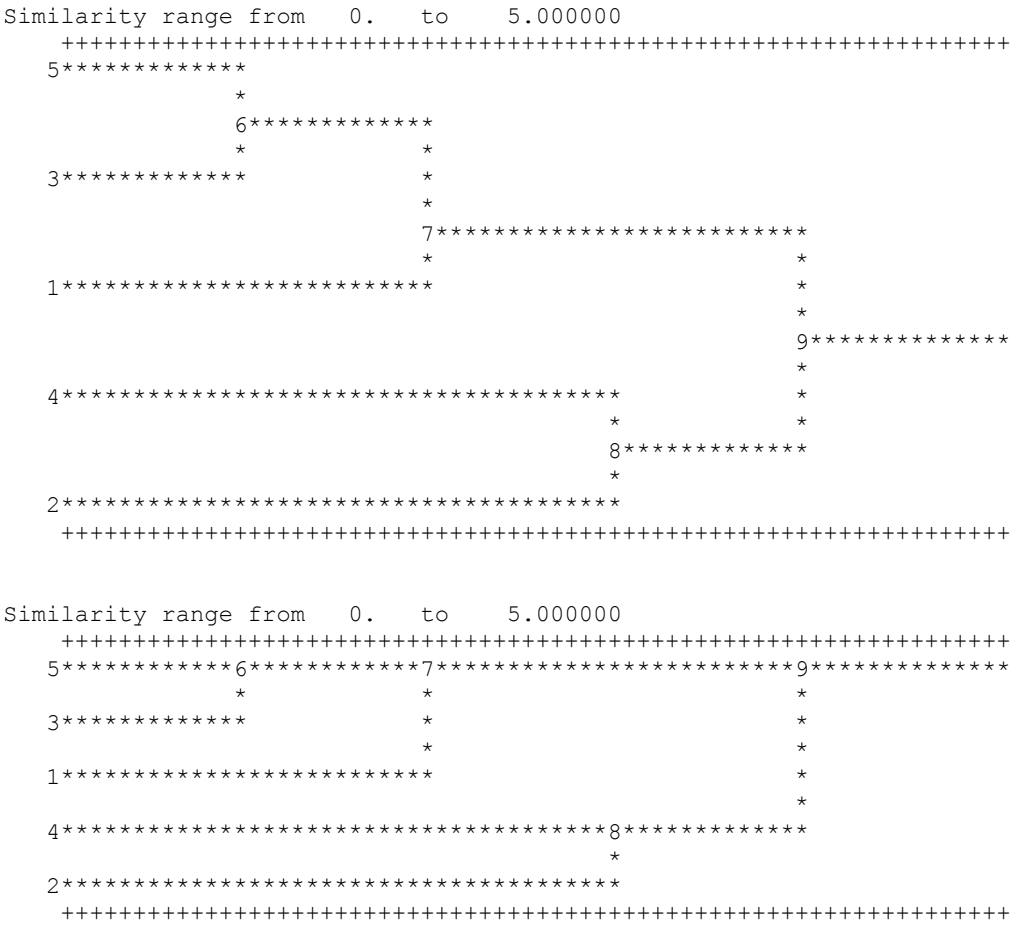
Single: `CALL TREEP (NODE, ICLSON, ICRSON, IMETH, CLEVEL, IROOT, NSCALE, NFILL, SCALE, NODENM)`

Double: The double precision name is DTREEP.

Example

```
      USE PGOPT_INT
      USE TREEP_INT
      USE UMACH_INT
      INTEGER    NODE
      PARAMETER  (NODE=5)
!
      INTEGER    ICLSON(NODE-1), ICRSON(NODE-1), IMETH, IPAGE, NOUT, NSCALE
      REAL       CLEVEL(NODE-1), SCALE(2)
      CHARACTER  NODENM(NODE)*7
!
      DATA ICLSON/5, 6, 4, 7/
      DATA ICRSON/3, 1, 2, 8/
      DATA NODENM/'DEFAULT', ' ', ' ', ' ', ' '/
      DATA CLEVEL/1., 2., 3., 4./
      DATA SCALE/0., 5./
!                                Set page width
      IPAGE = 70
      CALL PGOPT (-1, IPAGE)
      NSCALE = 1
!                                Horizontal tree
      IMETH = 1
      CALL TREEP (ICLSON, ICRSON, CLEVEL, NSCALE, SCALE, NODENM, &
                  IMETH=IMETH)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999)
99999 FORMAT (1X, //////)
!                                Horizontal I-tree
      IMETH = 2
      CALL TREEP (ICLSON, ICRSON, CLEVEL, NSCALE, SCALE, NODENM, &
                  IMETH=IMETH)
!
      END
```

Output



Comments

1. Workspace may be explicitly provided, if desired, by use of `T2EEP/DT2EEP`. The reference is:

   ```
   CALL T2EEP (NODE, ICLSON, ICRSON, IMETH, CLEVEL, IROOT, NSCALE,
   NFILL, SCALE, NODENM, IDTREE, ISTREE, IOTREE, INTREE, TLTREE)
   ```

   The additional arguments are as follows:

   IDTREE — Work vector of length `IROOT`. `IDTREE` is used to store the distance of each node from the vertical axis in vertical tree.

   ISTREE — Work vector of length `IROOT` used to store all the nodes. `IROOT` is the first element of the array.

IOTREE — Work vector of length IROOT + 1 used to store the index of each node as TLTREE is sorted.

INTREE — Work vector of length IROOT.

TLTREE — Work vector of length IROOT + 1 used to store the level of each node in descending order in a vertical tree. It is used to store the distance of each node from the top of the horizontal line in ascending order in a horizontal tree.

2. Printing starts on a new page with default page width 78. The user may change it by calling the routine PGOPT in advance.

Description

Routine TREEP prints a binary tree which may represent results of hierarchical clustering algorithm such as the routine CLINK.

Let M = NODE indicate the number of nodes. A binary tree is composed of M terminal nodes and $M - 1$ nonterminal nodes uniquely numbered 1 to M and $M + 1$ to $M + (M - 1)$, respectively. Each nonterminal node joins together two son nodes which may or may not be terminal. Nonterminal nodes $M + K$ are printed on the vertical scale interval $[S_1, S_2]$ at the level given in C_K, for $K = 1, 2, \ldots, M - 1$, where S_1 = SCALE(1), S_2 = SCALE(2), and C_K = CLEVEL(K).

Chapter 17: Probability Distribution Functions and Inverses

Routines

Usage Notes

Comments

Definitions and discussions of the terms basic to this chapter can be found in Johnson and Kotz (1969, 1970a, 1970b). These are also good references for the specific distributions.

In order to keep the calling sequences simple, whenever possible, the subprograms described in this chapter are written for standard forms of statistical distributions. Hence, the number of parameters for any given distribution may be fewer than the number often associated with the distribution. For example, while a gamma distribution is often characterized by two parameters (or even a third, "location"), there is only one parameter that is necessary, the "shape". The "scale" parameter can be used to scale the variable to the standard gamma distribution. Also, the functions relating to the normal distribution, `ANORDF` (page 1262) and `ANORIN` (page 1264), are for a normal distribution with mean equal to zero and variance equal to one. For other means and variances, it is very easy for the user to standardize the variables by subtracting the mean and dividing by the square root of the variance.

The *distribution function* for the (real, single-valued) random variable X is the function F defined for all real x by

$$F(x) = \text{Prob}(X \leq x)$$

where Prob(·) denotes the probability of an event. The distribution function is often called the *cumulative distribution function* (CDF).

For distributions with finite ranges, such as the beta distribution, the CDF is 0 for values less than the left endpoint and 1 for values greater than the right endpoint. The subprograms described in this chapter return the correct values for the distribution functions when values outside of the range of the random variable are input, but warning error conditions are set in these cases.

Discrete Random Variables

For discrete distributions, the function giving the probability that the random variable takes on specific values is called the *probability function*, defined by

$$p(x) = \text{Prob}(X = x)$$

The "`PR`" routines described in this chapter evaluate probability functions.

The CDF for a discrete random variable is

$$F(x) = \sum_{A} p(k)$$

where A is the set such that $k \le x$. The "DF" routines in this chapter evaluate cumulative distribution functions. Since the distribution function is a step function, its inverse does not exist uniquely.

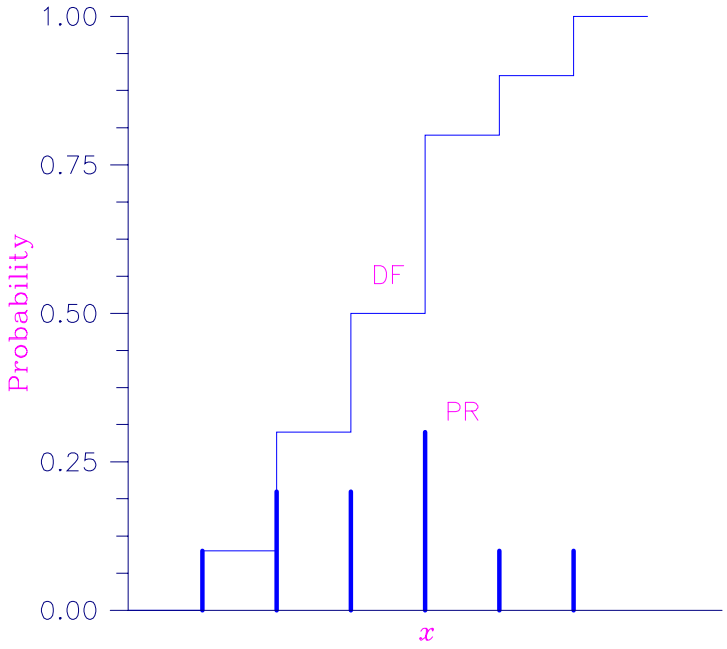


Figure 17-1 Discrete Random Variable

In the plot above, a routine like BINPR (page 1248) in this chapter evaluates the individual probability, given X. A routine like BINDF (page 1246) would evaluate the sum of the probabilities up to and including the probability at X.

Continuous Distributions

For continuous distributions, a probability function, as defined above, would not be useful because the probability of any given point is 0. For such distributions, the useful analog is the *probability density function* (PDF). The integral of the PDF is the probability over the interval, if the continuous random variable X has PDF f, then

$$\text{Prob}(a < X \le b) = \int_a^b f(x)dx$$

The relationship between the CDF and the PDF is

$$F(x) = \int_{-\infty}^{x} f(t)dt$$

as shown in Figure 17-2.

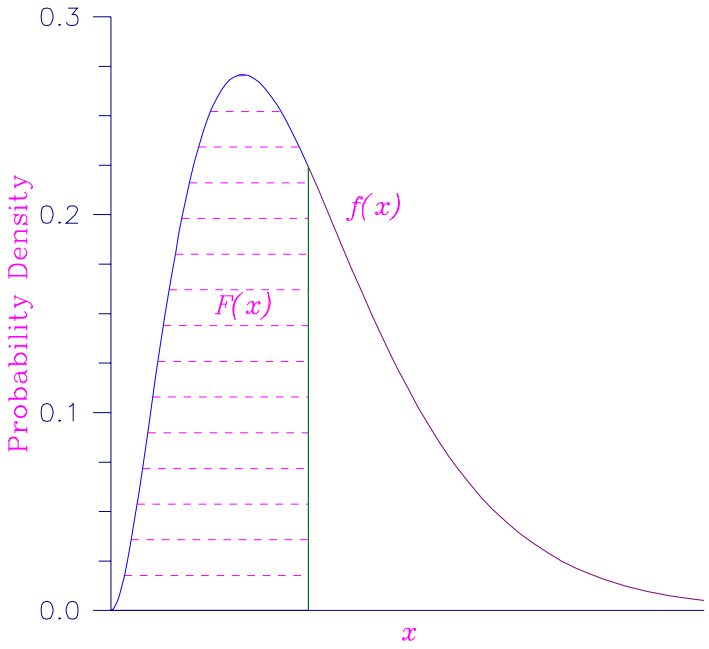


Figure 17-2 Probability Density Function

The "DF" routines described in this chapter evaluate cumulative distribution functions.

For (absolutely) continuous distributions, the value of $F(x)$ uniquely determines x within the support of the distribution. The "IN" routines described in this chapter compute the inverses of the distribution functions, that is, given $F(x)$ (called "P" for "probability"), a routine such as BETIN (page 1268) computes x. The inverses are defined only over the open interval (0,1).

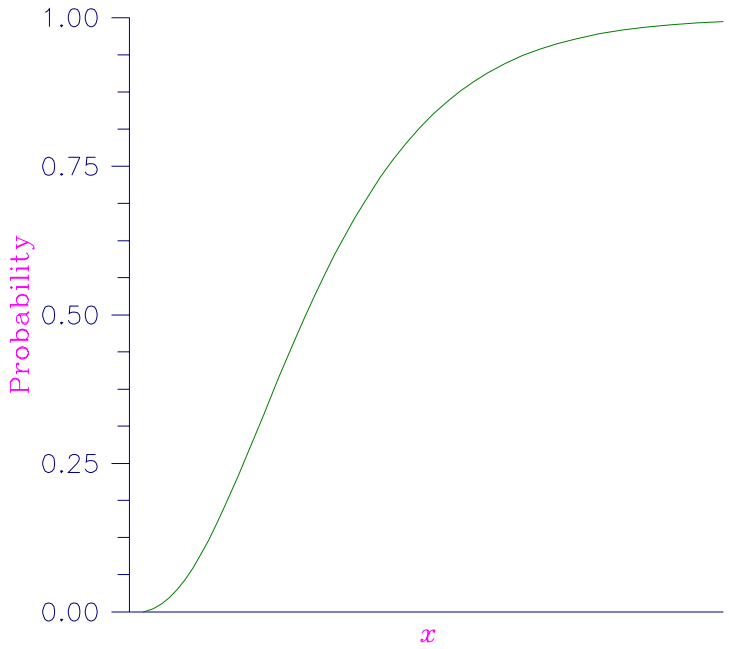


Figure 17-3 Cumulative Probability Distribution Function

There are three routines described in this chapter that deal with general continuous distribution functions. The routine GCDF (page 1295) computes a distribution function using values of the density function, and the routine GCIN (page 1298) computes the inverse. These two routines may be useful when the user has an estimate of a probability density, as perhaps computed by the routine DESPL or DESKN (see Chapter 17, Probability Distribution Functions and Inverses), or computed from a frequency polygon. The routine GFNIN (page 1301) computes the inverse of a distribution function that is specified as a FORTRAN function.

Additional Comments

Whenever a probability close to 1.0 results from a call to a distribution function or is to be input to an inverse function, it is often impossible to achieve good accuracy because of the nature of the representation of numeric values. In this case, it may be better to work with the complementary distribution function (one minus the distribution function). If the distribution is symmetric about some point (as the normal distribution, for example) or is reflective about some point (as the beta distribution, for example), the complementary distribution function has a simple relationship with the distribution function. For example, to evaluate the standard normal distribution at 4.0, using ANORIN (page 1264) directly, the result to six places is 0.999968. Only two of those digits are really useful, however. A more useful result may be 1.000000 minus this value, which can be obtained to six significant figures as 3.16713E-05 by evaluating ANORIN at −4.0. For the normal distribution, the two values are related by $\Phi(x) = 1 - \Phi(-x)$, where $\Phi(\cdot)$ is the normal distribution function. Another example is the beta distribution with parameters 2 and 10. This distribution is skewed to the right, so evaluating BETDF (page 1265) at 0.7, we obtain 0.999953. A more precise

result is obtained by evaluating BETDF with parameters 10 and 2 at 0.3. This yields 4.72392E-5. (In both of these examples, it is wise not to trust the last digit.)

Many of the algorithms used by routines in this chapter are discussed by Abramowitz and Stegun (1964). The algorithms make use of various expansions and recursive relationships and often use different methods in different regions.

Cumulative distribution functions are defined for all real arguments, however, if the input to one of the distribution functions in this chapter is outside the range of the random variable, an error of Type 1 is issued, and the output is set to zero or one, as appropriate. A Type 1 error is of lowest severity, a "note", and, by default, no printing or stopping of the program occurs. The other common errors that occur in the routines of this chapter are Type 2, "alert", for a function value being set to zero due to underflow, Type 3, "warning", for considerable loss of accuracy in the result returned, and Type 5, "terminal", for incorrect and/or inconsistent input, complete loss of accuracy in the result returned, or inability to represent the result (because of overflow). When a Type 5 error occurs, the result is set to NaN (not a number, also used as a missing value code, obtained by routine AMACH(6). (See the section "User Errors" in the Reference Material.)

BINDF

This function evaluates the binomial distribution function.

Function Return Value

BINDF — Function value, the probability that a binomial random variable takes a value less than or equal to K. (Output)
BINDF is the probability that K or fewer successes occur in N independent Bernoulli trials, each of which has a P probability of success.

Required Arguments

K — Argument for which the binomial distribution function is to be evaluated. (Input)

N — Number of Bernoulli trials. (Input)

P — Probability of success on each trial. (Input)

FORTRAN 90 Interface

Generic: `BINDF(K, N, P)`

Specific: The specific interface names are `S_BINDF` and `D_BINDF`.

FORTRAN 77 Interface

Single: `BINDF(K, N, P)`

Double: The double precision name is `DBINDF`.

Example

Suppose X is a binomial random variable with $n = 5$ and $p = 0.95$. In this example, we find the probability that X is less than or equal to 3.

```
      USE UMACH_INT
      USE BINDF_INT
      INTEGER    K, N, NOUT
      REAL       P, PR
!
      CALL UMACH (2, NOUT)
      K  = 3
      N  = 5
      P  = 0.95
      PR = BINDF(K,N,P)
      WRITE (NOUT,99999) PR
99999 FORMAT (' The probability that X is less than or equal to 3 is ' &
            , F6.4)
      END
```

Output

```
The probability that X is less than or equal to 3 is 0.0226
```

Comments

Informational errors

Type	Code	
1	3	The input argument, K, is less than zero.
1	4	The input argument, K, is greater than the number of Bernoulli trials, N.

Description

Function BINDF evaluates the distribution function of a binomial random variable with parameters n and p. It does this by summing probabilities of the random variable taking on the specific values in its range. These probabilities are computed by the recursive relationship

$$\Pr(X=j) = \frac{(n+1-j)p}{j(1-p)} \Pr(X = j-1)$$

To avoid the possibility of underflow, the probabilities are computed forward from 0, if k is not greater than n times p, and are computed backward from n, otherwise. The smallest positive machine number, ε, is used as the starting value for summing the probabilities, which are rescaled by $(1 - p)^n\varepsilon$ if forward computation is performed and by $p^n\varepsilon$ if backward computation is done. For the special case of $p = 0$, BINDF is set to 1; and for the case $p = 1$, BINDF is set to 1 if $k = n$ and to 0 otherwise.

BINPR

This function evaluates the binomial probability function.

Function Return Value

BINPR Function value, the probability that a binomial random variable takes a value equal to K. (Output)

Required Arguments

K — Argument for which the binomial probability function is to be evaluated. (Input)

N — Number of Bernoulli trials. (Input)

P — Probability of success on each trial. (Input)

FORTRAN 90 Interface

Generic: `BINPR(K, N, P)`

Specific: The specific interface names are `S_BINPR` and `D_BINPR`.

FORTRAN 77 Interface

Single: `BINPR(K, N, P)`

Double: The double precision name is `DBINPR`.

Example

Suppose X is a binomial random variable with $n = 5$ and $p = 0.95$. In this example, we find the probability that X is equal to 3.

```
      USE UMACH_INT
      USE BINPR_INT
      INTEGER    K, N, NOUT
      REAL       P, PR
!
      CALL UMACH (2, NOUT)
      K  = 3
      N  = 5
      P  = 0.95
      PR = BINPR(K,N,P)
      WRITE (NOUT,99999) PR
99999 FORMAT (' The probability that X is equal to 3 is ', F6.4)
      END
```

Output

```
The probability that X is equal to 3 is 0.0214
```

Comments

Informational errors

Type	Code	
1	3	The input argument, K, is less than zero.
1	4	The input argument, K, is greater than the number of Bernoulli trials, N.

Description

The function BINPR evaluates the probability that a binomial random variable with parameters n and p takes on the value k. It does this by computing probabilities of the random variable taking on the values in its range less than (or the values greater than) k. These probabilities are computed by the recursive relationship

$$\Pr(X = j) = \frac{(n+1-j)p}{j(1-p)} \Pr(X = j-1)$$

To avoid the possibility of underflow, the probabilities are computed forward from 0, if k is not greater than n times p, and are computed backward from n, otherwise. The smallest positive machine number, ε, is used as the starting value for computing the probabilities, which are rescaled by $(1-p)^n\varepsilon$ if forward computation is performed and by $p^n\varepsilon$ if backward computation is done.

For the special case of $p = 0$, BINPR is set to 0 if k is greater than 0 and to 1 otherwise; and for the case $p = 1$, BINPR is set to 0 if k is less than n and to 1 otherwise.

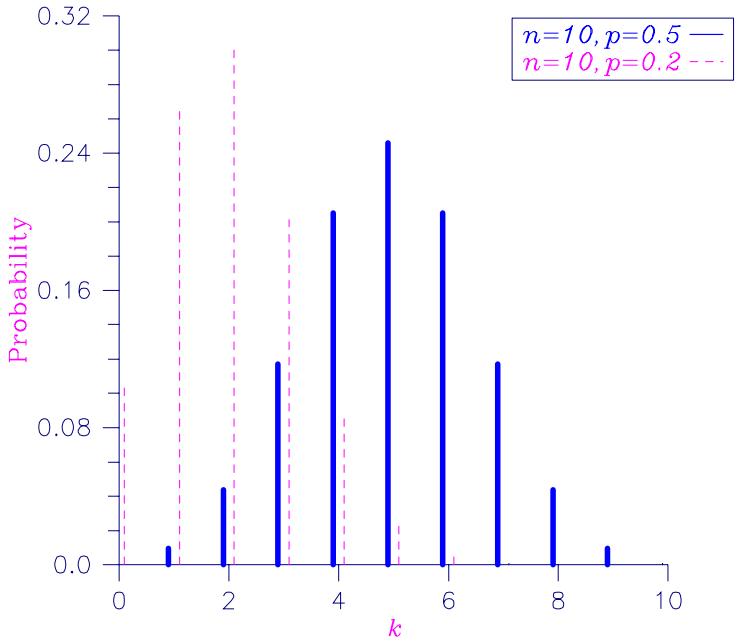


Figure 17-4 Binomial Probability Function

HYPDF

This function evaluates the hypergeometric distribution function.

Function Return Value

HYPDF — Function value, the probability that a hypergeometric random variable takes a value less than or equal to `K`. (Output)
`HYPDF` is the probability that `K` or fewer defectives occur in a sample of size `N` drawn from a lot of size `L` that contains `M` defectives.
See Comment 1.

Required Arguments

K — Argument for which the hypergeometric distribution function is to be evaluated. (Input)

N — Sample size. (Input)
`N` must be greater than zero and greater than or equal to `K`.

M — Number of defectives in the lot. (Input)

L — Lot size. (Input)
`L` must be greater than or equal to `N` and `M`.

FORTRAN 90 Interface

Generic: `HYPDF(K, N, M, L)`

Specific: The specific interface names are `S_HYPDF` and `D_HYPDF`.

FORTRAN 77 Interface

Single: `HYPDF(K, N, M, L)`

Double: The double precision name is `DHYPDF`.

Example

Suppose X is a hypergeometric random variable with $n = 100$, $l = 1000$, and $m = 70$. In this example, we evaluate the distribution function at 7.

```
      USE UMACH_INT
      USE HYPDF_INT
      INTEGER    K, L, M, N, NOUT
      REAL       DF
!
      CALL UMACH (2, NOUT)
      K  = 7
      N  = 100
      L  = 1000
      M  = 70
      DF = HYPDF(K,N,M,L)
      WRITE (NOUT,99999) DF
99999 FORMAT (' The probability that X is less than or equal to 7 is ' &
            , F6.4)
      END
```

Output

```
The probability that X is less than or equal to 7 is 0.5995
```

Comments

1. If the generic version of this function is used, the immediate result must be stored in a variable before use in an expression. For example:

   ```
   X = HYPDF(K, N, M, L)
   Y = SQRT(X)
   ```

   must be used rather than

   ```
   Y = SQRT(HYPDF(K, N, M, L))
   ```

   If this is too much of a restriction on the programmer, then the specific name can be used without this restriction.

2. Informational errors

Type	Code	
1	5	The input argument, K, is less than zero.
1	6	The input argument, K, is greater than the sample size.

Description

The function HYPDF evaluates the distribution function of a hypergeometric random variable with parameters n, l, and m. The hypergeometric random variable X can be thought of as the number of items of a given type in a random sample of size n that is drawn without replacement from a population of size l containing m items of this type. The probability function is

$$\Pr(X = j) = \frac{\binom{m}{j}\binom{l-m}{n-j}}{\binom{l}{n}} \quad \text{for } j = i, i+1, i+2, \ldots \min(n, m)$$

where $i = \max(0, n - l + m)$.

If k is greater than or equal to i and less than or equal to $\min(n, m)$, HYPDF sums the terms in this expression for j going from i up to k. Otherwise, HYPDF returns 0 or 1, as appropriate. So, as to avoid rounding in the accumulation, HYPDF performs the summation differently depending on whether or not k is greater than the mode of the distribution, which is the greatest integer less than or equal to $(m + 1)(n + 1)/(l + 2)$.

HYPPR

This function evaluates the hypergeometric probability function.

Function Return Value

HYPPR — Function value, the probability that a hypergeometric random variable takes a value equal to K. (Output)
HYPPR is the probability that exactly K defectives occur in a sample of size N drawn from a lot of size L that contains M defectives.
See Comment 1.

Required Arguments

K — Argument for which the hypergeometric probability function is to be evaluated. (Input)

N — Sample size. (Input)
N must be greater than zero and greater than or equal to K.

M — Number of defectives in the lot. (Input)

L — Lot size. (Input)
L must be greater than or equal to N and M.

FORTRAN 90 Interface

Generic: `HYPPR(K, N, M, L)`

Specific: The specific interface names are S_HYPPR and D_HYPPR.

FORTRAN 77 Interface

Single: `HYPPR(K, N, M, L)`

Double: The double precision name is DHYPPR.

Example

Suppose X is a hypergeometric random variable with $n = 100$, $l = 1000$, and $m = 70$. In this example, we evaluate the probability function at 7.

```
      USE UMACH_INT
      USE HYPPR_INT
      INTEGER    K, L, M, N, NOUT
      REAL       PR
!
      CALL UMACH (2, NOUT)
      K  = 7
      N  = 100
      L  = 1000
      M  = 70
      PR = HYPPR(K,N,M,L)
      WRITE (NOUT,99999) PR
99999 FORMAT (' The probability that X is equal to 7 is ', F6.4)
      END
```

Output

```
The probability that X is equal to 7 is 0.1628
```

Comments

1. If the generic version of this function is used, the immediate result must be stored in a variable before use in an expression. For example:

   ```
   X = HYPPR(K, N, M, L)
   Y = SQRT(X)
   ```

   must be used rather than

   ```
   Y = SQRT(HYPPR(K, N, M, L))
   ```

If this is too much of a restriction on the programmer, then the specific name can be used without this restriction.Informational errors

Type	Code	
1	5	The input argument, K, is less than zero.
1	6	The input argument, K, is greater than the sample size.

Description

The function HYPPR evaluates the probability function of a hypergeometric random variable with parameters n, l, and m. The hypergeometric random variable X can be thought of as the number of items of a given type in a random sample of size n that is drawn without replacement from a population of size l containing m items of this type. The probability function is

$$\Pr(X = k) = \frac{\binom{m}{k}\binom{l-m}{n-k}}{\binom{l}{n}} \quad \text{for } k = i, i+1, i+2, \ldots \min(n, m)$$

where $i = \max(0, n - l + m)$. HYPPR evaluates the expression using log gamma functions.m,

POIDF

This function evaluates the Poisson distribution function.

Function Return Value

POIDF — Function value, the probability that a Poisson random variable takes a value less than or equal to K. (Output)

Required Arguments

K — Argument for which the Poisson distribution function is to be evaluated. (Input)

THETA — Mean of the Poisson distribution. (Input)
THETA must be positive.

FORTRAN 90 Interface

Generic: `POIDF(K, THETA)`

Specific: The specific interface names are `S_POIDF` and `D_POIDF`.

FORTRAN 77 Interface

Single: `POIDF(K, THETA)`

Double: The double precision name is `DPOIDF`.

Example

Suppose X is a Poisson random variable with $\theta = 10$. In this example, we evaluate the distribution function at 7.

```
      USE UMACH_INT
      USE POIDF_INT

      INTEGER    K, NOUT
      REAL       DF, THETA
!
      CALL UMACH (2, NOUT)
      K     = 7
      THETA = 10.0
      DF    = POIDF(K,THETA)
      WRITE (NOUT,99999) DF
99999 FORMAT (' The probability that X is less than or equal to ', &
             '7 is ', F6.4)
      END
```

Output

```
The probability that X is less than or equal to 7 is 0.2202
```

Comments

Informational error

Type	Code	
1	1	The input argument, `K`, is less than zero.

Description

The function `POIDF` evaluates the distribution function of a Poisson random variable with parameter `THETA`. `THETA`, which is the mean of the Poisson random variable, must be positive. The probability function (with θ = `THETA`) is

$$f(x) = e^{-\theta}\,\theta^{x}/x!, \qquad \text{for } x = 0, 1, 2,\ldots$$

The individual terms are calculated from the tails of the distribution to the mode of the distribution and summed. `POIDF` uses the recursive relationship

$$f(x+1) = f(x)\theta/(x+1), \qquad \text{for } x = 0, 1, 2, \ldots k-1,$$

with $f(0) = e^{-\theta}$.

POIPR

This function evaluates the Poisson probability function.

Function Return Value

POIPR — Function value, the probability that a Poisson random variable takes a value equal to K. (Output)

Required Arguments

K — Argument for which the Poisson distribution function is to be evaluated. (Input)

THETA — Mean of the Poisson distribution. (Input)
THETA must be positive.

FORTRAN 90 Interface

Generic: `POIPR(K, THETA)`

Specific: The specific interface names are `S_POIPR` and `D_POIPR`.

FORTRAN 77 Interface

Single: `POIPR(K, THETA)`

Double: The double precision name is `DPOIPR`.

Example

Suppose X is a Poisson random variable with $\theta = 10$. In this example, we evaluate the probability function at 7.

```
      USE UMACH_INT
      USE POIPR_INT

      INTEGER    K, NOUT
      REAL       PR, THETA
!
      CALL UMACH (2, NOUT)
      K     = 7
      THETA = 10.0
      PR    = POIPR(K,THETA)
      WRITE (NOUT,99999) PR
99999 FORMAT (' The probability that X is equal to 7 is ', F6.4)
      END
```

Output

```
The probability that X is equal to 7 is 0.0901
```

Comments

Informational error

Type	Code	
1	1	The input argument, K, is less than zero.

Description

The function POIPR evaluates the probability function of a Poisson random variable with parameter THETA. THETA, which is the mean of the Poisson random variable, must be positive. The probability function (with θ = THETA) is

$$f(x) = e^{-\theta}\,\theta^{k}/k!, \qquad \text{for } k = 0, 1, 2,\ldots$$

POIPR evaluates this function directly, taking logarithms and using the log gamma function.

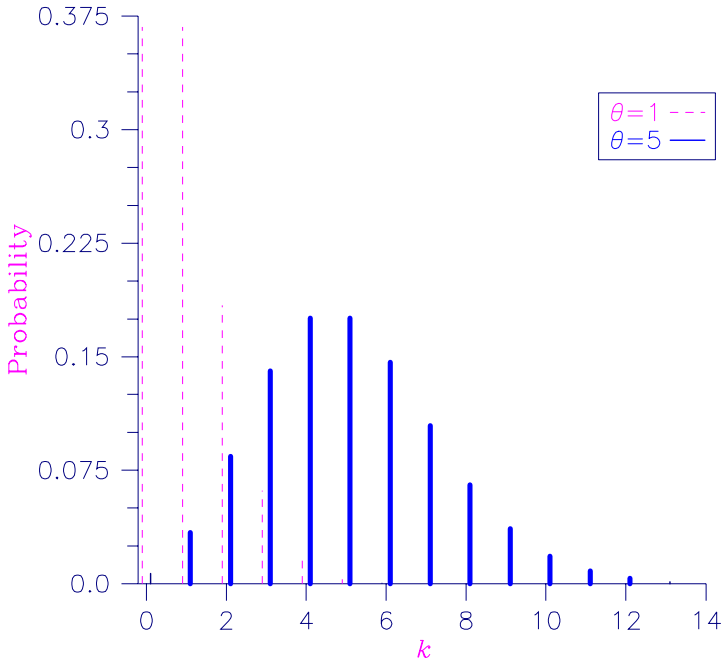


Figure 17-5 Poisson Probability Function

AKS1DF

This function evaluates the distribution function of the one-sided Kolmogorov-Smirnov goodness of fit D^+ or D^- test statistic based on continuous data for one sample.

Function Return Value

AKS1DF — The probability of a smaller `D`. (Output)

Required Arguments

NOBS — The total number of observations in the sample. (Input)

D — The D^+ or D^- test statistic. (Input)
`D` is the maximum positive difference of the empirical cumulative distribution function (CDF) minus the hypothetical CDF or the maximum positive difference of the hypothetical CDF minus the empirical CDF.

FORTRAN 90 Interface

Generic: `AKS1DF(NOBS, D)`

Specific: The specific interface names are `S_AKS1DF` and `D_AKS1DF`.

FORTRAN 77 Interface

Single: `AKS1DF(NOBS, D)`

Double: The double precision name is `DKS1DF`.

Example

In this example, the exact one-sided probabilities for the tabled values of D^+ or D^-, given, for example, in Conover (1980, page 462), are computed. Tabled values at the 10% level of significance are used as input to `AKS1DF` for sample sizes of 5 to 50 in increments of 5 (the last two tabled values are obtained using the asymptotic critical values of

$$1.07 / \sqrt{\text{NOBS}}$$

The resulting probabilities should all be close to 0.90.

```
      USE UMACH_INT
      USE AKS1DF_INT
      INTEGER    I, NOBS, NOUT
      REAL       D(10)
!
      DATA D/0.447, 0.323, 0.266, 0.232, 0.208, 0.190, 0.177, 0.165, &
          0.160, 0.151/
!
      CALL UMACH (2, NOUT)
!
      DO 10  I=1, 10
         NOBS = 5*I
!
         WRITE (NOUT,99999) D(I), NOBS, AKS1DF(NOBS,D(I))
!
```

```
99999    FORMAT (' One-sided Probability for D = ', F8.3, ' with NOBS ' &
               , '= ', I2, ' is ', F8.4)
   10 CONTINUE
      END
```

Output

```
One-sided Probability for D =    0.447 with NOBS =  5 is   0.9000
One-sided Probability for D =    0.323 with NOBS = 10 is   0.9006
One-sided Probability for D =    0.266 with NOBS = 15 is   0.9002
One-sided Probability for D =    0.232 with NOBS = 20 is   0.9009
One-sided Probability for D =    0.208 with NOBS = 25 is   0.9002
One-sided Probability for D =    0.190 with NOBS = 30 is   0.8992
One-sided Probability for D =    0.177 with NOBS = 35 is   0.9011
One-sided Probability for D =    0.165 with NOBS = 40 is   0.8987
One-sided Probability for D =    0.160 with NOBS = 45 is   0.9105
One-sided Probability for D =    0.151 with NOBS = 50 is   0.9077
```

Comments

1. Workspace may be explicitly provided, if desired, by use of AK21DF/DK21DF. The reference is:

   ```
   AK2DF(NOBS, D, WK)
   ```

   The additional argument is:

   WK — Work vector of length 3 * NOBS + 3 if NOBS ≤ 80. WK is not used if NOBS is greater than 80.

2. Informational errors

Type	Code	
1	2	Since the D test statistic is less than zero, the distribution function is zero at D.
1	3	Since the D test statistic is greater than one, the distribution function is one at D.

3. If NOBS ≤ 80, then exact one-sided probabilities are computed. In this case, on the order of $NOBS^2$ operations are required. For NOBS > 80, approximate one-sided probabilities are computed. These approximate probabilities require very few computations.

4. An approximate two-sided probability for the D = max (D^+, D^-) statistic can be computed as twice the AKS1DF probability for D(minus one, if the probability from AKS1DF is greater than 0.5).

Description

Routine AKS1DF computes the cumulative distribution function (CDF) for the one-sided Kolmogorov-Smirnov one-sample D^+ or D^- statistic when the theoretical CDF is strictly continuous. Let $F(x)$ denote the theoretical distribution function, and let $S_n(x)$ denote the

empirical distribution function obtained from a sample of size NOBS. Then, the D^+ statistic is computed as

$$D^+ = \sup_x [F(x) - S_n(x)]$$

while the one-sided D^- statistic is computed as

$$D^- = \sup_x [S_n(x) - F(x)]$$

Exact probabilities are computed according to a method given by Conover (1980, page 350) for sample sizes of 80 or less. For sample sizes greater than 80, Smirnov's asymptotic result is used, that is, the value of the CDF is taken as $1 - \exp(-2nd^2)$, where d is D^+ or D^- (Kendall and Stuart, 1979, page 482). This asymptotic expression is conservative (the value returned by AKS1DF is smaller than the exact value, when the sample size exceeds 80).

Programming Notes

Routine AKS1DF requires on the order of NOBS^2 operations to compute the exact probabilities, where an operation consists of taking ten or so logarithms. Because so much computation is occurring within each "operation," AKS1DF is much slower than its two-sample counterpart, function AKS2DF (page 1260).

AKS2DF

This function evaluates the distribution function of the Kolmogorov-Smirnov goodness of fit D test statistic based on continuous data for two samples.

Function Return Value

AKS2DF — The probability of a smaller D. (Output)

Required Arguments

NOBSX — The total number of observations in the first sample. (Input)

NOBSY — The total number of observations in the second sample. (Input)

D — The D test statistic. (Input)
D is the maximum absolute difference between empirical cumulative distribution functions (CDFs) of the two samples.

FORTRAN 90 Interface

Generic: `AKS2DF(NOBSX, NOBSY, D)`

Specific: The specific interface names are S_AKS2DF and D_AKS2DF.

FORTRAN 77 Interface

Single: `AKS2DF(NOBSX, NOBSY, D)`

Double: The double precision name is `DKS2DF`.

Example

Function `AKS2DF` is used to compute the probability of a smaller *D* statistic for a variety of sample sizes using values close to the 0.95 probability value.

```
      USE UMACH_INT
      USE AKS2DF_INT
      INTEGER    I, NOBSX(10), NOBSY(10), NOUT
      REAL       D(10)
!
      DATA NOBSX/5, 20, 40, 70, 110, 200, 200, 200, 100, 100/
      DATA NOBSY/10, 10, 10, 10, 10, 20, 40, 60, 80, 100/
      DATA D/0.7, 0.55, 0.475, 0.4429, 0.4029, 0.2861, 0.2113, 0.1796, &
          0.18, 0.18/
!
      CALL UMACH (2, NOUT)
!
      DO 10  I=1, 10
!
         WRITE (NOUT,99999) D(I), NOBSX(I), NOBSY(I), &
                            AKS2DF(NOBSX(I),NOBSY(I),D(I))
!
99999    FORMAT (' Probability for D = ', F5.3, ' with NOBSX = ', I3, &
               ' and NOBSY = ', I3, ' is ', F9.6, '.')
   10 CONTINUE
      END
```

Output

```
Probability for D = 0.700 with NOBSX =   5 and NOBSY =  10 is  0.980686.
Probability for D = 0.550 with NOBSX =  20 and NOBSY =  10 is  0.987553.
Probability for D = 0.475 with NOBSX =  40 and NOBSY =  10 is  0.972423.
Probability for D = 0.443 with NOBSX =  70 and NOBSY =  10 is  0.961646.
Probability for D = 0.403 with NOBSX = 110 and NOBSY =  10 is  0.928667.
Probability for D = 0.286 with NOBSX = 200 and NOBSY =  20 is  0.921126.
Probability for D = 0.211 with NOBSX = 200 and NOBSY =  40 is  0.917110.
Probability for D = 0.180 with NOBSX = 200 and NOBSY =  60 is  0.914520.
Probability for D = 0.180 with NOBSX = 100 and NOBSY =  80 is  0.908185.
Probability for D = 0.180 with NOBSX = 100 and NOBSY = 100 is  0.946098.
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `AK22DF/DK22DF`. The reference is:

   ```
   AK22DF(NOBSX, NOBSY, D, WK)
   ```

   The additional argument is:

WK — Work vector of length max(NOBSX, NOBSY) + 1.

2. Informational errors

Type	Code	
1	2	Since the D test statistic is less than zero, then the distribution function is zero at D.
1	3	Since the D test statistic is greater than one, then the distribution function is one at D.

Description

Function AKS2DF computes the cumulative distribution function (CDF) for the two-sided Kolmogorov-Smirnov two-sample *D* statistic when the theoretical CDF is strictly continuous. Exact probabilities are computed according to a method given by Kim and Jennrich (1973). Approximate asymptotic probabilities are computed according to methods also given in this reference.

Let $F_n(x)$ and $G_m(x)$ denote the empirical distribution functions for the two samples, based on n = NOBSX and m = NOBSY observations. Then, the D statistic is computed as

$$D = \sup_x \left| F_n(x) - G_m(x) \right|$$

Programming Notes

Function AKS2DF requires on the order of NOBSX * NOBSY operations to compute the exact probabilities, where an operation consists of an addition and a multiplication. For NOBSX * NOBSY less than 10000, the exact probability is computed. If this is not the case, then the Smirnov approximation discussed by Kim and Jennrich (1973) is used if the minimum of NOBSX and NOBSY is greater than ten percent of the maximum of NOBSX and NOBSY, or if the minimum is greater than 80. Otherwise, the Kolmogorov approximation discussed by Kim and Jennrich (1973) is used.

ANORDF

This fuction evaluates the standard normal (Gaussian) distribution function.

Function Return Value

ANORDF — Function value, the probability that a normal random variable takes a value less than or equal to X. (Output)

Required Arguments

X — Argument for which the normal distribution function is to be evaluated. (Input)

FORTRAN 90 Interface

Generic: `ANORDF(X)`

Specific: The specific interface names are `S_ANORDF` and `D_ANORDF`.

FORTRAN 77 Interface

Single: `ANORDF(X)`

Double: The double precision name is `DNORDF`.

Example

Suppose X is a normal random variable with mean 100 and variance 225. In this example, we find the probability that X is less than 90, and the probability that X is between 105 and 110.

```
      USE UMACH_INT
      USE ANORDF_INT
      INTEGER    NOUT
      REAL       P, X1, X2
!
      CALL UMACH (2, NOUT)
      X1 = (90.0-100.0)/15.0
      P  = ANORDF(X1)
      WRITE (NOUT,99998) P
99998 FORMAT (' The probability that X is less than 90 is ', F6.4)
      X1 = (105.0-100.0)/15.0
      X2 = (110.0-100.0)/15.0
      P  = ANORDF(X2) - ANORDF(X1)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that X is between 105 and 110 is ', &
            F6.4)
      END
```

Output

```
The probability that X is less than 90 is 0.2525
The probability that X is between 105 and 110 is 0.1169
```

Description

Function `ANORDF` evaluates the distribution function, Φ, of a standard normal (Gaussian) random variable, that is,

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt$$

The value of the distribution function at the point x is the probability that the random variable takes a value less than or equal to x.

The standard normal distribution (for which ANORDF is the distribution function) has mean of 0 and variance of 1. The probability that a normal random variable with mean and variance σ^2 is less than y is given by ANORDF evaluated at $(y - \mu)/\sigma$.

$\Phi(x)$ is evaluated by use of the complementary error function, erfc. (See ERFC, IMSL MATH/LIBRARY Special Functions). The relationship is:

$$\Phi(x) = \text{erfc}(-x/\sqrt{2.0}) / 2$$

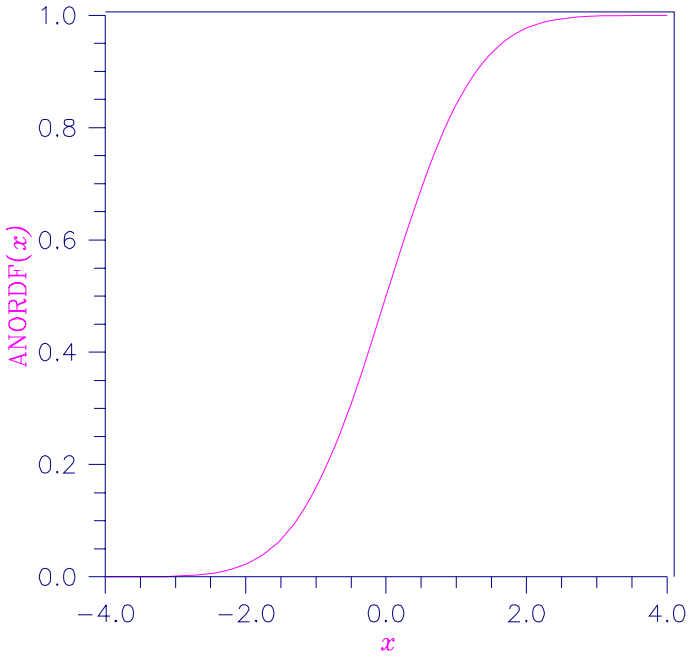


Figure 17-6 Standard Normal Distribution Function

ANORIN

This function evaluates the inverse of the standard normal (Gaussian) distribution function.

Function Return Value

ANORIN — Function value. (Output)
The probability that a standard normal random variable takes a value less than or equal to ANORIN is P.

Required Arguments

P — Probability for which the inverse of the normal distribution function is to be evaluated. (Input)
P must be in the open interval (0.0, 1.0).

FORTRAN 90 Interface

Generic: `ANORIN(P)`

Specific: The specific interface names are `S_ANORIN` and `D_ANORIN`.

FORTRAN 77 Interface

Single: `ANORIN(P)`

Double: The double precision name is `DNORIN`.

Example

In this example, we compute the point such that the probability is 0.9 that a standard normal random variable is less than or equal to this point.

```
      USE UMACH_INT
      USE ANORIN_INT
      INTEGER    NOUT
      REAL       P, X
!
      CALL UMACH (2, NOUT)
      P = 0.9
      X = ANORIN(P)
      WRITE (NOUT,99999) X
99999 FORMAT (' The 90th percentile of a standard normal is ', F6.4)
      END
```

Output

```
The 90th percentile of a standard normal is 1.2816
```

Description

Function `ANORIN` evaluates the inverse of the distribution function, Φ, of a standard normal (Gaussian) random variable, that is, `ANORIN(P)` = $\Phi^{-1}(p)$, where

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt$$

The value of the distribution function at the point x is the probability that the random variable takes a value less than or equal to x. The standard normal distribution has a mean of 0 and a variance of 1.

BETDF

This function evaluates the beta probability distribution function.

Function Return Value

BETDF — Probability that a random variable from a beta distribution having parameters PIN and QIN will be less than or equal to X. (Output)

Required Arguments

X — Argument for which the beta distribution function is to be evaluated. (Input)

PIN — First beta distribution parameter. (Input)
PIN must be positive.

QIN — Second beta distribution parameter. (Input)
QIN must be positive.

FORTRAN 90 Interface

Generic: `BETDF(X, PIN, QIN)`

Specific: The specific interface names are `S_BETDF` and `D_BETDF`.

FORTRAN 77 Interface

Single: `BETDF(X, PIN, QIN)`

Double: The double precision name is `DBETDF`.

Example

Suppose *X* is a beta random variable with parameters 12 and 12. (*X* has a symmetric distribution.) In this example, we find the probability that *X* is less than 0.6 and the probability that *X* is between 0.5 and 0.6. (Since *X* is a symmetric beta random variable, the probability that it is less than 0.5 is 0.5.)

```
      USE UMACH_INT
      USE BETDF_INT
      INTEGER    NOUT
      REAL       P, PIN, QIN, X
!
      CALL UMACH (2, NOUT)
      PIN = 12.0
      QIN = 12.0
      X   = 0.6
      P   = BETDF(X,PIN,QIN)
      WRITE (NOUT,99998) P
99998 FORMAT (' The probability that X is less than 0.6 is ', F6.4)
      X = 0.5
      P = P - BETDF(X,PIN,QIN)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that X is between 0.5 and 0.6 is ', &
```

```
      F6.4)
END
```

Output

```
The probability that X is less than 0.6 is 0.8364
The probability that X is between 0.5 and 0.6 is 0.3364
```

Comments

Informational errors

Type	Code	
1	1	Since the input argument X is less than or equal to zero, the distribution function is equal to zero at X.
1	2	Since the input argument X is greater than or equal to one, the distribution function is equal to one at X.

Description

Function BETDF evaluates the distribution function of a beta random variable with parameters PIN and QIN. This function is sometimes called the *incomplete beta ratio* and, with p = PIN and q = QIN, is denoted by $I_x(p, q)$. It is given by

$$I_x(p,q) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} \int_o^x t^{p-1}(1-t)^{q-1} dt$$

where $\Gamma(\cdot)$ is the gamma function. The value of the distribution function $I_x(p, q)$ is the probability that the random variable takes a value less than or equal to x.

The integral in the expression above is called the *incomplete beta function* and is denoted by $\beta_x(p, q)$. The constant in the expression is the reciprocal of the *beta function* (the incomplete function evaluated at one) and is denoted by $\beta(p, q)$.

Function BETDF uses the method of Bosten and Battiste (1974).

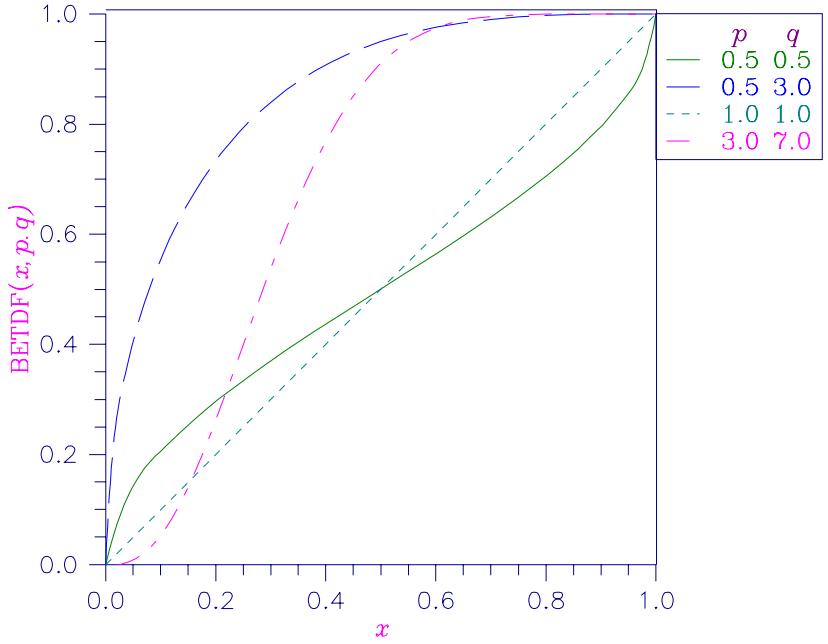


Figure 17-7 Beta Distribution Function

BETIN

This function evaluates the inverse of the beta distribution function.

Function Return Value

BETIN — Function value. (Output)
The probability that a beta random variable takes a value less than or equal to `BETIN` is `P`.

Required Arguments

P — Probability for which the inverse of the beta distribution function is to be evaluated. (Input)
`P` must be in the open interval (0.0, 1.0).

PIN — First beta distribution parameter. (Input)
`PIN` must be positive.

QIN — Second beta distribution parameter. (Input)
`QIN` must be positive.

FORTRAN 90 Interface

Generic: `BETIN(P, PIN, QIN)`

Specific: The specific interface names are `S_BETIN` and `D_BETIN`.

FORTRAN 77 Interface

Single: `BETIN(P, PIN, QIN)`

Double: The double precision name is `DBETIN`.

Example

Suppose X is a beta random variable with parameters 12 and 12. (X has a symmetric distribution.) In this example, we find the value x_0 such that the probability that $X \le x_0$ is 0.9.

```
      USE UMACH_INT
      USE BETIN_INT
      INTEGER    NOUT
      REAL       P, PIN, QIN, X
!
      CALL UMACH (2, NOUT)
      PIN = 12.0
      QIN = 12.0
      P   = 0.9
      X   = BETIN(P,PIN,QIN)
      WRITE (NOUT,99999) X
99999 FORMAT (' X is less than ', F6.4, ' with probability 0.9.')
      END
```

Output

```
X is less than 0.6299 with probability 0.9.
```

Comments

Informational error

Type	Code	
3	1	The value for the inverse Beta distribution could not be found in 100 iterations. The best approximation is used.

Description

The function `BETIN` evaluates the inverse distribution function of a beta random variable with parameters `PIN` and `QIN`, that is, with P = `P`, p = `PIN`, and q = `QIN`, it determines x (equal to `BETIN`(`P`, `PIN`, `QIN`)), such that

$$P = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} \int_o^x t^{p-1}(1-t)^{q-1} dt$$

where $\Gamma(\cdot)$ is the gamma function. The probability that the random variable takes a value less than or equal to x is P .

BNRDF

This function evaluates the bivariate normal distribution function.

Function Return Value

BNRDF — Function value, the probability that a bivariate normal random variable with correlation `RHO` takes a value less than or equal to `X` and less than or equal to `Y`. (Output)

Required Arguments

X — One argument for which the bivariate normal distribution function is to be evaluated. (Input)

Y — The other argument for which the bivariate normal distribution function is to be evaluated. (Input)

RHO — Correlation coefficient. (Input)

FORTRAN 90 Interface

Generic: `BNRDF(X, Y, RHO)`

Specific: The specific interface names are `S_BNRDF` and `D_BNRDF`.

FORTRAN 77 Interface

Single: `BNRDF(X, Y, RHO)`

Double: The double precision name is `DBNRDF`.

Example

Suppose (X, Y) is a bivariate normal random variable with mean (0, 0) and variance-covariance matrix

$$\begin{pmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{pmatrix}$$

In this example, we find the probability that X is less than −2.0 and Y is less than 0.0.

```
      USE BNRDF_INT
      USE UMACH_INT
      INTEGER    NOUT
      REAL       P, RHO, X, Y
!
      CALL UMACH (2, NOUT)
      X   = -2.0
      Y   = 0.0
      RHO = 0.9
      P   = BNRDF(X,Y,RHO)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that X is less than -2.0  and Y ', &
             'is less than 0.0 is ', F6.4)
      END
```

Output

```
The probability that X is less than −2.0 and Y is less than 0.0 is 0.0228
```

Description

Function BNRDF evaluates the distribution function F of a bivariate normal distribution with means of zero, variances of one, and correlation of RHO; that is, with ρ = RHO, and $|\rho| < 1$,

$$F(x,y) = \frac{1}{2\pi\sqrt{1-\rho^2}} \int_{-\infty}^{x} \int_{-\infty}^{y} \exp\left(-\frac{u^2 - 2\rho uv + v^2}{2(1-\rho^2)}\right) du\ dv$$

To determine the probability that $U \le u_0$ and $V \le v_0$, where $(U, V)^T$ is a bivariate normal random variable with mean $\mu = (\mu_U, \mu_V)^T$ and variance-covariance matrix

$$\Sigma = \begin{pmatrix} \sigma_U^2 & \sigma_{UV} \\ \sigma_{UV} & \sigma_V^2 \end{pmatrix}$$

transform $(U, V)^T$ to a vector with zero means and unit variances. The input to BNRDF would be X = $(u_0 - \mu_U)/\sigma_U$, Y = $(v_0 - \mu_V)/\sigma_V$, and $\rho = \sigma_{UV}/(\sigma_U\sigma_V)$.

Function BNRDF uses the method of Owen (1962, 1965). Computation of Owen's T-function is based on code by M. Patefield and D. Tandy (2000). For $|\rho| = 1$, the distribution function is computed based on the univariate statistic, $Z = \min(x, y)$, and on the normal distribution function ANORDF (page 1262).

CHIDF

This function evaluates the chi-squared distribution function.

Function Return Value

CHIDF — Function value, the probability that a chi-squared random variable takes a value less than or equal to CHSQ. (Output)

Required Arguments

CHSQ — Argument for which the chi-squared distribution function is to be evaluated. (Input)

DF — Number of degrees of freedom of the chi-squared distribution. (Input)
`DF` must be greater than or equal to 0.5.

FORTRAN 90 Interface

Generic: `CHIDF(CHSQ, DF)`

Specific: The specific interface names are `S_CHIDF` and `D_CHIDF`.

FORTRAN 77 Interface

Single: `CHIDF(CHSQ, DF)`

Double: The double precision name is `DCHIDF`.

Example

Suppose X is a chi-squared random variable with 2 degrees of freedom. In this example, we find the probability that X is less than 0.15 and the probability that X is greater than 3.0.

```
      USE UMACH_INT
      USE CHIDF_INT
      INTEGER    NOUT
      REAL       CHSQ, DF, P
!
      CALL UMACH (2, NOUT)
      DF   = 2.0
      CHSQ = 0.15
      P    = CHIDF(CHSQ,DF)
      WRITE (NOUT,99998) P
99998 FORMAT (' The probability that chi-squared with 2 df is less ', &
             'than 0.15 is ', F6.4)
      CHSQ = 3.0
      P    = 1.0 - CHIDF(CHSQ,DF)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that chi-squared with 2 df is greater ' &
            , 'than 3.0 is ', F6.4)
      END
```

Output

```
The probability that chi-squared with 2 df is less than 0.15 is 0.0723
The probability that chi-squared with 2 df is greater than 3.0 is 0.2231
```

Comments

Informational errors

Type	Code	
1	1	Since the input argument, CHSQ, is less than zero, the distribution function is zero at CHSQ.
2	3	The normal distribution is used for large degrees of freedom. However, it has produced underflow. Therefore, the probability, CHIDF, is set to zero.

Description

Function CHIDF evaluates the distribution function, F, of a chi-squared random variable with DF degrees of freedom, that is, with v = DF, and x = CHSQ,

$$F(x) = \frac{1}{2^{v/2}\Gamma(v/2)} \int_o^x e^{-t/2} t^{v/2-1} dt$$

where $\Gamma(\cdot)$ is the gamma function. The value of the distribution function at the point x is the probability that the random variable takes a value less than or equal to x.

For $v > 65$, CHIDF uses the Wilson-Hilferty approximation (Abramowitz and Stegun 1964, equation 26.4.17) to the normal distribution, and routine ANORDF (page 1262) is used to evaluate the normal distribution function.

For $v \le 65$, CHIDF uses series expansions to evaluate the distribution function. If $x < \max(v/2, 26)$, CHIDF uses the series 6.5.29 in Abramowitz and Stegun (1964), otherwise, it uses the asymptotic expansion 6.5.32 in Abramowitz and Stegun.

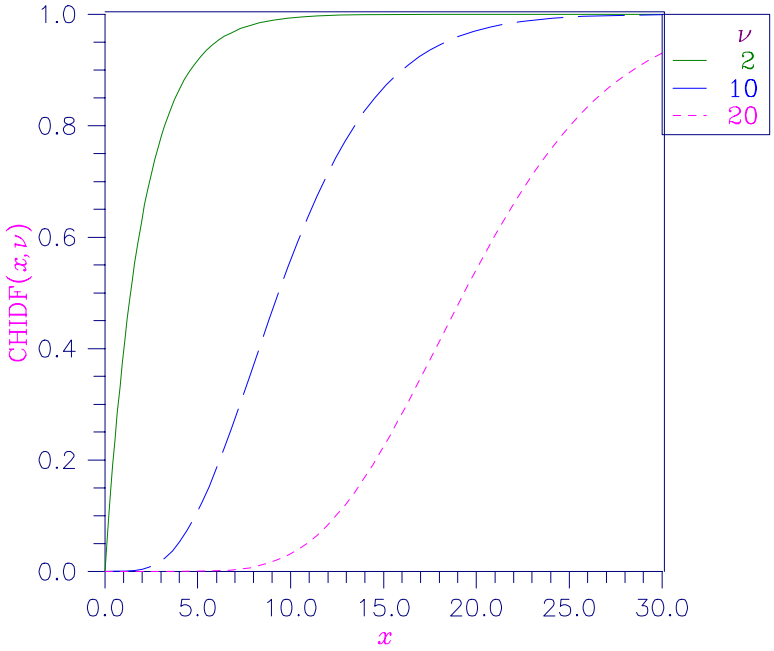


Figure 17-8 Chi-Squared Distribution Function

CHIIN

This function evaluates the inverse of the chi-squared distribution function.

Function Return Value

CHIIN — Function value. (Output)
The probability that a chi-squared random variable takes a value less than or equal to CHIIN is P.

Required Arguments

P — Probability for which the inverse of the chi-squared distribution function is to be evaluated. (Input)
P must be in the open interval (0.0, 1.0).

DF — Number of degrees of freedom of the chi-squared distribution. (Input)
DF must be greater than or equal to 0.5.

FORTRAN 90 Interface

Generic: `CHIIN (P, DF)`

Specific: The specific interface names are `S_CHIIN` and `D_CHIIN`.

FORTRAN 77 Interface

Single: `CHIIN(P, DF)`

Double: The double precision name is `DCHIIN`.

Example

In this example, we find the 99-th percentage points of a chi-squared random variable with 2 degrees of freedom and of one with 64 degrees of freedom.

```
      USE UMACH_INT
      USE CHIIN_INT
      INTEGER    NOUT
      REAL       DF, P, X
!
      CALL UMACH (2, NOUT)
      P  = 0.99
      DF = 2.0
      X  = CHIIN(P,DF)
      WRITE (NOUT,99998) X
99998 FORMAT (' The 99-th percentage point of chi-squared with  2 df ' &
             , 'is ', F7.3)
      DF = 64.0
```

```
      X  = CHIIN(P,DF)
      WRITE (NOUT,99999) X
99999 FORMAT (' The 99-th percentage point of chi-squared with 64 df ' &
            , 'is ', F7.3)
      END
```

Output

```
The 99-th percentage point of chi-squared with  2 df is   9.210
The 99-th percentage point of chi-squared with 64 df is  93.217
```

Comments

Informational errors

Type	Code	
4	1	Over 100 iterations have occurred without convergence. Convergence is assumed.

Description

Function CHIIN evaluates the inverse distribution function of a chi-squared random variable with DF degrees of freedom, that is, with P = P and v = DF, it determines x (equal to CHIIN(P, DF)), such that

$$P = \frac{1}{2^{v/2}\Gamma(v/2)} \int_o^x e^{-t/2} t^{v/2-1} dt$$

where $\Gamma(\cdot)$ is the gamma function. The probability that the random variable takes a value less than or equal to x is P.

For $v < 40$, CHIIN uses bisection (if $v \le 2$ or $P > 0.98$) or regula falsi to find the point at which the chi-squared distribution function is equal to P. The distribution function is evaluated using routine CHIDF (page 1271).

For $40 \le v < 100$, a modified Wilson-Hilferty approximation (Abramowitz and Stegun 1964, equation 26.4.18) to the normal distribution is used, and routine ANORIN (page 1264) is used to evaluate the inverse of the normal distribution function. For $v \ge 100$, the ordinary Wilson-Hilferty approximation (Abramowitz and Stegun 1964, equation 26.4.17) is used.

CSNDF

This function evaluates the noncentral chi-squared distribution function.

Function Return Value

CSNDF — Function value, the probability that a noncentral chi-squared random variable takes a value less than or equal to CHSQ. (Output)

Required Arguments

CHSQ — Argument for which the noncentral chi-squared distribution function is to be evaluated. (Input)

DF —Number of degrees of freedom of the noncentral chi-squared distribution. (Input)
`DF` must be greater than or equal to 0.5 and less than or equal to 200,000.

ALAM — The noncentrality parameter. (Input)
`ALAM` must be nonnegative, and `ALAM` + `DF` must be less than or equal to 200,000.

FORTRAN 90 Interface

Generic: `CSNDF(CHSQ, DF, ALAM)`

Specific: The specific interface names are `S_CSNDF` and `D_CSNDF`.

FORTRAN 77 Interface

Single: `CSNDF(CHSQ, DF, ALAM)`

Double: The double precision name is `DCSNDF`.

Example

In this example, `CSNDF` is used to compute the probability that a random variable that follows the noncentral chi-squared distribution with noncentrality parameter of 1 and with 2 degrees of freedom is less than or equal to 8.642.

```
      USE UMACH_INT
      USE CSNDF_INT
      INTEGER    NOUT
      REAL       ALAM, CHSQ, DF, P
!
      CALL UMACH (2, NOUT)
      DF   = 2.0
      ALAM = 1.0
      CHSQ = 8.642
      P    = CSNDF(CHSQ,DF,ALAM)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that a noncentral chi-squared random', &
             /, ' variable with 2 df and noncentrality 1.0 is less', &
             /, ' than 8.642 is ', F5.3)
      END
```

Output

```
The probability that a noncentral chi-squared random
variable with 2 df and noncentrality 1.0 is less
than 8.642 is 0.950
```

Comments

1. Informational errors

Type	Code	
1	1	Since the input argument, CHSQ, is less than or equal to zero, the distribution function is zero at CHSQ.
3	2	Convergence was not obtained. The best approximation to the probability is returned.

2. This subroutine sums terms of an infinite series of central chi-squared distribution functions weighted by Poisson terms. Summing terminates when either the current term is less than 10 * AMACH(4) times the current sum or when 1000 terms have been accumulated. In the latter case, a warning error is issued.

Description

Function CSNDF evaluates the distribution function of a noncentral chi-squared random variable with DF degrees of freedom and noncentrality parameter ALAM, that is, with v = DF, λ = ALAM, and x = CHSQ,

$$\text{CSNDF}(x) = \sum_{i=0}^{\infty} \frac{e^{-\lambda/2}(\lambda/2)^i}{i!} \int_0^x \frac{t^{(v+2i)/2-1}e^{-t/2}}{2^{v+2i)/2}\Gamma(\frac{v+2i}{2})} dt$$

where $\Gamma(\cdot)$ is the gamma function. This is a series of central chi-squared distribution functions with Poisson weights. The value of the distribution function at the point x is the probability that the random variable takes a value less than or equal to x.

The noncentral chi-squared random variable can be defined by the distribution function above, or alternatively and equivalently, as the sum of squares of independent normal random variables. If Y_i have independent normal distributions with means μ_i and variances equal to one and

$$X = \sum_{i=1}^{n} Y_i^2$$

then X has a noncentral chi-squared distribution with n degrees of freedom and noncentrality parameter equal to

$$\sum_{i=1}^{n} \mu_i^2$$

With a noncentrality parameter of zero, the noncentral chi-squared distribution is the same as the chi-squared distribution.

Function CSNDF determines the point at which the Poisson weight is greatest, and then sums forward and backward from that point, terminating when the additional terms are sufficiently small or when a maximum of 1000 terms have been accumulated. The recurrence relation 26.4.8 of Abramowitz and Stegun (1964) is used to speed the evaluation of the central chi-squared distribution functions.

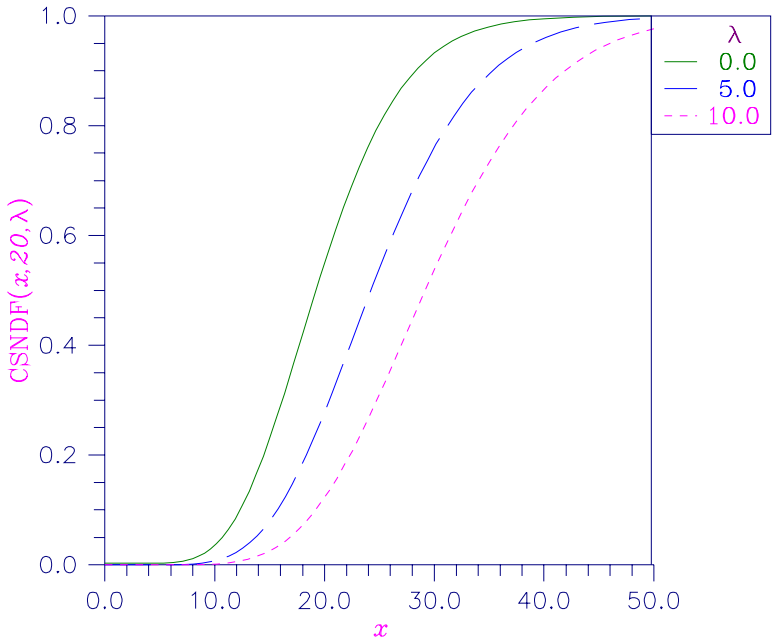


Figure 17-9 Noncentral Chi-squared Distribution Function

CSNIN

This function evaluates the inverse of the noncentral chi-squared function.

Function Return Value

CSNIN — Function value. (Output)
The probability that a noncentral chi-squared random variable takes a value less than or equal to `CSNIN` is `P`.

Required Arguments

P — Probability for which the inverse of the noncentral chi-squared distribution function is to be evaluated. (Input)
`P` must be in the open interval (0.0, 1.0).

DF — Number of degrees of freedom of the noncentral chi-squared distribution. (Input)
`DF` must be greater than or equal to 0.5 and less than or equal to 200,000.

ALAM — The noncentrality parameter. (Input)
`ALAM` must be nonnegative, and `ALAM` + `DF` must be less than or equal to 200,000.

FORTRAN 90 Interface

Generic: `CSNIN(P, DF, ALAM)`

Specific: The specific interface names are `S_CSNIN` and `D_CSNIN`.

FORTRAN 77 Interface

Single: `CSNIN(P, DF, ALAM)`

Double: The double precision name is `DCSNIN`.

Example

In this example, we find the 95-th percentage point for a noncentral chi-squared random variable with 2 degrees of freedom and noncentrality parameter 1.

```
      USE CSNIN_INT
      USE UMACH_INT
      INTEGER    NOUT
      REAL       ALAM, CHSQ, DF, P
!
      CALL UMACH (2, NOUT)
      DF   = 2.0
      ALAM = 1.0
      P    = 0.95
      CHSQ = CSNIN(P,DF,ALAM)
      WRITE (NOUT,99999) CHSQ
!
99999 FORMAT (' The 0.05 noncentral chi-squared critical value is ', &
             F6.3, '.')
!
      END
```

Output

```
The 0.05 noncentral chi-squared critical value is  8.642.
```

Comments

Informational error

Type	Code	
4	1	Over 100 iterations have occurred without convergence. Convergence is assumed.

Description

Function CSNIN evaluates the inverse distribution function of a noncentral chi-squared random variable with DF degrees of freedom and noncentrality parameter ALAM; that is, with P = P, ν = DF, and = λ = ALAM, it determines c_0 (= CSNIN(P, DF, ALAM)), such that

$$P = \sum_{i=0}^{\infty} \frac{e^{-\lambda/2}(\lambda/2)^i}{i!} \int_0^{c_0} \frac{x^{(\nu+2i)/2-1} e^{-x/2}}{2^{(\nu+2i)/2}\,\Gamma(\frac{\nu+2i}{2})} dx$$

where $\Gamma(\cdot)$ is the gamma function. The probability that the random variable takes a value less than or equal to c_0 is P .

Function CSNIN uses bisection and modified regula falsi to invert the distribution function, which is evaluated using routine CSNDF (page 1275). See CSNDF for an alternative definition of the noncentral chi-squared random variable in terms of normal random variables.

FDF

This function evaluates the F distribution function.

Function Return Value

FDF — Function value, the probability that an F random variable takes a value less than or equal to the input F. (Output)

Required Arguments

F — Argument for which the F distribution function is to be evaluated. (Input)

DFN — Numerator degrees of freedom. (Input)
DFN must be positive.

DFD — Denominator degrees of freedom. (Input)
DFD must be positive.

FORTRAN 90 Interface

Generic: FDF(F, DFN, DFD)

Specific: The specific interface names are S_FDF and D_FDF.

FORTRAN 77 Interface

Single: FDF(F, DFN, DFD)

Double: The double precision name is DFDF.

Example

In this example, we find the probability that an F random variable with one numerator and one denominator degree of freedom is greater than 648.

```
      USE UMACH_INT
      USE FDF_INT
      INTEGER    NOUT
      REAL       DFD, DFN, F, P
!
      CALL UMACH (2, NOUT)
      F   = 648.0
      DFN = 1.0
      DFD = 1.0
      P   = 1.0 - FDF(F,DFN,DFD)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that an F(1,1) variate is greater ', &
             'than 648 is ', F6.4)
      END
```

Output

```
The probability that an F(1, 1) variate is greater than 648 is 0.0250
```

Comments

Informational error

Type	Code	
1	3	Since the input argument F is not positive, the distribution function is zero at F.

Description

Function FDF evaluates the distribution function of a Snedecor's F random variable with DFN numerator degrees of freedom and DFD denominator degrees of freedom. The function is evaluated by making a transformation to a beta random variable and then using the routine BETDF (page 1265). If X is an F variate with ν_1 and ν_2 degrees of freedom and $Y = \nu_1 X/(\nu_2 + \nu_1 X)$, then Y is a beta variate with parameters $p = \nu_1/2$ and $q = \nu_2/2$. The function FDF also uses a relationship between F random variables that can be expressed as follows.

```
FDF(X, DFN, DFD) = 1.0 – FDF(1.0/X, DFD, DFN)
```

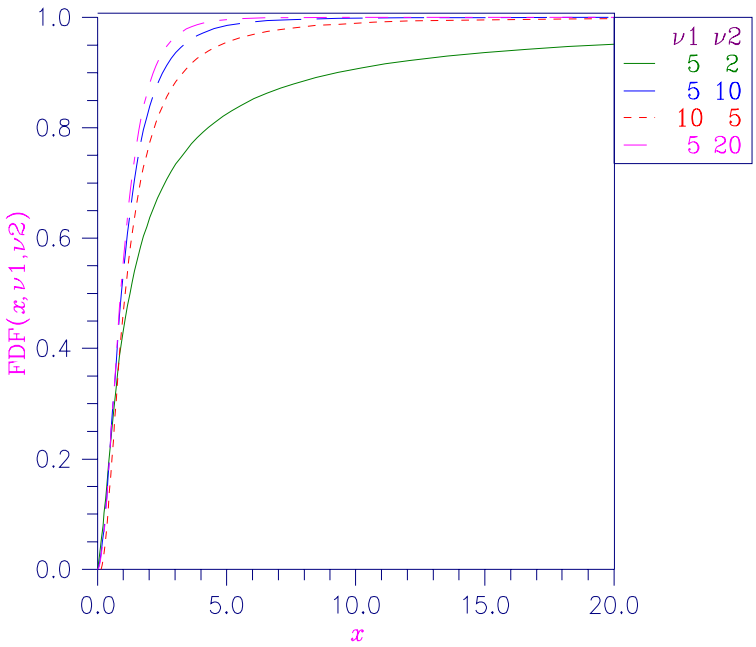


Figure 17-10 *F* Distribution Function

FIN

This function evaluates the inverse of the *F* distribution function.

Function Return Value

FIN — Function value. (Output)
The probability that an *F* random variable takes a value less than or equal to `FIN` is `P`.

Required Arguments

P — Probability for which the inverse of the *F* distribution function is to be evaluated. (Input)
`P` must be in the open interval (0.0, 1.0).

DFN — Numerator degrees of freedom. (Input)
`DFN` must be positive.

DFD — Denominator degrees of freedom. (Input)
`DFD` must be positive.

FORTRAN 90 Interface

Generic: `FIN(P, DFN, DFD)`

Specific: The specific interface names are `S_FDF` and `D_FDF`.

FORTRAN 77 Interface

Single: `FIN(P, DFN, DFD)`

Double: The double precision name is `DFDF`.

Example

In this example, we find the 99-th percentage point for an F random variable with 1 and 7 degrees of freedom.

```
      USE UMACH_INT
      USE FIN_INT
      INTEGER    NOUT
      REAL       DFD, DFN, F, P
!
      CALL UMACH (2, NOUT)
      P   = 0.99
      DFN = 1.0
      DFD = 7.0
      F   = FIN(P,DFN,DFD)
      WRITE (NOUT,99999) F
99999 FORMAT (' The F(1,7) 0.01 critical value is ', F6.3)
      END
```

Output

```
The F(1, 7) 0.01 critical value is 12.246
```

Comments

Informational error

Type	Code	
4	4	`FIN` is set to machine infinity since overflow would occur upon modifying the inverse value for the F distribution with the result obtained from the inverse beta distribution.

Description

Function `FIN` evaluates the inverse distribution function of a Snedecor's F random variable with `DFN` numerator degrees of freedom and `DFD` denominator degrees of freedom. The function is evaluated by making a transformation to a beta random variable and then using the routine `BETIN` (page 1268). If X is an F variate with v_1 and v_2 degrees of freedom and $Y = v_1X/(v_2 +$

$v_1 X$), then Y is a beta variate with parameters $p = v_1/2$ and $q = v_2/2$. If $P \leq 0.5$, FIN uses this relationship directly, otherwise, it also uses a relationship between F random variables that can be expressed as follows, using routine FDF (page 1280), which is the F cumulative distribution function:

```
FDF(F, DFN, DFD) = 1.0 – FDF(1.0/F, DFD, DFN).
```

GAMDF

This function evaluates the gamma distribution function.

Function Return Value

GAMDF — Function value, the probability that a gamma random variable takes a value less than or equal to X. (Output)

Required Arguments

X — Argument for which the gamma distribution function is to be evaluated. (Input)

A — The shape parameter of the gamma distribution. (Input)
This parameter must be positive.

FORTRAN 90 Interface

Generic: `GAMDF(X, A)`

Specific: The specific interface names are `S_GAMDF` and `D_GAMDF`.

FORTRAN 77 Interface

Single: `GAMDF(X, A)`

Double: The double precision name is `DGAMDF`.

Example

Suppose X is a gamma random variable with a shape parameter of 4. (In this case, it has an *Erlang distribution* since the shape parameter is an integer.) In this example, we find the probability that X is less than 0.5 and the probability that X is between 0.5 and 1.0.

```
      USE UMACH_INT
      USE GAMDF_INT
      INTEGER    NOUT
      REAL       A, P, X
!
      CALL UMACH (2, NOUT)
      A = 4.0
      X = 0.5
```

```
      P = GAMDF(X,A)
      WRITE (NOUT,99998) P
99998 FORMAT (' The probability that X is less than 0.5 is ', F6.4)
      X = 1.0
      P = GAMDF(X,A) - P
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that X is between 0.5 and 1.0 is ', &
            F6.4)
      END
```

Output

```
The probability that X is less than 0.5 is 0.0018
The probability that X is between 0.5 and 1.0 is 0.0172
```

Comments

Informational error

Type	Code	
1	2	Since the input argument X is less than zero, the distribution function is set to zero.

Description

Function GAMDF evaluates the distribution function, F, of a gamma random variable with shape parameter a; that is,

$$F(x) = \frac{1}{\Gamma(a)} \int_0^x e^{-t} t^{a-1} dt$$

where $\Gamma(\cdot)$ is the gamma function. (The gamma function is the integral from 0 to ∞ of the same integrand as above). The value of the distribution function at the point x is the probability that the random variable takes a value less than or equal to x.

The gamma distribution is often defined as a two-parameter distribution with a scale parameter b (which must be positive), or even as a three-parameter distribution in which the third parameter c is a location parameter. In the most general case, the probability density function over (c, ∞) is

$$f(t) = \frac{1}{b^a \Gamma(a)} e^{-(t-c)/b} (x-c)^{a-1}$$

If T is such a random variable with parameters a, b, and c, the probability that $T \le t_0$ can be obtained from GAMDF by setting X = $(t_0 - c)/b$.

If X is less than a or if X is less than or equal to 1.0, GAMDF uses a series expansion. Otherwise, a continued fraction expansion is used. (See Abramowitz and Stegun, 1964.)

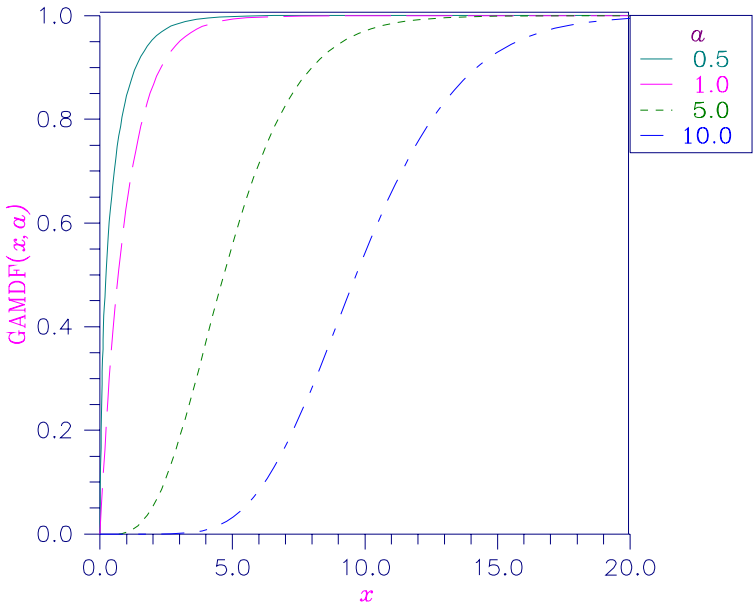


Figure 17-11 Gamma Distribution Function

GAMIN

This function evaluates the inverse of the gamma distribution function.

Function Return Value

GAMIN — Function value. (Output)
The probability that a gamma random variable takes a value less than or equal to `GAMIN` is `P`.

Required Arguments

P — Probability for which the inverse of the gamma distribution function is to be evaluated. (Input)
`P` must be in the open interval (0.0, 1.0).

A — The shape parameter of the gamma distribution. (Input)
This parameter must be positive.

FORTRAN 90 Interface

Generic: `GAMIN (P, A)`

Specific: The specific interface names are `S_GAMIN` and `D_GAMIN`.

FORTRAN 77 Interface

Single: `GAMIN(P, A)`

Double: The double precision name is `DGAMIN`.

Example

In this example, we find the 95-th percentage point for a gamma random variable with shape parameter of 4.

```
      USE UMACH_INT
      USE GAMIN_INT
      INTEGER    NOUT
      REAL       A, P, X
!
      CALL UMACH (2, NOUT)
      A = 4.0
      P = 0.95
      X = GAMIN(P,A)
      WRITE (NOUT,99999) X
!
99999 FORMAT (' The 0.05 gamma(4) critical value is ', F6.3, &
             '.')
!
      END
```

Output

```
The 0.05 gamma(4) critical value is 7.754.
```

Comments

Informational error

Type	Code	
4	1	Over 100 iterations have occurred without convergence. Convergence is assumed.

Description

Function `GAMIN` evaluates the inverse distribution function of a gamma random variable with shape parameter a, that is, it determines x (= `GAMIN(P, A)`), such that

$$P = \frac{1}{\Gamma(a)} \int_0^x e^{-t} t^{a-1} dt$$

where $\Gamma(\cdot)$ is the gamma function. The probability that the random variable takes a value less than or equal to x is P. See the documentation for routine `GAMDF` (page 1284) for further discussion of the gamma distribution.

Function GAMIN uses bisection and modified regula falsi to invert the distribution function, which is evaluated using routine GAMDF.

TDF

This function evaluates the Student's *t* distribution function.

Function Return Value

TDF — Function value, the probability that a Student's *t* random variable takes a value less than or equal to the input T. (Output)

Required Arguments

T — Argument for which the Student's *t* distribution function is to be evaluated. (Input)

DF — Degrees of freedom. (Input)
DF must be greater than or equal to 1.0.

FORTRAN 90 Interface

Generic: TDF(T, DF)

Specific: The specific interface names are S_TDF and D_TDF.

FORTRAN 77 Interface

Single: TDF(T, DF)

Double: The double precision name is DTDF.

Example

In this example, we find the probability that a *t* random variable with 6 degrees of freedom is greater in absolute value than 2.447. We use the fact that *t* is symmetric about 0.

```
      USE TDF_INT
      USE UMACH_INT
      INTEGER    NOUT
      REAL       DF, P, T
!
      CALL UMACH (2, NOUT)
      T  = 2.447
      DF = 6.0
      P  = 2.0*TDF(-T,DF)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that a t(6) variate is greater ', &
             'than 2.447 in', /, ' absolute value is ', F6.4)
      END
```

Output

```
The probability that a t(6) variate is greater than 2.447 in absolute value
is 0.0500
```

Description

Function TDF evaluates the distribution function of a Student's *t* random variable with DF degrees of freedom. If the square of T is greater than or equal to DF, the relationship of a *t* to an *F* random variable (and subsequently, to a beta random variable) is exploited, and routine BETDF (page 1265) is used. Otherwise, the method described by Hill (1970) is used. If DF is not an integer, if DF is greater than 19, or if DF is greater than 200, a Cornish-Fisher expansion is used to evaluate the distribution function. If DF is less than 20 and ABS(T) is less than 2.0, a trigonometric series (see Abramowitz and Stegun 1964, equations 26.7.3 and 26.7.4, with some rearrangement) is used. For the remaining cases, a series given by Hill (1970) that converges well for large values of T is used.

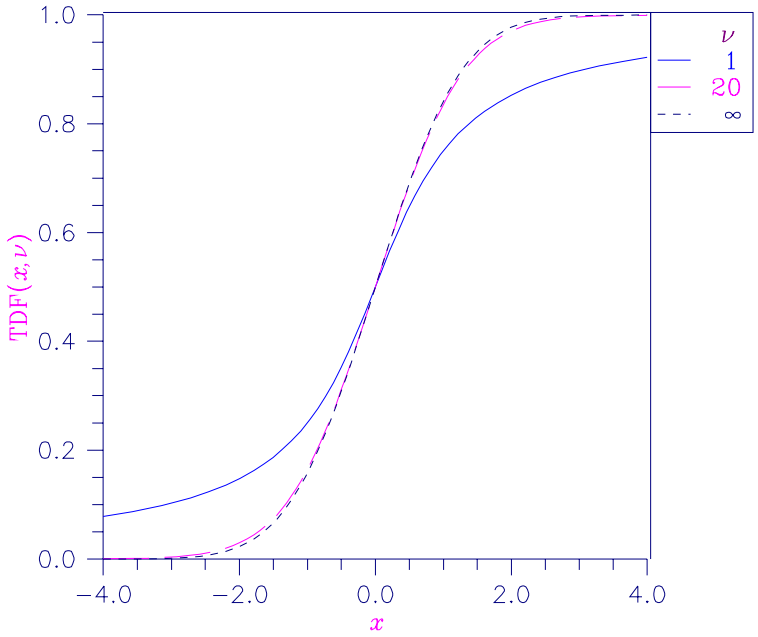


Figure 17-12 Student's *t* Distribution Function

TIN

This function evaluates the inverse of the Student's *t* distribution function.

Function Return Value

TIN — Function value. (Output)
The probability that a Student's *t* random variable takes a value less than or equal to `TIN` is `P`.

Required Arguments

P — Probability for which the inverse of the Student's *t* distribution function is to be evaluated. (Input)
`P` must be in the open interval (0.0, 1.0).

DF — Degrees of freedom. (Input)
`DF` must be greater than or equal to 1.0.

FORTRAN 90 Interface

Generic: `TIN(P, DF)`

Specific: The specific interface names are `S_TIN` and `D_TIN`.

FORTRAN 77 Interface

Single: `TIN(P, DF)`

Double: The double precision name is `DTIN`.

Example

In this example, we find the 0.05 critical value for a two-sided *t* test with 6 degrees of freedom.

```
      USE TIN_INT
      USE UMACH_INT
      INTEGER    NOUT
      REAL       DF, P, T
!
      CALL UMACH (2, NOUT)
      P  = 0.975
      DF = 6.0
      T  = TIN(P,DF)
      WRITE (NOUT,99999) T
99999 FORMAT (' The two-sided t(6) 0.05 critical value is ', F6.3)
      END
```

Output

```
The two-sided t(6) 0.05 critical value is  2.447
```

Comments

Informational error

Type	Code	
4	3	TIN is set to machine infinity since overflow would occur upon modifying the inverse value for the *F* distribution with the result obtained from the inverse β distribution.

Description

Function TIN evaluates the inverse distribution function of a Student's *t* random variable with DF degrees of freedom. Let v = DF. If v equals 1 or 2, the inverse can be obtained in closed form, if v is between 1 and 2, the relationship of a *t* to a beta random variable is exploited and routine BETIN (page 1268) is used to evaluate the inverse; otherwise the algorithm of Hill (1970) is used. For small values of v greater than 2, Hill's algorithm inverts an integrated expansion in $1/(1 + t^2/v)$ of the *t* density. For larger values, an asymptotic inverse Cornish-Fisher type expansion about normal deviates is used.

TNDF

This function evaluates the noncentral Student's *t* distribution function.

Function Return Value

TNDF — Function value, the probability that a noncentral Student's *t* random variable takes a value less than or equal to T. (Output)

Required Arguments

T — Argument for which the noncentral Student's *t* distribution function is to be evaluated. (Input)

IDF — Number of degrees of freedom of the noncentral Student's *t* distribution. (Input) IDF must be positive.

DELTA — The noncentrality parameter. (Input)

FORTRAN 90 Interface

Generic: `TNDF(T, IDF, DELTA)`

Specific: The specific interface names are `S_TNDF` and `D_TNDF`.

FORTRAN 77 Interface

Single: `TNDF(T, IDF, DELTA)`

Double: The double precision name is `DTNDF`.

Example

Suppose T is a noncentral t random variable with 6 degrees of freedom and noncentrality parameter 6. In this example, we find the probability that T is less than 12.0. (This can be checked using the table on page 111 of Owen 1962, with $\eta = 0.866$, which yields $\lambda = 1.664$.)

```
      USE UMACH_INT
      USE TNDF_INT
      INTEGER    IDF, NOUT
      REAL       DELTA, P, T
!
      CALL UMACH (2, NOUT)
      IDF   = 6
      DELTA = 6.0
      T     = 12.0
      P     = TNDF(T,IDF,DELTA)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that T is less than 12.0 is ', F6.4)
      END
```

Output

```
The probability that T is less than 12.0 is 0.9501
```

Comments

Informational error

Type	Code	
5	5	An accurate result cannot be computed due to possible underflow for the machine precision available. `DELTA*SQRT(IDF/(IDF+T**2))` must be less than `SQRT(-1.9*ALOG(S))`, where `S=AMACH(1)`.

Description

Function `TNDF` evaluates the distribution function F of a noncentral t random variable with `IDF` degrees of freedom and noncentrality parameter `DELTA`; that is, with v = `IDF`, δ = `DELTA` , and t_0 = `T`,

$$F(t_0) = \int_{-\infty}^{t_0} \frac{v^{v/2} e^{-\delta^2/2}}{\sqrt{\pi}\Gamma(v/2)(v+x^2)^{(v+1)/2}} \sum_{i=0}^{\infty} \Gamma((v+i+1)/2)\left(\frac{\delta^i}{i!}\right)\left(\frac{2x^2}{v+x^2}\right)^{i/2} dx$$

where $\Gamma(\cdot)$ is the gamma function. The value of the distribution function at the point t_0 is the probability that the random variable takes a value less than or equal to t_0.

The noncentral t random variable can be defined by the distribution function above, or alternatively and equivalently, as the ratio of a normal random variable and an independent chi-squared random variable. If w has a normal distribution with mean δ and variance equal to one, u has an independent chi-squared distribution with v degrees of freedom, and

$$x = w / \sqrt{u / v}$$

then x has a noncentral t distribution with degrees of freedom and noncentrality parameter δ.

The distribution function of the noncentral t can also be expressed as a double integral involving a normal density function (see, for example, Owen 1962, page 108). The function TNDF uses the method of Owen (1962, 1965), which uses repeated integration by parts on that alternate expression for the distribution function.

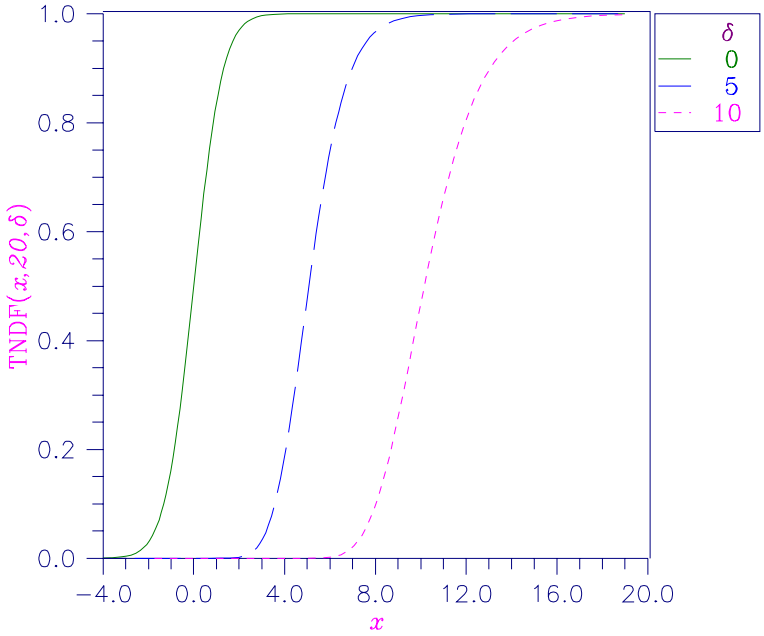


Figure 17-13 Noncentral Student's t Distribution Function

TNIN

This function evaluates the inverse of the noncentral Student's t distribution function.

Function Return Value

TNIN — Function value. (Output)
The probability that a noncentral Student's t random variable takes a value less than or equal to TNIN is P.

Required Arguments

P — Probability for which the inverse of the noncentral Student's t distribution function is to be evaluated. (Input)
P must be in the open interval (0.0, 1.0).

IDF — Number of degrees of freedom of the noncentral Student's *t* distribution. (Input) `IDF` must be positive.

DELTA — The noncentrality parameter. (Input)

FORTRAN 90 Interface

Generic: `TNIN(P, IDF, DELTA)`

Specific: The specific interface names are `S_TNIN` and `D_TNIN`.

FORTRAN 77 Interface

Single: `TNIN(P, IDF, DELTA)`

Double: The double precision name is `DTNIN`.

Example

In this example, we find the 95-th percentage point for a noncentral *t* random variable with 6 degrees of freedom and noncentrality parameter 6.

```
      USE TNIN_INT
      USE UMACH_INT
      INTEGER    IDF, NOUT
      REAL       DELTA, P, T
!
      CALL UMACH (2, NOUT)
      IDF   = 6
      DELTA = 6.0
      P     = 0.95
      T     = TNIN(P,IDF,DELTA)
      WRITE (NOUT,99999) T
!
99999 FORMAT (' The 0.05 noncentral t critical value is ', F6.3, &
             '.')
!
      END
```

Output

```
The 0.05 noncentral t critical value is 11.995.
```

Comments

Informational error

Type	Code	
4	1	Over 100 iterations have occurred without convergence. Convergence is assumed.

Description

Function TNIN evaluates the inverse distribution function of a noncentral t random variable with IDF degrees of freedom and noncentrality parameter DELTA; that is, with P = P, v = IDF, and δ = DELTA, it determines t_0 (= TNIN(P, IDF, DELTA)), such that

$$P = \int_{-\infty}^{t_0} \frac{v^{v/2} e^{-\delta^2/2}}{\sqrt{\pi}\Gamma(v/2)(v+x^2)^{(v+1)/2}} \sum_{i=0}^{\infty} \Gamma((v+i+1)/2)\left(\frac{\delta^i}{i!}\right)\left(\frac{2x^2}{v+x^2}\right)^{i/2} dx$$

where $\Gamma(\cdot)$ is the gamma function. The probability that the random variable takes a value less than or equal to t_0 is P. See TNDF (page 1291) for an alternative definition in terms of normal and chi-squared random variables. The function TNIN uses bisection and modified regula falsi to invert the distribution function, which is evaluated using routine TNDF.

GCDF

This function evaluates a general continuous cumulative distribution function given ordinates of the density.

Function Return Value

GCDF — Function value, the probability that a random variable whose density is given in F takes a value less than or equal to X0. (Output)

Required Arguments

X0 —Point at which the distribution function is to be evaluated. (Input)

X — Array containing the abscissas or the endpoints. (Input)
If IOPT = 1 or 3, X is of length 2. If IOPT = 2 or 4, X is of length M. For IOPT = 1 or 3, X(1) contains the lower endpoint of the support of the distribution and X(2) is the upper endpoint. For IOPT = 2 or 4, X contains, in strictly increasing order, the abscissas such that X(I) corresponds to F(I).

F — Vector of length M containing the probability density ordinates corresponding to increasing abscissas. (Input)
If IOPT = 1 or 3, for I = 1, 2, …, M, F(I) corresponds to X(1) + (I – 1) * (X(2) – X(1))/(M –1); otherwise, F and X correspond one for one.

Optional Arguments

IOPT — Indicator of the method of interpolation. (Input)
Default: IOPT = 1.

IOPT	Interpolation Method
1	Linear interpolation with equally spaced abscissas.

2 Linear interpolation with possibly unequally spaced abscissas.

3 A cubic spline is fitted to equally spaced abscissas.

4 A cubic spline is fitted to possibly unequally spaced abscissas.

M —Number of ordinates of the density supplied. (Input)
`M` must be greater than 1 for linear interpolation (`IOPT` = 1 or 2) and greater than 3 if a curve is fitted through the ordinates (`IOPT` = 3 or 4).
Default: `M` = size (`F`,1).

FORTRAN 90 Interface

Generic: `GCDF(X0, X, F [,…])`

Specific: The specific interface names are `S_GCDF` and `D_GCDF`.

FORTRAN 77 Interface

Single: `GCDF(X0, IOPT, M, X, F)`

Double: The double precision name is `DGCDF`.

Example

In this example, we evaluate the beta distribution function at the point 0.6. The probability density function of a beta random variable with parameters p and q is

$$f(x) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} x^{p-1}(1-x)^{q-1} \quad \text{for } 0 \le x \le 1$$

where $\Gamma(\cdot)$ is the gamma function. The density is equal to 0 outside the interval [0, 1]. We compute a constant multiple (we can ignore the constant gamma functions) of the density at 300 equally spaced points and input this information in `X` and `F`. Knowing that the probability density of this distribution is very peaked in the vicinity of 0.5, we could perhaps get a better fit by using unequally spaced abscissas, but we will keep it simple. Note that this is the same example as one used in the description of routine `BETDF` (page 1265). The result from `BETDF` would be expected to be more accurate than that from `GCDF` since `BETDF` is designed specifically for this distribution.

```
      USE UMACH_INT
      USE GCDF_INT
      INTEGER    M
      PARAMETER  (M=300)
!
      INTEGER    I, IOPT, NOUT
      REAL       F(M), H, P, PIN1, QIN1, X(2), X0, XI
!
      CALL UMACH (2, NOUT)
```

```
      X0   = 0.6
      IOPT = 3
!                                  Initializations for a beta(12,12)
!                                  distribution.
      PIN1 = 11.0
      QIN1 = 11.0
      XI   = 0.0
      H    = 1.0/(M-1.0)
      X(1) = XI
      F(1) = 0.0
      XI   = XI + H
!                                  Compute ordinates of the probability
!                                  density function.
      DO 10  I=2, M - 1
         F(I) = XI**PIN1*(1.0-XI)**QIN1
         XI   = XI + H
   10 CONTINUE
      X(2) = 1.0
      F(M) = 0.0
      P    = GCDF(X0, X, F, IOPT=IOPT)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that X is less than 0.6 is ', F6.4)
      END
```

Output

```
The probability that X is less than 0.6 is 0.8364
```

Comments

If IOPT = 3, automatic workspace usage is:

GCDF 6 * M units, or

DGCDF 11 * M units.

If IOPT = 4, automatic workspace usage is

GCDF 5 * M units, or

DGCDF 9 * M units.

Workspace may be explicitly provided, if desired, by the use of G4DF/DG4DF. The reference is:

```
G4DF(P, IOPT, M, X, F, WK, IWK)
```

The arguments in addition to those of GCDF are:

WK — Work vector of length 5 * M if IOPT = 3, and of length 4 * M if IOPT = 4.

IWK — Work vector of length M.

Description

Function GCDF evaluates a continuous distribution function, given ordinates of the probability density function. It requires that the range of the distribution be specified in X. For distributions with infinite ranges, endpoints must be chosen so that most of the probability content is included. The function GCDF first fits a curve to the points given in X and F with either a piecewise linear interpolant or a C^1 cubic spline interpolant based on a method by Akima (1970). Function GCDF then determines the area, A, under the curve. (If the distribution were of finite range and if the fit were exact, this area would be 1.0.) Using the same fitted curve, GCDF next determines the area up to the point x_0(= X0). The value returned is the area up to x_0 divided by A. Because of the scaling by A, it is not assumed that the integral of the density defined by X and F is 1.0. For most distributions, it is likely that better approximations to the distribution function are obtained when IOPT equals 3 or 4, that is, when a cubic spline is used to approximate the function. It is also likely that better approximations can be obtained when the abscissas are chosen more densely over regions where the density and its derivatives (when they exist) are varying greatly.

GCIN

This function evaluates the inverse of a general continuous cumulative distribution function given ordinates of the density.

Function Return Value

GCIN — Function value. (Output)
The probability that a random variable whose density is given in F takes a value less than or equal to GCIN is P.

Required Arguments

P —Probability for which the inverse of the distribution function is to be evaluated. (Input)
P must be in the open interval (0.0, 1.0).

X —Array containing the abscissas or the endpoints. (Input)
If IOPT = 1 or 3, X is of length 2. If IOPT = 2 or 4, X is of length M. For IOPT = 1 or 3, X(1) contains the lower endpoint of the support of the distribution and X(2) is the upper endpoint. For IOPT = 2 or 4, X contains, in strictly increasing order, the abscissas such that X(I) corresponds to F(I).

F —Vector of length M containing the probability density ordinates corresponding to increasing abscissas. (Input)
If IOPT = 1 or 3, for I = 1, 2, …, M, F(I) corresponds to X(1) + (I – 1) * (X(2) – X(1))/(M – 1); otherwise, F and X correspond one for one.

Optional Arguments

IOPT — Indicator of the method of interpolation. (Input)
Default: IOPT = 1.

IOPT	Interpolation Method
1	Linear interpolation with equally spaced abscissas.
2	Linear interpolation with possibly unequally spaced abscissas.
3	A cubic spline is fitted to equally spaced abscissas.
4	A cubic spline is fitted to possibly unequally spaced abscissas.

M —Number of ordinates of the density supplied. (Input)
M must be greater than 1 for linear interpolation (IOPT = 1 or 2) and greater than 3 if a curve is fitted through the ordinates (IOPT = 3 or 4).
Default: M = size (F,1).

FORTRAN 90 Interface

Generic: `GCIN (P, X, F [,…])`

Specific: The specific interface names are `S_GCIN` and `D_GCIN`.

FORTRAN 77 Interface

Single: `GCIN(P, IOPT, M, X, F)`

Double: The double precision name is `DGCIN`.

Example

In this example, we find the 90-th percentage point for a beta random variable with parameters 12 and 12. The probability density function of a beta random variable with parameters p and q is

$$f(x) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} x^{p-1}(1-x)^{q-1} \quad \text{for } 0 \le x \le 1$$

where $\Gamma(\cdot)$ is the gamma function. The density is equal to 0 outside the interval [0, 1]. With $p = q$, this is a symmetric distribution. Knowing that the probability density of this distribution is very peaked in the vicinity of 0.5, we could perhaps get a better fit by using unequally spaced abscissas, but we will keep it simple and use 300 equally spaced points. Note that this is the same example that is used in the description of routine BETIN (page 1268). The result from BETIN would be expected to be more accurate than that from GCIN since BETIN is designed specifically for this distribution.

```
      USE GCIN_INT
      USE UMACH_INT
      INTEGER    M
      PARAMETER  (M=300)
!
      INTEGER    I, IOPT, NOUT
      REAL       C, F(M), H, P, PIN, PIN1, QIN, QIN1, X(2), X0, XI
!
      CALL UMACH (2, NOUT)
      P    = 0.9
      IOPT = 3
!                                Initializations for a beta(12,12)
!                                distribution.
      PIN  = 12.0
      QIN  = 12.0
      PIN1 = PIN - 1.0
      QIN1 = QIN - 1.0
      C    = 1.0/BETA(PIN,QIN)
      XI   = 0.0
      H    = 1.0/(M-1.0)
      X(1) = XI
      F(1) = 0.0
      XI   = XI + H
!                                Compute ordinates of the probability
!                                density function.
      DO 10  I=2, M - 1
         F(I) = C*XI**PIN1*(1.0-XI)**QIN1
         XI   = XI + H
   10 CONTINUE
      X(2) = 1.0
      F(M) = 0.0
      X0   = GCIN(P,X,F,IOPT=IOPT)
      WRITE (NOUT,99999) X0
99999 FORMAT (' X is less than ', F6.4, ' with probability 0.9.')
      END
```

Output

```
X is less than 0.6304 with probability 0.9.
```

Comments

If IOPT = 3, automatic workspace usage is

GCIN	6 * M units, or
DGCIN	11 * M units.

If IOPT = 4, automatic workspace usage is

GCIN	5 * M units, or
DGCIN	9 * M units.

Workspace may be explicitly provided, if desired, by the use of G3IN/DG3IN. The reference is:

```
G3IN(P, IOPT, M, X, F, WK, IWK)
```

The arguments in addition to those of GCIN are:

WK — Work vector of length 5 * M if IOPT = 3, and of length 4 * M if IOPT = 4.

IWK — Work vector of length M.

Description

Function GCIN evaluates the inverse of a continuous distribution function, given ordinates of the probability density function. The range of the distribution must be specified in X. For distributions with infinite ranges, endpoints must be chosen so that most of the probability content is included.

The function GCIN first fits a curve to the points given in X and F with either a piecewise linear interpolant or a *C* cubic spline interpolant based on a method by Akima (1970). Function GCIN then determines the area, *A*, under the curve. (If the distribution were of finite range and if the fit were exact, this area would be 1.0.) It next finds the maximum abscissa up to which the area is less than *AP* and the minimum abscissa up to which the area is greater than *AP* . The routine then interpolates for the point corresponding to *AP*. Because of the scaling by *A*, it is not assumed that the integral of the density defined by X and F is 1.0.

For most distributions, it is likely that better approximations to the distribution function are obtained when IOPT equals 3 or 4, that is, when a cubic spline is used to approximate the function. It is also likely that better approximations can be obtained when the abscissas are chosen more densely over regions where the density and its derivatives (when they exist) are varying greatly.

GFNIN

This function evaluates the inverse of a general continuous cumulative distribution function given in a subprogram.

Function Return Value

GFNIN — The inverse of the function F at the point P. (Output)
F(GFNIN) is "close" to P.

Required Arguments

F — User-supplied FUNCTION to be inverted. F must be continuous and strictly monotone.
The form is F(X), where
X — The argument to the function. (Input)
F — The value of the function at X. (Output)
F must be declared EXTERNAL in the calling program.

P — The point at which the inverse of F is desired. (Input)

GUESS — An initial estimate of the inverse of `F` at `P`. (Input)

Optional Arguments

EPS — Convergence criterion. (Input)
When the relative change in `GFNIN` from one iteration to the next is less than `EPS`, convergence is assumed. A common value for `EPS` is 0.0001. Another common value is 100 times the machine epsilon.
Default: `EPS` = 100 times the machine epsilon.

FORTRAN 90 Interface

Generic: `GFNIN(F, P, GUESS [,…])`

Specific: The specific interface names are `S_GFNIN` and `D_GFNIN`.

FORTRAN 77 Interface

Single: `GFNIN(F, P, EPS, GUESS)`

Double: The double precision name is `DGFNIN`.

Example

In this example, we find the 99–th percentage point for an *F* random variable with 1 and 7 degrees of freedom. (This problem could be solved easily using routine `FIN` (page 1282). Compare the example for `FIN`). The function to be inverted is the *F* distribution function, for which we use routine `FDF` (page 1280). Since `FDF` requires the degrees of freedom in addition to the point at which the function is evaluated, we write another function `F` that receives the degrees of freedom via a common block and then calls `FDF`. The starting point (initial guess) is taken as two standard deviations above the mean (since this would be a good guess for a normal distribution). It is not necessary to supply such a good guess. In this particular case, an initial estimate of 1.0, for example, yields the same answer in essentially the same number of iterations. (In fact, since the *F* distribution is skewed, the initial guess, 7.0, is really not that close to the final answer.)

```
      USE UMACH_INT
      USE GFNIN_INT
      INTEGER    NOUT
      REAL       DFD, DFN, F, F0, GUESS, P, SQRT
      COMMON     /FCOM/ DFN, DFD
      INTRINSIC  SQRT
      EXTERNAL   F
!
      CALL UMACH (2, NOUT)
      P   = 0.99
      DFN = 1.0
      DFD = 7.0
!                                Compute GUESS as two standard
```

```
!                                  deviations above the mean.
      GUESS = DFD/(DFD-2.0) + 2.0*SQRT(2.0*DFD*DFD*(DFN+DFD-2.0)/(DFN* &
              (DFD-2.0)**2*(DFD-4.0)))
      F0    = GFNIN(F,P,GUESS)
      WRITE (NOUT,99999) F0
99999 FORMAT (' The F(1,7) 0.01 critical value is ', F6.3)
      END
!
      REAL FUNCTION F (X)
      REAL       X
!
      REAL       DFD, DFN, FDF
      COMMON     /FCOM/ DFN, DFD
      EXTERNAL   FDF
!
      F = FDF(X,DFN,DFD)
      RETURN
      END
```

Output

```
The F(1,7) 0.01 critical value is 12.246
```

Comments

1. Informational errors

Type	Code	
4	1	After 100 attempts, a bound for the inverse cannot be determined. Try again with a different initial estimate.
4	2	No unique inverse exists.
4	3	Over 100 iterations have occurred without convergence. Convergence is assumed.

2. The function to be inverted need not be a distribution function, it can be any continuous, monotonic function.

Description

Function GFNIN evaluates the inverse of a continuous, strictly monotone function. Its most obvious use is in evaluating inverses of continuous distribution functions that can be defined by a FORTRAN function. If the distribution function cannot be specified in a FORTRAN function, but the density function can be evaluated at a number of points, then routine GCIN (page 1298) can be used.

Function GFNIN uses regula falsi and/or bisection, possibly with the Illinois modification (see Dahlquist and Bjorck 1974). A maximum of 100 iterations are performed.

Chapter 18: Random Number Generation

Routines

Usage Notes

In the following discussions, the phrases "random numbers," "random deviates," "deviates," and "variates" are used interchangeably. The phrase "pseudorandom" is sometimes used to emphasize that the numbers generated are not really "random" since they result from a deterministic process. The usefulness of pseudorandom numbers derives from the similarity, in a statistical sense, of samples of the pseudorandom numbers to samples of observations from the specified distributions. In short, while the pseudorandom numbers are completely deterministic and repeatable, they *simulate* the realizations of independent and identically distributed random variables.

The Basic Uniform Generators

The random number generators in this chapter use either a multiplicative congruential method or a generalized feedback shift register (GFSR) method. The selection of the type of generator is made by calling the routine RNOPT (page 1311). If no selection is made explicitly, a multiplicative generator (with multiplier 16807) is used. Whatever distribution is being simulated, uniform (0, 1) numbers are first generated and then transformed if necessary. The generation of the uniform (0, 1) numbers is done by the routine RNUN (page 1316), or by its function analog RNUNF. These routines are *portable* in the sense that, given the same seed and for a given type of generator, they produce the same sequence in all computer/compiler environments. There are many other issues that must be considered in developing programs for the methods described below (see Gentle 1981 and 1990).

The Multiplicative Congruential Generators

The form of the multiplicative congruential generators is

$$x_i \equiv cx_{i-1} \bmod \left(2^{31} - 1\right)$$

Each x_i is then scaled into the unit interval (0, 1). If the multiplier, c, is a primitive root modulo $2^{31} - 1$ (which is a prime), then the generator will have maximal period of $2^{31} - 2$. There are several other considerations, however. The lattice structure induced by congruential generators (see Marsaglia 1968) can be assessed by the lattice test of Marsaglia (1972) or the spectral test of Coveyou and MacPherson (1967) (see also Knuth 1981, pages 89–113). Also, empirical studies, such as by Fishman and Moore (1982 and 1986), indicate that different values of multipliers, all of which perform well under the lattice test and the spectral test, may yield quite different performances where the criterion is similarity of samples generated to samples from a true uniform distribution.

There are three possible choices for c in the IMSL generators: 16807 (which is 7^5), 397204094 (which is $2 \cdot 7^2 \cdot 4053103$), and 950706376 (which is $2^3 \cdot 118838297$). The selection is made by the routine RNOPT (page 1311). The choice of 16807 will result in the fastest execution time (see Gentle 1981), but Fishman and Moore's studies would seem to indicate that the performance of 950706376 is best among these three choices. If no selection is made explicitly, the routines use the multiplier 16807, which has been in use for some time (Lewis, Goodman, and Miller 1969). It is the "minimal standard generator" discussed by Park and Miller (1988).

The user can also select a shuffled version of the multiplicative congruential generators using RNOPT (page 1311). The shuffled generators use a scheme due to Learmonth and Lewis (1973a). In this scheme, a table is filled with the first 128 uniform (0, 1) numbers resulting from the simple multiplicative congruential generator. Then, for each x_i from the simple generator, the low-order bits of x_i are used to select a random integer, j, from 1 to 128. The j-th entry in the table is then delivered as the random number; and x_i, after being scaled into the unit interval, is inserted into the j-th position in the table.

The Generalized Feedback Shift Register Generator

The GFSR generator uses the recursion $X_t = X_{t\text{-}1563} \oplus X_{t\text{-}96}$. This generator, which is different from earlier GFSR generators, was proposed by Fushimi (1990), who discusses the theory behind

the generator and reports on several empirical tests of it. Background discussions on this type of generator can be found in Kennedy and Gentle (1980), pages 150–162.

Setting the Seed

The seed of the generator can be set in RNSET (page 1311) and can be retrieved by RNGET (page 1312). Prior to invoking any generator in this chapter , the user can call RNSET to initialize the seed, which is an integer variable taking a value between 1 and 2147483646. If it is not initialized by RNSET, a random seed is obtained from the system clock. Once it is initialized, the seed need not be set again. The seed is updated and passed from one routine to another by means of a named COMMON block, R2NCOM.

If the user wishes to restart a simulation, RNGET can be used to obtain the final seed value of one run to be used as the starting value in a subsequent run. Also, if two random number streams are desired in one run, RNSET and RNGET can be used before and after the invocations of the generators in each stream. If a shuffled generator or the GFSR generator is used, in addition to resetting the seed, the user must also reset some values in a table. For the shuffled generators, this is done using the routines RNGES (page 1312) and RNSES (page 1312); and for the GFSR generator, the table is retrieved and set by the routines RNGEF (page 1312) and RNSEF (page 1312). The tables for the shuffled generators are separate for single and double precision; so, if precisions are mixed in a program, it is necessary to manage each precision separately for the shuffled generators.

Timing Considerations

The generation of the uniform (0,1) numbers is done by the routine RNUN (page 1316) or by its function analog RNUNF (page 1317). The particular generator selected in RNOPT (page 1311), that is, the value of the multiplier and whether shuffling is done or whether the GFSR generator is used, affects the speed of RNUN and RNUNF. The smaller multiplier (16807, selected by IOPT = 1) is faster than the other multipliers. The multiplicative congruential generators that do not shuffle are faster than the ones that do. The GFSR generator is roughly as fast as the fastest multiplicative congruential generator, but the initialization for it (required only on the first invocation) takes longer than the generation of thousands of uniform random numbers. Precise statements of relative speeds depend on the computing system.

Whether RNUN or RNUNF is used also has an effect on the speed due to the overhead in invoking an external routine, or due to the program's inability to optimize computations by holding some operands in registers. This effect, of course, may be different in different environments. On an array processor or other computers with pipelined instructions, RNUN is likely to be considerably faster than RNUNF when several random numbers are to be generated at one time. In the case of array processors, the multiplicative congruential generators in RNUN are coded to generate subsequences in larger blocks (see Gentle 1990).

Use of Customized Uniform Generators

The basic uniform (0, 1) generators RNUN or RNUNF are used by all other routines in this chapter. If, for some reason, the user would prefer a different basic uniform generator, routines named "RNUN" and "RNUNF" can be written so that they include the named COMMON, through which the seed is passed, and that calls the user's custom generator. The named COMMON is

```
COMMON /R2NCOM/ D2P31A, DSEED, D2P31R, DWK, DINTTB, INDCTR, &
      INTTB, WK, ICEED, IDSTFS, INTFS, ISRCFS, S2P31R, IWFS
DOUBLE PRECISION D2P31A, D2P31R, DSEED, DWK(128)
REAL    S2P31R, WK(128)
INTEGER ICEED, IDSTFS, INDCTR, ISRCFS, IWFS(1563)
LOGICAL DINTTB, INTTB, INTFS
SAVE    /R2NCOM/
```

The user's "RNUN" and "RNUNF" can pass the seed through any of the variables, but the routines RNSET (page 1311) and RNGET (page 1312) expect the seed to be in ICEED. (The user should not expect to use any utility routines other than RNSET and RNGET if customized versions of RNUN or RNUNF are used.) The double precision versions of the nonuniform generators, such as DRNBET and DRNGAM (page 1349), use the double precision versions of the uniform generators, DRNUN (page 1316) and DRNUNF (page 1317), so to use the double precision nonuniform generators with customized uniform generators, the user would supply routines to replace DRNUN and DRNUNF.

Distributions Other than the Uniform

The nonuniform generators use a variety of transformation procedures. All of the transformations used are exact (mathematically). The most straightforward transformation is the *inverse CDF technique*, but it is often less efficient than others involving *acceptance/rejection* and *mixtures*. See Kennedy and Gentle (1980) for discussion of these and other techniques.

Many of the nonuniform generators in this chapter use different algorithms depending on the values of the parameters of the distributions. This is particularly true of the generators for discrete distributions. Schmeiser (1983) gives an overview of techniques for generating deviates from discrete distributions.

Although, as noted above, the uniform generators yield the same sequences on different computers, because of rounding, the nonuniform generators that use acceptance/rejection may occasionally produce different sequences on different computer/compiler environments.

Although the generators for nonuniform distributions use fast algorithms, if a very large number of deviates from a fixed distribution are to be generated, it might be worthwhile to consider a table sampling method, as implemented in the routines RNGDA (page 1321), RNGDS (page 1324), RNGDT (page 1327), RNGCS (page 1351), and RNGCT (page 1354). After an initialization stage, which may take some time, the actual generation may proceed very fast.

Order Statistics and Antithetic Variates

For those generators, such as RNCHY (page 1344) and RNNOR (page 1361), that use the inverse CDF technique, it is possible to generate any set of order statistics directly by use of a customized uniform generator, as discussed above, by generating order statistics in a custom "RNUN" or "RNUNF". In some routines that employ an inverse CDF technique, such as RNEXP (page 1346) and RNWIB (page 1368), instead of directly using the uniform (0, 1) deviate u from RNUN (page 1316), the uniform (0, 1) deviate $1 - u$ is used. In such routines the i-th order, statistic from the uniform will yield the $(n + 1 - i)$-th order statistic from the nonuniform distribution.

A similar technique can be used to get antithetic variates. For each uniform deviate u, a second deviate $1 - u$ could be produced by a custom "RNUN" or "RNUNF". As with order statistics, this technique would only be reasonable for routines that use the inverse CDF technique.

Tests

Extensive empirical tests of some of the uniform random number generators available in RNUN (page 1316) and RNUNF (page 1317) are reported by Fishman and Moore (1982 and 1986). Results of tests on the generator using the multiplier 16807 with and without shuffling are reported by Learmonth and Lewis (1973b). If the user wishes to perform additional tests, the routines in Chapter 7, "Tests of Goodness of Fit and Randomness," may be of use. The user may also wish to compute some basic statistics or to make some plots of the output of the random number generator being used. The routines in Chapter 1, "Basic Statistics," and Chapter 16, "Line Printer Graphics," may be used for this purpose. Often in Monte Carlo applications, it is appropriate to construct an ad hoc test that is sensitive to departures that are important in the given application. For example, in using Monte Carlo methods to evaluate a one-dimensional integral, autocorrelations of order one may not be harmful, but they may be disastrous in evaluating a two-dimensional integral. Although generally the routines in this chapter for generating random deviates from nonuniform distributions use exact methods, and, hence, their quality depends almost solely on the quality of the underlying uniform generator, it is often advisable to employ an ad hoc test of goodness of fit for the transformations that are to be applied to the deviates from the nonuniform generator.

Other Notes on Usage

The generators for continuous distributions are available in both single and double precision versions. This is merely for the convenience of the user; the double precision versions should not be considered more "accurate," except possibly for the multivariate distributions.

The names of all of the routines for random number generation begin with "RN" for single precision and "DRN" for double precision. In most routines, the first argument, NR, is the number of variates to generate; and the last variable, either R or IR, is the vector of random variates.

Error handling and workspace allocation in the routines for random number generation are done somewhat differently than in most other IMSL routines. In general, there is less error checking than in other routines since there is more emphasis on speed in the random number generation routines. Simple checks for gross errors are made in all routines; and the routines for setup do complete checking since it is assumed that they would not be called frequently. Some routines, such as those that construct tables or interpolate from tables, require that the user explicitly provide some work arrays.

Random Number Generation Utility Routines

All of the random number generators in this chapter depend on the generation of uniform (0, 1) numbers, which is done by the routine RNUN (page 1316), or by its function analog RNUNF (page 1317). These basic generators use either a multiplicative congruential method or a generalized feedback shift register (GFSR) method to yield a subsequence of a fixed cyclic sequence. The beginning of the subsequence is determined by the seed.

The utility routines for the random number generators allow the user to select the type of the generator (or to determine the type of the generator being used) and to set or retrieve the seed.

Selection of the Type of the Generator

For generating uniform (0, 1) random numbers either a multiplicative congruential or a GFSR method is used. In the multiplicative congruential method one of three different multipliers, with or without shuffling, can be chosen. The selection of the type of generator is made by calling the routine RNOPT, choosing one of seven different options. The usage is

```
CALL RNOPT (IOPT)
```

The argument is:

IOPT — The indicator of the generator. (Input)

IOPT	Generator
1	Congruential, with multiplier 16807 is used.
2	Congruential, with multiplier 16807 is used with shuffling.
3	Congruential, with multiplier 397204094 is used.
4	Congruential, with multiplier 397204094 is used with shuffling.
5	Congruential, with multiplier 950706376 is used.
6	Congruential, with multiplier 950706376 is used with shuffling.
7	GFSR, with the recursion $X_t = X_{t\text{-}1563} \oplus X_{t\text{-}96}$ is used.

If no selection is made explicitly, a multiplicative generator (with multiplier 16807) is used (equivalent to IOPT = 1).

The type of generator being used can be determined by calling the routine RNOPG. The usage is

```
CALL RNOPG (IOPT)
```

IOPT is an output variable in RNOPG.

Setting the Seed

Before using any of the random number generators, the generator must be initialized by selecting a *seed*, or starting value. The user does not have to do this, but it can done by calling the routine RNSET. If the user does not select a seed, one is generated using the system clock. A seed needs to be selected only once in a program unless there is some desire to maintain two separate streams of random numbers. The usage is

```
CALL RNSET (ISEED)
```

The argument is:

ISEED — The seed of the random number generator. (Input)

ISEED must be in the range (0, 2147483646). If ISEED is zero (or if RNSET is not called before the generation of random numbers begins), a value is computed using the system clock; and, hence, the results of programs using the IMSL random number generators will be different at different times.

Stopping and Restarting Simulations and Controlling More Than One Stream of Random Numbers

For most purposes, even if several simulations are being run in the same program or if the simulation is being conducted in blocks, it is best to use the sequence of uniform random deviates just as produced by RNUN (page 1316) or RNUNF (page 1317) without concern for from where in the underlying cyclic sequence the numbers are coming.

If, however, the simulations are being conducted incrementally or if simulations are being run in parallel, it may be necessary to exercise more control over the sequence. The routines that are used in stopping and restarting simulations come in pairs, one to get the current value and one to set the value. The argument for each pair is the same within the pair; it is output in one case and input in the other. (RNSET is an exception since it is often used at the beginning of a simulation before any seed is ever set.) If a nonshuffled form of the multiplicative congruential generators is used (that is IOPT in RNOPT, page 1311, is 1, 3, or 5), the only thing that must be controlled is the seed, so in this case the only routines needed are

RNGET — Retrieve the current value of the seed

RNSET — Initialize the seed used in the generators

The usages are:

```
CALL RNGET (ISEED)  (ISEED is output.)
CALL RNSET (ISEED)  (ISEED is input.)
```

ISEED is an integer in the range 1 to 2147483646 (except, as noted above, it can be input to RNSET as 0 to indicate that the system clock is to be used to generate a seed).

If a shuffled generator or the GFSR generator is used, in addition to controlling the seed as described above, another array must be maintained if the user wishes to stop and restart the simulation. It is a floating-point array for the shuffled generators and an integer array for the GFSR generator. The routines are:

RNGES — Retrieve the current table used in the shuffled generators.

RNSES — Initialize the table used in the shuffled generators.

RNGEF — Retrieve the current table used in the GFSR generator.

RNSEF — Initialize the table used in the GFSR generator.

There are different tables used in the single and double precision versions of the shuffled generators, so RNGES and RNSES can also be used in double precision.

The usages are:

```
CALL RNGES (TABLE)  (TABLE is output.)
CALL RNSES (TABLE)  (TABLE is input.)
CALL RNGEF (IARRAY)  (IARRAY is output.)
CALL RNSEF (IARRAY)  (IARRAY is input.)
```

The arguments are:

TABLE — Array of length 128 used in the shuffled generators.

IARRAY — Array of length 1565 used in the GFSR generators.

The values in both TABLE and IARRAY are initialized by the IMSL random number generators. The values are all positive in both arrays except if the user wishes to reinitialize the array, in which case the first element of the array is input as a nonpositive value. (Usually, one should avoid reinitializing these arrays, but it might be necessary sometimes in restarting a simulation.) If the first element of TABLE or IARRAY is set to a nonpositive value on the call to RNSES or RNSEF, on the next invocation of a routine to generate random numbers using shuffling (if RNSES) or a GFSR method (if RNSEF), the appropriate array will be reinitialized.

In addition to controlling separate streams of random numbers, sometimes it is desirable to insure from the beginning that two streams do not overlap. This can be done with the congruential generators that do not do shuffling by using RNISD to get a seed that will generate random numbers beginning 100,000 numbers farther along.

The usage is:

```
CALL RNISD (ISEED1, ISEED2)
```

The arguments are:

ISEED1 — The seed that yields the first stream. (Input)

ISEED2 — The seed that yields a stream beginning 100,000 numbers beyond the stream that begins with ISEED1. (Output)

Given a seed, ISEED1, RNISD determines another seed, ISEED2, such that if one of the IMSL multiplicative congruential generators, using no shuffling, went through 100,000 generations starting with ISEED1, the next number in that sequence would be the first number in the sequence that begins with the seed ISEED2. This can be described more simply by stating that RN1 and RN2 in the following sequence of FORTRAN are assigned the same values.

```
    CALL RNISD(ISEED1, ISEED2)
    CALL RNSET(ISEED1)
    DO 10 I = 1, 100000
       RN1 = RNUNF()
 10 CONTINUE
    RN1 = RNUNF()
    CALL RNSET(ISEED2)
    RN2 = RNUNF()
```

To obtain seeds that generate sequences with beginning values separated by 200,000 numbers, call RNISD twice:

```
            CALL RNISD(ISEED1, ISEED2)
            CALL RNISD(ISEED2, ISEED2)
```

Note that RNISD works only when a multiplicative congruential generator without shuffling is used. This means that either the routine RNOPT (page 1311) has not been called at all or that it has been last called with IOPT taking a value of 1, 3, or 5.

For many of the IMSL generators for nonuniform distributions that do not use the inverse CDF method, the distance between the sequences generated starting with `ISEED1` and starting with `ISEED2` may be less than 100,000. This is because the nonuniform generators that use other techniques may require more than one uniform deviate for each output deviate.

The reason that one may want two seeds that generate sequences a known distance apart is for blocking Monte Carlo experiments or for running parallel streams.

Example 1

Selecting the Type of Generator and Stopping and Restarting the Simulations

In this example, three separate simulation streams are used, each with a different form of the generator. Each stream is stopped and restarted. (Although this example is obviously an artificial one, there may be reasons for maintaining separate streams and stopping and restarting them because of the nature of the usage of the random numbers coming from the separate streams.)

```
      USE IMSL_LIBRARIES
      INTEGER    I, IARRAY(1565), ISEED1, ISEED2, ISEED7, NOUT, NR
      REAL       R(5), TABLE(128)
!
      CALL UMACH (2, NOUT)
      NR     = 5
      ISEED1 = 123457
      ISEED2 = 123457
      ISEED7 = 123457
!                                Begin first stream, IOPT = 1 (by
!                                default)
      CALL RNSET (ISEED1)
      CALL RNUN (R)
      CALL RNGET (ISEED1)
      WRITE (NOUT,99997) (R(I),I=1,NR), ISEED1
!                                Begin second stream, IOPT = 2
      CALL RNOPT (2)
      CALL RNSET (ISEED2)
      CALL RNUN (R)
      CALL RNGET (ISEED2)
      CALL RNGES (TABLE)
      WRITE (NOUT,99998) (R(I),I=1,NR), ISEED2
!                                Begin third stream, IOPT = 7
      CALL RNOPT (7)
      CALL RNSET (ISEED7)
      CALL RNUN (R)
      CALL RNGET (ISEED7)
      CALL RNGEF (IARRAY)
      WRITE (NOUT,99999) (R(I),I=1,NR), ISEED7
!                                Reinitialize seed
!                                Resume first stream
      CALL RNOPT (1)
      CALL RNSET (ISEED1)
      CALL RNUN (R)
      CALL RNGET (ISEED1)
      WRITE (NOUT,99997) (R(I),I=1,NR), ISEED1
```

```
!                                  Reinitialize seed and table for
!                                  shuffling
!                                  Resume second stream
      CALL RNOPT (2)
      CALL RNSET (ISEED2)
      CALL RNSES (TABLE)
      CALL RNUN (R)
      CALL RNGET (ISEED2)
      WRITE (NOUT,99998) (R(I),I=1,NR), ISEED2
!                                  Reinitialize seed and table for GFSR
!                                  Resume third stream
      CALL RNOPT (7)
      CALL RNSET (ISEED7)
      CALL RNSEF (IARRAY)
      CALL RNUN (R)
      CALL RNGET (ISEED7)
      WRITE (NOUT,99999) (R(I),I=1,NR), ISEED7
!
99997 FORMAT (/, '  First stream  ', 5F8.4, /, '  Output seed = ', &
             I11)
99998 FORMAT (/, '  Second stream ', 5F8.4, /, '  Output seed = ', &
             I11)
99999 FORMAT (/, '  Third stream  ', 5F8.4, /, '  Output seed = ', &
             I11)
!
      END
```

Output

```
First stream    0.9662  0.2607  0.7663  0.5693  0.8448
Output seed =   1814256879
Second stream   0.7095  0.1861  0.4794  0.6038  0.3790
Output seed =   1965912801
Third stream    0.3914  0.0263  0.7622  0.0281  0.8997
Output seed =   1932158269
First stream    0.0443  0.9872  0.6014  0.8964  0.3809
Output seed =   817878095
Second stream   0.2557  0.4788  0.2258  0.3455  0.5811
Output seed =   2108806573
Third stream    0.7519  0.5084  0.9070  0.0910  0.6917
Output seed =   1485334679
```

Additional Example

Example 2: Determining Seeds for Separate Streams

In this example, RNISD (page 1313) is used to determine seeds for 4 separate streams, each 200,000 numbers apart, for a multiplicative congruential generator without shuffling. (Since RNOPT is not invoked to select a generator, the multiplier is 16807.) To get each seed requires two invocations of RNISD. All of the streams are non-overlapping, since the period of the underlying generator is 2,147,483,646.

```
USE UMACH_INT
USE RNISD_INT
```

```
      INTEGER    ISEED1, ISEED2, ISEED3, ISEED4, NOUT
!
      CALL UMACH (2, NOUT)
      ISEED1 = 123457
      CALL RNISD (ISEED1, ISEED2)
      CALL RNISD (ISEED2, ISEED2)
      CALL RNISD (ISEED2, ISEED3)
      CALL RNISD (ISEED3, ISEED3)
      CALL RNISD (ISEED3, ISEED4)
      CALL RNISD (ISEED4, ISEED4)
      WRITE (NOUT,99999) ISEED1, ISEED2, ISEED3, ISEED4
!
99999 FORMAT (’  Seeds for four separate streams: ’, /, ’  ’, 4I11)
!
      END
```

Output

```
Seeds for four separate streams:
     123457 2016130173   85016329  979156171
```

RNUN

Generates pseudorandom numbers from a uniform (0, 1) distribution.

Required Arguments

R — Vector of length NR containing the random uniform (0, 1) deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNUN (R [,…])`

Specific: The specific interface names are `S_RNUN` and `D_RNUN`.

FORTRAN 77 Interface

Single: `CALL RNUN (NR, R)`

Double: The double precision name is `DRNUN`.

Example

In this example, RNUN is used to generate five pseudorandom uniform numbers. Since RNOPT (page 1311) is not called, the generator used is a simple multiplicative congruential one with a multiplier of 16807.

```
      USE RNUN_INT
      USE UMACH_INT
      USE RNSET_INT

      INTEGER    ISEED, NOUT, NR
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      NR    = 5
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNUN (R)
      WRITE (NOUT,99999) R
99999 FORMAT (’      Uniform random deviates: ’, 5F8.4)
      END
```

Output

```
Uniform random deviates:    .9662   .2607   .7663   .5693   .8448
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNUN generates pseudorandom numbers from a uniform (0, 1) distribution using one of the algorithms described in the introduction to the chapter on random number generation. The algorithm used is determined by RNOPT (page 1311). The values returned in R by RNUN are positive and less than 1.0. Values in R may be smaller than the smallest relative spacing, however. Hence, it may be the case that some value R(i) is such that 1.0 – R(i) = 1.0.

Deviates from the distribution with uniform density over the interval (A, B) can be obtained by scaling the output from RNUN. The following statements (in single precision) would yield random deviates from a uniform (A, B) distribution:

```
USE IMSL_LIBRARIES (R, NR)
CALL RNUN (NR, R)
CALL SSCAL (NR, B-A, R, 1)
CALL SADD (NR, A, R, 1)
```

RNUNF

This function generates a pseudorandom number from a uniform (0, 1) distribution.

Function Return Value

RNUNF — Function value, a random uniform (0, 1) deviate. (Output)
See Comment 1.

Required Arguments

None.

FORTRAN 90 Interface

Generic: `RNUNF()`

Specific: The specific interface names are `S_RNUNF` and `D_RNUNF`.

FORTRAN 77 Interface

Single: `CALL RNUNF()`

Double: The double precision name is `DRNUNF`.

Example

In this example, `RNUNF` is used to generate five pseudorandom uniform numbers. Since `RNOPT` (page 1311) is not called, the generator used is a simple multiplicative congruential one with a multiplier of 16807.

```
      USE UMACH_INT
      USE RNSET_INT
      USE RNUNF_INT
      INTEGER    I, ISEED, NOUT
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      ISEED = 123457
      CALL RNSET (ISEED)
      DO 10  I=1, 5
         R(I) = RNUNF()
   10 CONTINUE
      WRITE (NOUT,99999) R
99999 FORMAT (’      Uniform random deviates: ’, 5F8.4)
      END
```

Output

```
Uniform random deviates:   0.9662  0.2607  0.7663  0.5693  0.8448
```

Comments

1. If the generic version of this function is used, the immediate result must be stored in a variable before use in an expression. For example:

```
X = RNUNF()
Y = SQRT(X)
```

must be used rather than

```
Y = SQRT(RNUNF())
```

If this is too much of a restriction on the programmer, then the specific name can be used without this restriction.

2. Routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

3. This function has a side effect: it changes the value of the seed, which is passed through a common block.

Description

Routine RNUNF is the function form of RNUN (page 1316). The routine RNUNF generates pseudorandom numbers from a uniform (0, 1) distribution . The algorithm used is determined by RNOPT (page 1311). The values returned by RNUNF are positive and less than 1.0.

If several uniform deviates are needed, it may be more efficient to obtain them all at once by a call to RNUN rather than by several references to RNUNF.

RNBIN

Generates pseudorandom numbers from a binomial distribution.

Required Arguments

N — Number of Bernoulli trials. (Input)

P — Probability of success on each trial. (Input)
P must be greater than 0.0 and less than 1.0.

IR — Vector of length NR containing the random binomial deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (IR,1).

FORTRAN 90 Interface

Generic: `CALL RNBIN (N, P, IR [,…])`

Specific: The specific interface names i S_RNBIN.

FORTRAN 77 Interface

Single: CALL RNBIN (NR, N, P, IR)

Example

In this example, RNBIN is used to generate five pseudorandom binomial variates with parameters 20 and 0.5.

```
      USE RNBIN_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    IR(NR), ISEED, N, NOUT
      REAL       P
!
      CALL UMACH (2, NOUT)
      N     = 20
      P     = 0.5
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNBIN (N, P, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT (' Binomial (20, 0.5) random deviates: ', 5I4)
      END
```

Output

```
Binomial (20, 0.5) random deviates:   14   9  12  10  12
```

Comments

The IMSL routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNBIN generates pseudorandom numbers from a binomial distribution with parameters N and P. N and P must be positive, and P must be less than 1. The probability function (with n = N and p = P) is

$$f(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

for $x = 0, 1, 2, \ldots, n$.

The algorithm used depends on the values of n and p. If $np < 10$ or if p is less than a machine epsilon (AMACH(4) (Reference Material)), the inverse CDF technique is used; otherwise, the BTPE algorithm of Kachitvichyanukul and Schmeiser (see Kachitvichyanukul 1982) is used.

This is an acceptance/rejection method using a composition of four regions. (TPE = Triangle, Parallelogram, Exponential, left and right.)

RNGDA

Generates pseudorandom numbers from a general discrete distribution using an alias method.

Required Arguments

IOPT — Indicator of whether the alias vectors are to be initialized. (Input)

IOPT	Action
0	The alias vectors are to be initialized using the probabilities in PROBS. IOPT is set to 0 on the first call to RNGDA.
1	The alias vectors IWK and WK are used but PROBS is not used.

IMIN — Smallest value the random deviate can assume. (Input)
This is the value corresponding to the probability in PROBS(1).

PROBS — Vector of length NMASS containing probabilities associated with the individual mass points. (Input)
The elements of PROBS must be nonnegative and must sum to 1.0.

IWK — Index vector of length NMASS. (Input, if IOPT = 1; output, if IOPT = 0) IWK is a work vector.

WK — Index vector of length NMASS. (Input, if IOPT = 1; output, if IOPT = 0) WK is a work vector.

IR — Vector of length NR containing the random discrete deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (IR,1).

NMASS — Number of mass points in the discrete distribution. (Input)
Default: NMASS = size (PROBS,1).

FORTRAN 90 Interface

Generic: `CALL RNGDA (IOPT, IMIN, PROBS, IWK, WK, IR [,…])`

Specific: The specific interface names are S_RNGDA and D_RNGDA.

FORTRAN 77 Interface

Single: `CALL RNGDA (NR, IOPT, IMIN, NMASS, PROBS, IWK, WK, IR)`

Double: The double precision name is `DRNGDA`.

Example 1

In this example, `RNGDA` is used to generate five pseudorandom variates from the discrete distribution:

$$\Pr(X = 1) = .05$$
$$\Pr(X = 2) = .45$$
$$\Pr(X = 3) = .31$$
$$\Pr(X = 4) = .04$$
$$\Pr(X = 5) = .15$$

When `RNGDA` is called the first time, `IOPT` is input as 0. This causes the work arrays to be initialized. In the next call, `IOPT` is 1, so the setup phase is bypassed.

```
      USE RNGDA_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    NMASS, NR
      PARAMETER  (NMASS=5, NR=5)
!
      INTEGER    IMIN, IOPT, IR(NR), ISEED, IWK(NMASS), NOUT
      REAL       PROBS(NMASS), WK(NMASS)
!
      CALL UMACH (2, NOUT)
      IMIN     = 1
      PROBS(1) = 0.05
      PROBS(2) = 0.45
      PROBS(3) = 0.31
      PROBS(4) = 0.04
      PROBS(5) = 0.15
      IOPT     = 0
      ISEED    = 123457
      CALL RNSET (ISEED)
      CALL RNGDA (IOPT, IMIN, PROBS, IWK, WK, IR)
      WRITE (NOUT,99998) IR
99998 FORMAT (’          Random deviates: ’, 5I4)
      IOPT = 1
      CALL RNGDA (IOPT, IMIN, PROBS, IWK, WK, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT (’                           ’, 5I4)
      END
```

Output

```
Random deviates:    3   2   2   3   5
                    1   3   4   5   3
```

Comments

1. In the interest of efficiency, this routine does only limited error checking when IOPT = 1.

2. The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNGDA generates pseudorandom numbers from a discrete distribution with probability function given in the vector PROBS; that is

$$\Pr(X = i) = p_j$$

for $i = i_{\min}, i_{\min} + 1, \ldots, i_{\min} + n_m - 1$ where $j = i - i_{\min} + 1$, p_j = PROBS(j), $i_{\min}$ = IMIN, and n_m = NMASS.

The algorithm is the *alias* method, due to Walker (1974), with modifications suggested by Kronmal and Peterson (1979). The method involves a setup phase, in which the vectors IWK and WK are filled. After the vectors are filled, the generation phase is very fast.

Additional Example

Example 2

In this example, RNGDA is used to generate five pseudorandom binomial variates with parameters 20 and 0.5.

```
      USE UMACH_INT
      USE RNSET_INT
      USE RNGDA_INT
      USE BINPR_INT
      INTEGER    NMASS, NR
      PARAMETER  (NMASS=21, NR=5)
!
      INTEGER    IMIN, IOPT, IR(NR), ISEED, IWK(NMASS), K, N, NOUT
      REAL       P, PROBS(NMASS), WK(NMASS)
!
      CALL UMACH (2, NOUT)
      N    = 20
      P    = 0.5
      IMIN = 0
      DO 10  K=1, NMASS
         PROBS(K) = BINPR(K-1,N,P)
   10 CONTINUE
      IOPT  = 0
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNGDA (IOPT, IMIN, PROBS, IWK, WK, IR)
```

```
      WRITE (NOUT,99999) IR
99999 FORMAT (’   Binomial (20, .5) deviates: ’, 5I4)
      END
```

Output

```
Binomial (20, .5) deviates:   12  10  16  12  11
```

RNGDS

Sets up table to generate pseudorandom numbers from a general discrete distribution.

Required Arguments

PRF — User-supplied FUNCTION to compute the probability associated with each mass point of the distribution. The form is PRF(IX), where

IX – Point at which the probability function is to be evaluated. (Input)
IX can range from IMIN to the value at which the cumulative probability is greater than or equal to 1.0 – DEL.

PRF – Value of the probability function at IX. (Output)
PRF must be declared EXTERNAL in the calling program.

DEL — Maximum absolute error allowed in computing the cumulative probability. (Input)
Probabilities smaller than DEL are ignored; hence, DEL should be a small positive number. If DEL is too small, however, CUMPR(NMASS) must be exactly 1.0 since that value is compared to 1.0 – DEL.

NNDX — The number of elements of CUMPR available to be used as indexes. (Input)
NNDX must be greater than or equal to 1. In general, the larger NNDX is, to within sixty or seventy percent of NMASS, the more efficient the generation of random numbers using RNGDS will be.

IMIN — Smallest value the random deviate can assume. (Input/Output)
IMIN is not used if IOPT = 1. If IOPT = 0, PRF is evaluated at IMIN. If this value is less than DEL, IMIN is incremented by 1 and again PRF is evaluated at IMIN. This process is continued until PRF(IMIN) ≥ DEL. IMIN is output as this value and CUMPR(1) is output as PRF(IMIN).

NMASS — The number of mass points in the distribution. (Input, if IOPT = 1; output, if IOPT = 0)
If IOPT = 0, NMASS is the smallest integer such that PRF(IMIN + NMASS – 1) > 1.0 – DEL. NMASS does include the points IMIN(in) + j for which PRF(IMIN(in) + j) < DEL, for j = 0, 1, …, IMIN(out) – IMIN(in), where IMIN(in) denotes the input value of IMIN and IMIN(out) denotes its output value.

CUMPR — Vector of length NMASS + NNDX containing in the first NMASS positions, the cumulative probabilities and in some of the remaining positions, indexes to speed access to the probabilities. (Output, if IOPT = 0; input/output, otherwise) CUMPR(NMASS + 1) + 1 is the actual number of index positions used.

Optional Arguments

IOPT — Indicator of the extent to which CUMPR is initialized prior to calling RNGDS. (Input)
Default: IOPT = 0.

IOPT	Action
0	RNGDS fills all of CUMPR, using PRF.
1	RNGDS fills only the index portion of CUMPR, using the values in the first NMASS positions. PRF is not used and may be a dummy function; also, IMIN and DEL are not used.

LCUMPR — Dimension of CUMPR exactly as specified in the dimension statement in the calling program. (Input)
Since the logical length of CUMPR is determined in RNGDS, LCUMPR is used for error checking.
Default : LCUMPR = size (CUMPR,1).

FORTRAN 90 Interface

Generic: `CALL RNGDS (PRF, DEL, NNDX, IMIN, NMASS, CUMPR [,…])`

Specific: The specific interface names are S_RNGDS and D_RNGDS.

FORTRAN 77 Interface

Single:
```
CALL RNGDS (PRF, IOPT, DEL, NNDX, IMIN, NMASS, CUMPR,
            LCUMPR)
```

Double: The double precision name is DRNGDS.

Example 1

In this example, RNGDS is used to set up a table to generate pseudorandom variates from the discrete distribution:

$$\Pr(X = 1) = .05$$
$$\Pr(X = 2) = .45$$
$$\Pr(X = 3) = .31$$
$$\Pr(X = 4) = .04$$
$$\Pr(X = 5) = .15$$

In this simple example, we input the cumulative probabilities directly in CUMPR and request 3 indexes to be computed (NNDX = 4). Since the number of mass points is so small, the indexes would not have much effect on the speed of the generation of the random variates.

```
      USE RNGDS_INT
      USE UMACH_INT
      INTEGER    LCUMPR
      PARAMETER  (LCUMPR=9)
!
      INTEGER    IMIN, IOPT, NMASS, NNDX, NOUT
      REAL       CUMPR(LCUMPR), DEL, PRF
      EXTERNAL   PRF
!
      CALL UMACH (2, NOUT)
      NMASS    = 5
      CUMPR(1) = 0.05
      CUMPR(2) = 0.50
      CUMPR(3) = 0.81
      CUMPR(4) = 0.85
      CUMPR(5) = 1.00
      IOPT     = 1
      NNDX     = 4
      DEL      = 0.00001
      CALL RNGDS (PRF, DEL, NNDX, IMIN, NMASS, CUMPR, IOPT=IOPT)
      WRITE (NOUT,99999) CUMPR
99999 FORMAT (’   Cumulative probabilities and indexes: ’, /, 9F6.2)
      END
!
!                                  Dummy function
      REAL FUNCTION PRF (IX)
      INTEGER    IX
!
      PRF = 0.0
      RETURN
      END
```

Output

```
Cumulative probabilities and indexes:
0.05  0.50  0.81  0.85  1.00  3.00  1.00  2.00  5.00
```

Comments

1. Informational error

Type	Code	
3	1	For some I, CUMPR(I) is computed to be less than 1.0 – DEL, and yet CUMPR(I + 1) – 1.0 is greater than 1.0 – CUMPR(I + 1). In this case, the maximum value that the random variable is allowed to take on is I; that is, CUMPR(I) is set to 1.0.

2. The routine RNGDT (page 1328) uses the table set up by RNGDS to generate random numbers from the distribution with CDF represented in CUMPR.

Description

Routine RNGDS sets up a table that routine RNGDT (page 1328) uses to generate pseudorandom deviates from a discrete distribution. The distribution can be specified either by its probability function PRF or by a vector of values of the cumulative probability function. Note that PRF is *not* the cumulative probability distribution function. If the cumulative probabilities are already available in CUMPR, the only reason to call RNGDS is to form an index vector in the upper portion of CUMPR so as to speed up the generation of random deviates by the routine RNGDT.

Additional Example

Example 2

This example, RNGDS is used to set up a table to generate binomial variates with parameters 20 and 0.5. The routine BINPR (see Chapter 17, Probability Distributions Functions and Inverses) is used to compute the probabilities.

```
      USE RNGDS_INT
      USE UMACH_INT
      INTEGER    LCUMPR
      PARAMETER  (LCUMPR=33)
!
      INTEGER    I, IMIN, N, NMASS, NNDX, NOUT
      REAL       CUMPR(LCUMPR), DEL, P, PRF
      COMMON     /BINCOM/ N, P
      EXTERNAL   PRF
!
      CALL UMACH (2, NOUT)
      N    = 20
      P    = 0.5
      IMIN = 0
      NNDX = 12
      DEL  = 0.00001
      CALL RNGDS (PRF, DEL, NNDX, IMIN, NMASS, CUMPR)
      WRITE (NOUT,99998) IMIN, NMASS
99998 FORMAT (' The smallest point with positive probability using ', &
             /, ' the given DEL is ', I1, ' and all points after ', /, &
             ' point number ', I2, ' (counting from the input value ', &
             /, ' of IMIN) have zero probability.')
```

```
      WRITE (NOUT,99999) (CUMPR(I),I=1,NMASS+NNDX)
99999 FORMAT (' Cumulative probabilities and indexes: ', /, (5X,8F8.4))
      END
!
!                                  Compute binomial probabilities
      REAL FUNCTION PRF (IX)
      INTEGER    IX
!
      INTEGER    N
      REAL       BINPR, P
      COMMON     /BINCOM/ N, P
      EXTERNAL   BINPR
!
      PRF = BINPR(IX,N,P)
      RETURN
      END
```

Output

```
The smallest point with positive probability using
the given DEL is 1 and all points after
point number 19 (counting from the input value
of IMIN) have zero probability.
Cumulative probabilities and indexes:
      0.0000  0.0002  0.0013  0.0059  0.0207  0.0577  0.1316  0.2517
      0.4119  0.5881  0.7483  0.8684  0.9423  0.9793  0.9941  0.9987
      0.9998  1.0000  1.0000 11.0000  1.0000  7.0000  8.0000  9.0000
      9.0000 10.0000 11.0000 11.0000 12.0000 13.0000 19.0000
```

RNGDT

Generates pseudorandom numbers from a general discrete distribution using a table lookup method.

Required Arguments

IMIN — Smallest value the random deviate can assume. (Input)
This is the value corresponding to the probability in CUMPR(1).

NMASS — Number of mass points in the discrete distribution. (Input)

CUMPR — Vector of length at least NMASS + 1 containing in the first NMASS positions the cumulative probabilities and, possibly, indexes to speed access to the probabilities. (Input)
IMSL routine RNGDS (page 1324) can be used to initialize CUMPR properly. If no elements of CUMPR are used as indexes, CUMPR(NMASS + 1) is 0.0 on input. The value in CUMPR(1) is the probability of IMIN. The value in CUMPR(NMASS) must be exactly 1.0 (since this is the CDF at the upper range of the distribution.

IR — Vector of length NR containing the random discrete deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (IR,1).

FORTRAN 90 Interface

Generic: `CALL RNGDT (IMIN, NMASS, CUMPR, IR [,…])`

Specific: The specific interface names are `S_RNGDT` and `D_RNGDT`.

FORTRAN 77 Interface

Single: `CALL RNGDT (NR, IMIN, NMASS, CUMPR, IR)`

Double: The double precision name is `DRNGDT`.

Example 1

These examples are the same ones used for the routine `RNGDS` (page 1324). In this first example, `RNGDS` is used to set up a table and then `RNGDT` is used to generate five pseudorandom variates from the discrete distribution:

$$\Pr(X = 1) = .05$$
$$\Pr(X = 2) = .45$$
$$\Pr(X = 3) = .31$$
$$\Pr(X = 4) = .04$$
$$\Pr(X = 5) = .15$$

The cumulative probabilities are input directly in `CUMPR`, and three indexes are computed by `RNGDS` (`NNDX` = 4). Since the number of mass points is so small, the indexes would not have much effect on the speed of the generation of the random variates.

```
      USE UMACH_INT
      USE RNGDS_INT
      USE RNSET_INT
      USE RNGDT_INT
      INTEGER    LCUMPR, NR
      PARAMETER  (LCUMPR=9, NR=5)
!
      INTEGER    IMIN, IOPT, IR(NR), ISEED, NMASS, NNDX, NOUT
      REAL       CUMPR(LCUMPR), DEL, PRF
      EXTERNAL   PRF
!
      CALL UMACH (2, NOUT)
      IMIN     = 1
      NMASS    = 5
      CUMPR(1) = 0.05
      CUMPR(2) = 0.50
      CUMPR(3) = 0.81
```

```
      CUMPR(4) = 0.85
      CUMPR(5) = 1.00
      IOPT     = 1
      NNDX     = 4
      DEL      = 0.00001
!                                Set up table
      CALL RNGDS (PRF, DEL, NNDX, IMIN, NMASS, CUMPR, IOPT=IOPT)
      ISEED = 123457
      CALL RNSET (ISEED)
!                                Generate variates
      CALL RNGDT (IMIN, NMASS, CUMPR, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT (' Discrete random deviates: ', 5I4)
      END
!
!                                Dummy function
      REAL FUNCTION PRF (IX)
      INTEGER    IX
!
      PRF = 0.0
      RETURN
      END
```

Output

```
Discrete random deviates:    5   2   3   3   4
```

Comments

1. Informational error

Type	Code	
3	1	The value in CUMPR(NMASS) is not exactly 1.0, but it was considered close enough to 1.0 that is was set to that value.

2. In the interest of efficiency, this routine does only limited error checking. If CUMPR is generated by the routine RNGDS (page 1324), the error checking is sufficient.

3. The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNGDT generates pseudorandom deviates from a discrete distribution, using the table CUMPR, which contains the cumulative probabilities of the distribution and, possibly, indexes to speed the search of the table. The routine RNGDS (page 1324) can be used to set up the table CUMPR. RNGDT uses the inverse CDF method to generate the variates.

Additional Example

Example 2

In this example, RNGDS (page 1324) is used to set up a table and then RNGDT is used to generate five pseudorandom variates from the binomial distribution with parameters 20 and 0.5. The routine BINPR (see Chapter 17. Probability Distribution Functions and Inverses) is used to compute the probabilities.

```
      USE UMACH_INT
      USE RNGDS_INT
      USE RNSET_INT
      USE RNGDT_INT
      INTEGER    LCUMPR, NR
      PARAMETER  (LCUMPR=33, NR=5)
!
      INTEGER    IMIN, IR(NR), ISEED, NMASS, NNDX, NOUT
      REAL       CUMPR(LCUMPR), DEL, PRF
      EXTERNAL   PRF
!
      CALL UMACH (2, NOUT)
      IMIN  = 0
      NMASS = 21
      NNDX  = 12
      DEL   = 0.00001
!                                Set up table
      CALL RNGDS (PRF, DEL, NNDX, IMIN, NMASS, CUMPR)
      ISEED = 123457
      CALL RNSET (ISEED)
!                                Generate variates
      CALL RNGDT (IMIN, NMASS, CUMPR, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT (' Binomial (20, 0.5) random deviates: ', 5I4)
      END
!
!                                Compute binomial probabilities
      REAL FUNCTION PRF (IX)
      USE BINPR_INT
      INTEGER    IX
!
      PRF = BINPR(IX,20,0.5)
      RETURN
      END
```

Output

```
Binomial (20, 0.5) random deviates:   14   9  12  10  12
```

RNGEO

Generates pseudorandom numbers from a geometric distribution.

Required Arguments

P — Probability of success on each trial. (Input)
P must be positive and less than 1.0.

IR — Vector of length NR containing the random geometric deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (IR,1).

FORTRAN 90 Interface

Generic: `CALL RNGEO (P, IR [,…])`

Specific: The specific interface name is `S_RNGEO`.

FORTRAN 77 Interface

Single: `CALL RNGEO (NR, P, IR)`

Example

In this example, RNGEO is used to generate five pseudorandom deviates from a geometric distribution with parameter P equal to 0.3.

```
      USE RNGEO_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    IR(NR), ISEED, NOUT
      REAL       P
!
      CALL UMACH (2, NOUT)
      P     = 0.3
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNGEO (P, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT (' Geometric(0.3) random deviates: ', 5I8)
      END
```

Output

```
Geometric(0.3) random deviates:        1       4       1       2       1
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNGEO generates pseudorandom numbers from a geometric distribution with parameter P, where P is the probability of getting a success on any trial. A geometric deviate can be interpreted as the number of trials until the first success (including the trial in which the first success is obtained). The probability function is

$$f(x) = P(1 - \mathrm{P})^{x-1}$$

for $x = 1, 2, \ldots$ and $0 < P < 1$

The geometric distribution as defined above has mean $1/P$.

The i-th geometric deviate is generated as the smallest integer not less than $\log(U_i)/\log(1 - P)$, where the U_i are independent uniform (0, 1) random numbers (see Knuth, 1981).

The geometric distribution is often defined on 0, 1, 2, …, with mean $(1 - P)/P$. Such deviates can be obtained by subtracting 1 from each element of IR.

RNHYP

Generates pseudorandom numbers from a hypergeometric distribution.

Required Arguments

N — Number of items in the sample. (Input)
N must be positive.

M — Number of special items in the population, or lot. (Input)
M must be positive.

L — Number of items in the lot. (Input)
L must be greater than both N and M.

IR — Vector of length NR containing the random hypergeometric deviates. (Output)
Each element of IR can be considered to be the number of special items in a sample of size N drawn without replacement from a population of size L that contains M such special items.

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (IR,1).

FORTRAN 90 Interface

Generic: `CALL RNHYP (N, M, L, IR [,…])`

Specific: The specific interface name is `S_RNHYP`.

FORTRAN 77 Interface

Single: `CALL RNHYP (NR, N, M, L, IR)`

Example

In this example, `RNHYP` is used to generate five pseudorandom deviates from a hypergeometric distribution to simulate taking random samples of size 4 from a lot containing 20 items of which 12 are defective. The resulting hypergeometric deviates represent the numbers of defectives in each of the five samples of size 4.

```
      USE RNHYP_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    IR(NR), ISEED, L, M, N, NOUT
!
      CALL UMACH (2, NOUT)
      N     = 4
      M     = 12
      L     = 20
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNHYP (N, M, L, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT ('   Hypergeometric random deviates: ', 5I8)
      END
```

Output

```
Hypergeometric random deviates:        4       2       3       3       3
```

Comments

The routine `RNSET` (page 1311) can be used to initialize the seed of the random number generator. The routine `RNOPT` (page 1311) can be used to select the form of the generator.

Description

Routine `RNHYP` generates pseudorandom numbers from a hypergeometric distribution with parameters N, M, and L. The hypergeometric random variable X can be thought of as the number of items of a given type in a random sample of size N that is drawn without replacement from a population of size L containing M items of this type. The probability function is

$$f(x) = \frac{\binom{M}{x}\binom{L-M}{N-x}}{\binom{L}{N}}$$

for $x = \max(0, N - L + M), 1, 2, \ldots, \min(N, M)$

If the hypergeometric probability function with parameters N, M, and L evaluated at $N - L + M$ (or at 0 if this is negative) is greater than the machine epsilon (AMACH(4) (Reference Material)), and less than 1.0 minus the machine epsilon, then RNHYP uses the inverse CDF technique. The routine recursively computes the hypergeometric probabilities, starting at $x = \max(0, N - L + M)$ and using the ratio $f(X = x + 1)/f(X = x)$ (see Fishman 1978, page 457).

If the hypergeometric probability function is too small or too close to 1.0, then RNHYP generates integer deviates uniformly in the interval $[1, L - i]$, for $i = 0, 1, \ldots$; and at the i-th step, if the generated deviate is less than or equal to the number of special items remaining in the lot, the occurrence of one special item is tallied and the number of remaining special items is decreased by one. This process continues until the sample size or the number of special items in the lot is reached, whichever comes first. This method can be much slower than the inverse CDF technique. The timing depends on N. If N is more than half of L (which in practical examples is rarely the case), the user may wish to modify the problem, replacing N by $L - N$, and to consider the deviates in IR to be the number of special items *not* included in the sample.

RNLGR

Generates pseudorandom numbers from a logarithmic distribution.

Required Arguments

A — Parameter of the logarithmic distribution. (Input)
A must be positive and less than 1.0.

IR — Vector of length NR containing the random logarithmic deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (IR,1).

FORTRAN 90 Interface

Generic: `CALL RNLGR (A, IR [,…])`

Specific: The specific interface name is S_RNLGR.

FORTRAN 77 Interface

Single: `CALL RNLGR (NR, A, IR)`

Example

In this example, RNLGR is used to generate 5 pseudo-random deviates from a logarithmic distribution with parameter A equal to 0.3.

```
      USE RNLGR_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    IR(NR), ISEED, NOUT
      REAL       A
!
      CALL UMACH (2, NOUT)
      A     = 0.3
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNLGR (A, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT (’  Logarithmic (0.3) random deviates: ’, 5I8)
      END
```

Output

```
Logarithmic (0.3) random deviates:        2       1       1       1       2
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNLGR generates pseudorandom numbers from a logarithmic distribution with parameter A. The probability function is

$$f(x) = -\frac{a^x}{x \ln(1-a)}$$

for $x = 1, 2, 3, \ldots$, and $0 < a < 1$.

The methods used are described by Kemp (1981) and depend on the value of A. If A is less than 0.95, Kemp’s algorithm LS, which is a “chop-down” variant of an inverse CDF technique, is used. Otherwise, Kemp’s algorithm LK, which gives special treatment to the highly probable values of 1 and 2, is used.

RNNBN

Generates pseudorandom numbers from a negative binomial distribution.

Required Arguments

RK — Negative binomial parameter. (Input)
RK must be positive.

P — Probability of success on each trial. (Input)
P must be greater than the machine epsilon, AMACH(4) (Reference Material) and less than 1.0.

IR — Vector of length NR containing the random negative binomial deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (IR,1).

FORTRAN 90 Interface

Generic: `CALL RNNBN (RK, P, IR [,…])`

Specific: The specific interface name is `S_RNNBN`.

FORTRAN 77 Interface

Single: `CALL RNNBN (NR, RK, P, IR)`

Example

In this example, RNNBN is used to generate five pseudorandom deviates from a negative binomial (Pascal) distribution with parameter *r* equal to 4 and *p* equal to 0.3.

```
      USE RNNBN_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    IR(NR), ISEED, NOUT
      REAL       P, RK
!
      CALL UMACH (2, NOUT)
      P     = 0.3
      RK    = 4.0
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNNBN (RK, P, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT (' Negative binomial (4.0, 0.3) random deviates: ', 5I4)
      END
```

Output

```
Negative binomial (4.0, 0.3) random deviates:    5    1    3    2    3
```

Comments

1. The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

2. If RK is an integer, the deviates in IR can be thought of as the number of failures in a sequence of Bernoulli trials before RK successes occur.

Description

Routine RNNBN generates pseudorandom numbers from a negative binomial distribution with parameters RK and P. RK and P must be positive and P must be less than 1. The probability function (with r = RK and p = P) is

$$f(x) = \binom{r+x-1}{x}(1-p)^r\ p^x$$

for $x = 0, 1, 2, \ldots$.

If r is an integer, the distribution is often called the Pascal distribution and can be thought of as modeling the length of a sequence of Bernoulli trials until r successes are obtained, where p is the probability of getting a success on any trial. In this form, the random variable takes values r, $r + 1$, $r + 2$, … and can be obtained from the negative binomial random variable defined above by adding r to the negative binomial variable. This latter form is also equivalent to the sum of r geometric random variables defined as taking values 1, 2, 3, … .

If $rp/(1 - p)$ is less than 100 and $(1 - p)^r$ is greater than the machine epsilon, RNNBN uses the inverse CDF technique; otherwise, for each negative binomial deviate, RNNBN generates a gamma $(r, p/(1 - p))$ deviate Y and then generates a Poisson deviate with parameter Y.

RNPOI

Generates pseudorandom numbers from a Poisson distribution.

Required Arguments

THETA — Mean of the Poisson distribution. (Input)
THETA must be positive.

IR — Vector of length NR containing the random Poisson deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (IR,1).

FORTRAN 90 Interface

Generic: `CALL RNPOI (THETA, IR [,…])`

Specific: The specific interface name is `S_RNPOI`.

FORTRAN 77 Interface

Single: `CALL RNPOI (NR, THETA, IR)`

Example

In this example, RNPOI is used to generate five pseudorandom deviates from a Poisson distribution with mean equal to 0.5.

```
      USE RNPOI_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    IR(NR), ISEED, NOUT
      REAL       THETA
!
      CALL UMACH (2, NOUT)
      THETA = 0.5
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNPOI (THETA, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT (’   Poisson(0.5) random deviates: ’, 5I8)
      END
```

Output

```
Poisson(0.5) random deviates:        2       0       1       0       1
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNPOI generates pseudorandom numbers from a Poisson distribution with parameter THETA. THETA, which is the mean of the Poisson random variable, must be positive. The probability function (with θ = THETA) is

$$f(x) = e^{-\theta}\,\theta^{x}/x!$$

for $x = 0, 1, 2, \ldots$

If THETA is less than 15, RNPOI uses an inverse CDF method; otherwise the PTPE method of Schmeiser and Kachitvichyanukul (1981) (see also Schmeiser 1983) is used.

The PTPE method uses a composition of four regions, a triangle, a parallelogram, and two negative exponentials. In each region except the triangle, acceptance/rejection is used. The execution time of the method is essentially insensitive to the mean of the Poisson.

RNUND

Generates pseudorandom numbers from a discrete uniform distribution.

Required Arguments

K — Parameter of the discrete uniform distribution. (Input)
The integers 1, 2, …, K occur with equal probability. K must be positive.

IR — Vector of length NR containing the random discrete uniform deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (IR,1).

FORTRAN 90 Interface

Generic: `CALL RNUND (K, IR [,…])`

Specific: The specific interface name is `S_RNUND`.

FORTRAN 77 Interface

Single: `CALL RNUND (NR, K, IR)`

Example

In this example, RNUND is used to generate five pseudorandom deviates from a discrete uniform distribution over the integers from 1 to 6.

```
      USE RNUND_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    IR(5), ISEED, K, NOUT, NR
!
      CALL UMACH (2, NOUT)
      K     = 6
      NR    = 5
```

```
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNUND (K, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT (’  Discrete uniform (1,6) random deviates: ’, 5I7)
      END
```

Output

```
Discrete uniform (1,6) random deviates:       6      2      5      4      6
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNUND generates pseudorandom numbers from a discrete uniform distribution over the integers 1, 2, …, K. A random integer is generated by multiplying K by a uniform (0, 1) random number, adding 1.0, and truncating the result to an integer. This, of course, is equivalent to sampling with replacement from a finite population of size K. To do the equivalent of sampling without replacement, the routine RNSRI (page 1399) can be used.

RNBET

Generates pseudorandom numbers from a beta distribution.

Required Arguments

PIN — First beta distribution parameter. (Input)
PIN must be positive.

QIN — Second beta distribution parameter. (Input)
QIN must be positive.

R — Vector of length NR containing the random standard beta deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNBET (PIN, QIN, R [,…])`

Specific: The specific interface names are S_RNBET and D_RNBET.

FORTRAN 77 Interface

Single: `CALL RNBET (NR, PIN, QIN, R)`

Double: The double precision name is `DRNBET`.

Example

In this example, `RNBET` is used to generate five pseudorandom beta (3, 2) variates.

```
      USE RNBET_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    ISEED, NOUT
      REAL       PIN, QIN, R(NR)
!
      CALL UMACH (2, NOUT)
      PIN   = 3.0
      QIN   = 2.0
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNBET (PIN, QIN, R)
      WRITE (NOUT,99999) R
99999 FORMAT (’    Beta (3,2) random deviates: ’, 5F7.4)
      END
```

Output

```
Beta (3,2) random deviates:  0.2814 0.9483 0.3984 0.3103 0.8296
```

Comments

The routine `RNSET` (page 1311) can be used to initialize the seed of the random number generator. The routine `RNOPT` (page 1311) can be used to select the form of the generator.

Description

Routine `RNBET` generates pseudorandom numbers from a beta distribution with parameters `PIN` and `QIN`, both of which must be positive. With p = `PIN` and q = `QIN`, the probability density function is

$$f(x) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} x^{p-1}(1-x)^{q-1} \quad \text{for } 0 \leq x \leq 1$$

where $\Gamma(\cdot)$ is the gamma function.

The algorithm used depends on the values of p and q. Except for the trivial cases of $p = 1$ or $q = 1$, in which the inverse CDF method is used, all of the methods use acceptance/rejection. If p and q are both less than 1, the method of Johnk (1964) is used; if either p or q is less than 1 and the other is greater than 1, the method of Atkinson (1979) is used; if both p and q are greater

than 1, algorithm BB of Cheng (1978), which requires very little setup time, is used if NR is less than 4; and algorithm B4PE of Schmeiser and Babu (1980) is used if NR is greater than or equal to 4. Note that for p and q both greater than 1, calling RNBET in a loop getting less than 4 variates on each call will not yield the same set of deviates as calling RNBET once and getting all the deviates at once.

The values returned in R are less than 1.0 and greater than ε, where ε is the smallest positive number such that $1.0 - \varepsilon$ is less than 1.0.

RNCHI

Generates pseudorandom numbers from a chi-squared distribution.

Required Arguments

DF — Degrees of freedom. (Input)
DF must be positive.

R — Vector of length NR containing the random chi-squared deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNCHI (DF, R [,…])`

Specific: The specific interface names are `S_RNCHI` and `D_RNCHI`.

FORTRAN 77 Interface

Single: `CALL RNCHI (NR, DF, R)`

Double: The double precision name is `DRNCHI`.

Example

In this example, RNCHI is used to generate five pseudorandom chi-squared deviates with 5 degrees of freedom.

```
      USE RNCHI_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    ISEED, NOUT, NR
      REAL       DF, R(5)
!
      CALL UMACH (2, NOUT)
```

```
      DF    = 5.0
      NR    = 5
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNCHI (DF, R)
      WRITE (NOUT,99999) R
99999 FORMAT ('  Chi-squared random deviates with 5 df: ', 5F7.3)
      END
```

Output

```
 Chi-squared random deviates with 5 df:  12.090  0.481  1.798 14.871  1.748
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNCHI generates pseudorandom numbers from a chi-squared distribution with DF degrees of freedom. If DF is an even integer less than 17, the chi-squared deviate r is generated as

$$r = -2\ln(\prod_{i=1}^{n} u_i)$$

where n = DF/2 and the u_i are independent random deviates from a uniform (0, 1) distribution. If DF is an odd integer less than 17, the chi-squared deviate is generated in the same way, except the square of a normal deviate is added to the expression above. If DF is greater than 16 or is not an integer, and if it is not too large to cause overflow in the gamma random number generator, the chi-squared deviate is generated as a special case of a gamma deviate, using routine RNGAM (page 1349). If overflow would occur in RNGAM, the chi-squared deviate is generated in the manner described above, using the logarithm of the product of uniforms, but scaling the quantities to prevent underflow and overflow.

RNCHY

Generates pseudorandom numbers from a Cauchy distribution.

Required Arguments

R — Vector of length NR containing the random Cauchy deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNCHY (R [,…])`

Specific: The specific interface names are `S_RNCHY` and `D_RNCHY`.

FORTRAN 77 Interface

Single: `CALL RNCHY (NR, R)`

Double: The double precision name is `DRNCHY`.

Example

In this example, `RNCHY` is used to generate five pseudorandom deviates from a Cauchy distribution.

```
      USE RNCHY_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    ISEED, NOUT, NR
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      NR    = 5
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNCHY (R)
      WRITE (NOUT,99999) R
99999 FORMAT ('      Cauchy random deviates: ', 5F8.4)
      END
```

Output

```
Cauchy random deviates:   3.5765  0.9353 15.5797  2.0815 -0.1333
```

Comments

The routine `RNSET` (page 1311) can be used to initialize the seed of the random number generator. The routine `RNOPT` (page 1311) can be used to select the form of the generator.

Description

Routine `RNCHY` generates pseudorandom numbers from a standard Cauchy distribution. The probability density function is

$$f(x) = \frac{1}{\pi\left(1+x^2\right)}$$

Use of the inverse CDF technique would yield a Cauchy deviate from a uniform (0, 1) deviate, u, as $\tan[\pi(u - .5)]$. Rather than evaluating a tangent directly, however, `RNCHY` generates two uniform (–1, 1) deviates, x_1 and x_2. These values can be thought of as sine and cosine values. If

$$x_1^2 + x_2^2$$

is less than or equal to 1, then x_1/x_2 is delivered as the Cauchy deviate; otherwise, x_1 and x_2 are rejected and two new uniform (–1, 1) deviates are generated. This method is also equivalent to taking the ratio of two independent normal deviates.

Deviates from the Cauchy distribution with median T and first quartile $T - S$, that is, with density

$$f(x) = \frac{S}{\pi\left[S^2 + (x-T)^2\right]}$$

can be obtained by scaling the output from RNCHY. The following statements (in single precision) would yield random deviates from this Cauchy distribution.

```
CALL RNCHY (NR, R)
CALL SSCAL (NR, S, R, 1)
CALL SADD (NR, T, R, 1)
```

The Cauchy distribution is a member of the symmetric stable family of distributions. The routine RNSTA (page 1362) can be used to generate deviates from this more general family of distributions or even from the stable family not requiring symmetry.

RNEXP

Generates pseudorandom numbers from a standard exponential distribution.

Required Arguments

R — Vector of length NR containing the random standard exponential deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNEXP (R [,…])`

Specific: The specific interface names are S_RNEXP and D_RNEXP.

FORTRAN 77 Interface

Single: `CALL RNEXP (NR, R)`

Double: The double precision name is DRNEXP.

Example

In this example, RNEXP is used to generate five pseudorandom deviates from a standard exponential distribution.

```
      USE RNEXP_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    ISEED, NOUT, NR
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      NR    = 5
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNEXP (R)
      WRITE (NOUT,99999) R
99999 FORMAT (’    Exponential random deviates: ’, 5F8.4)
      END
```

Output

```
Exponential random deviates:   0.0344  1.3443  0.2662  0.5633  0.1686
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNEXP generates pseudorandom numbers from a standard exponential distribution. The probability density function is $f(x) = e^{-x}$; for $x > 0$. RNEXP uses an antithetic inverse CDF technique; that is, a uniform random deviate U is generated and the inverse of the exponential cumulative distribution function is evaluated at $1.0 - U$ to yield the exponential deviate.

Deviates from the exponential distribution with mean THETA can be generated by using RNEXP and then multiplying each entry in R by THETA. The following statements (in single precision using the routine SSCAL (Reference Material)) would yield random deviates from such a distribution:

```
USE IMSL_LIBRARIES
        ⋮
CALL RNEXP (R, NR)
CALL SSCAL (NR, THETA, R, 1)
```

RNEXT

Generates pseudorandom numbers from a mixture of two exponential distributions.

Required Arguments

THETA1 — Mean of the exponential distribution that has the larger mean. (Input)

THETA2 — Mean of the exponential distribution that has the smaller mean. (Input)
THETA2 must be positive and less than or equal to THETA1.

P — Mixing parameter. (Input)
P must be nonnegative and less than or equal to THETA1/(THETA1 – THETA2).

R — Vector of length NR containing the random deviates from a mixture of exponentials. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNEXT (THETA1, THETA2, P, R [,…])`

Specific: The specific interface names are `S_RNEXT` and `D_RNEXT`.

FORTRAN 77 Interface

Single: `CALL RNEXT (NR, THETA1, THETA2, P, R)`

Double: The double precision name is `DRNEXT`.

Example

In this example, RNEXT is used to generate five pseudorandom deviates from a mixture of exponentials with means 2 and 1, respectively, and with mixing parameter 0.5.

```
      USE RNEXT_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    ISEED, NOUT, NR
      REAL       P, R(5), THETA1, THETA2
!
      CALL UMACH (2, NOUT)
      THETA1 = 2.0
      THETA2 = 1.0
      P      = 0.5
      NR     = 5
      ISEED  = 123457
      CALL RNSET (ISEED)
      CALL RNEXT (THETA1, THETA2, P, R)
      WRITE (NOUT,99999) R
```

```
99999 FORMAT (' Random deviates from a mixture of exponentials: ', /, &
             5X, 5F8.4)
      END
```

Output

```
Random deviates from a mixture of exponentials:
      0.0700  1.3024  0.6301  1.9756  0.3716
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNEXT generates pseudorandom numbers from a mixture of two exponential distributions. The probability density function is

$$f(x) = \frac{p}{\theta_1} e^{-x/\theta_1} + \frac{1-p}{\theta_2} e^{-x/\theta_2} \quad \text{for } x > 0$$

where p = P, θ_1 = THETA1, and θ_2 = THETA2.

In the case of a convex mixture, that is, the case $0 < p < 1$, the mixing parameter p is interpretable as a probability; and RNEXT with probability p generates an exponential deviate with mean θ_1, and with probability $1 - p$ generates an exponential with mean θ_2. When p is greater than 1, but less than $\theta_1/(\theta_1 - \theta_2)$, then either an exponential deviate with mean θ_1 or the sum of two exponentials with means θ_1 and θ_2 is generated. The probabilities are $q = p - (p-1)\theta_1/\theta_2$ and $1 - q$, respectively, for the single exponential and the sum of the two exponentials.

RNGAM

Generates pseudorandom numbers from a standard gamma distribution.

Required Arguments

A — The shape parameter of the gamma distribution. (Input)
This parameter must be positive.

R — Vector of length NR containing the random standard gamma deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNGAM (A, R [,…])`

Specific: The specific interface names are `S_RNGAM` and `D_RNGAM`.

FORTRAN 77 Interface

Single: `CALL RNGAM (NR, A, R)`

Double: The double precision name is `DRNGAM`.

Example

In this example, `RNGAM` is used to generate five pseudorandom deviates from a gamma (Erlang) distribution with shape parameter equal to 3.0.

```
      USE RNGAM_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    ISEED, NOUT
      REAL       A, R(NR)
!
      CALL UMACH (2, NOUT)
      A     = 3.0
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNGAM (A, R)
      WRITE (NOUT,99999) R
99999 FORMAT (’  Gamma(3) random deviates: ’, 5F8.4)
      END
```

Output

```
Gamma(3) random deviates:   6.8428  3.4452  1.8535  3.9992  0.7794
```

Comments

The routine `RNSET` (page 1311) can be used to initialize the seed of the random number generator. The routine `RNOPT` (page 1311) can be used to select the form of the generator.

Description

Routine `RNGAM` generates pseudorandom numbers from a gamma distribution with shape parameter a and unit scale parameter. The probability density function is

$$f(x) = \frac{1}{\Gamma(a)} x^{a-1} e^{-x} \qquad \text{for } x \geq 0$$

Various computational algorithms are used depending on the value of the shape parameter *a*. For the special case of $a = 0.5$, squared and halved normal deviates are used; and for the special case of $a = 1.0$, exponential deviates (from IMSL routine RNEXP, page 1346) are used. Otherwise, if *a* is less than 1.0, an acceptance-rejection method due to Ahrens, described in Ahrens and Dieter (1974), is used; if *a* is greater than 1.0, a ten-region rejection procedure developed by Schmeiser and Lal (1980) is used.

Deviates from the two-parameter gamma distribution with shape parameter a and scale parameter *b* can be generated by using RNGAM and then multiplying each entry in R by *b*. The following statements (in single precision) would yield random deviates from a gamma (*a*, *b*) distribution.

```
CALL RNGAM (NR, A, R)
CALL SSCAL (NR, B, R, 1)
```

The Erlang distribution is a standard gamma distribution with the shape parameter having a value equal to a positive integer; hence, RNGAM generates pseudorandom deviates from an Erlang distribution with no modifications required.

RNGCS

Sets up table to generate pseudorandom numbers from a general continuous distribution.

Required Arguments

CDF — User-supplied FUNCTION to compute the cumulative distribution function. The form is CDF(X), where

X — Point at which the distribution function is to be evaluated. (Input)
CDF — Value of the distribution function at X. (Output)
CDF must be declared EXTERNAL in the calling program.

IOPT — Indicator of the extent to which TABLE is initialized prior to calling RNGCS. (Input)

IOPT	Action
0	RNGCS fills the last four columns of TABLE. The user inputs the points at which the CDF is to be evaluated in the first column of TABLE. These must be in ascending order.
1	RNGCS fills the last three columns of TABLE. CDF is not used and may be a dummy function; instead, the cumulative distribution function is specified in the first two columns of TABLE. The abscissas (in the first column) must be in ascending order and the function must be strictly monotonically increasing.

TABLE — NDATA by 5 table to be used for interpolation of the cumulative distribution function. (Input and output)
The first column of TABLE contains abscissas of the cumulative distribution function in ascending order, the second column contains the values of the CDF (which must be

strictly increasing), and the remaining columns contain values used in interpolation. The first row of TABLE corresponds to the left limit of the support of the distribution and the last row corresponds to the right limit of the support; that is, TABLE(1, 2) = 0.0 and TABLE(NDATA, 2) = 1.0.

Optional Arguments

NDATA — Number of points at which the CDF is evaluated for interpolation. (Input)
NDATA must be greater than or equal to 4.
Default: NDATA = size (TABLE,1).

LDTABL — Leading dimension of TABLE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDTABL = size (TABLE,1).

FORTRAN 90 Interface

Generic: `CALL RNGCS (CDF, IOPT, TABLE [,…])`

Specific: The specific interface names are S_RNGCS and D_RNGCS.

FORTRAN 77 Interface

Single: `CALL RNGCS (CDF, IOPT, NDATA, TABLE, LDTABL)`

Double: The double precision name is DRNGCS.

Example

In this example, RNGCS is used to set up a table to generate pseudorandom variates from a beta distribution. This example is continued in the documentation for routine RNGCT (page 1354) to generate the random variates.

```
      USE RNGCS_INT
      USE UMACH_INT
      INTEGER    LDTABL
      PARAMETER  (LDTABL=100)
!
      INTEGER    I, IOPT, NINT, NOUT
      REAL       CDF, PIN, QIN, TABLE(LDTABL,5), X
      COMMON     /BCOM/ PIN, QIN
      EXTERNAL   CDF
!
      CALL UMACH (2, NOUT)
      PIN  = 3.0
      QIN  = 2.0
      IOPT = 0
      NINT = 100
      X    = 0.0
!                                Fill the first column of the table
```

```
!                                  with abscissas for interpolation.
      DO 10  I=1, NINT
         TABLE(I,1) = X
         X          = X + 0.01
   10 CONTINUE
      CALL RNGCS (CDF, IOPT, TABLE)
      WRITE (NOUT,99999) (TABLE(I,1),TABLE(I,2),I=1,10)
99999 FORMAT (’   First few elements of the table: ’, F4.2, F8.4, /, &
            (36X,F4.2,F8.4))
      END
!
!                                  Beta distribution function
      REAL FUNCTION CDF (X)
      REAL       X
!
      REAL       BETDF, PIN, QIN
      COMMON     /BCOM/ PIN, QIN
      EXTERNAL   BETDF
!
      CDF = BETDF(X,PIN,QIN)
      RETURN
      END
```

Output

```
*** WARNING  ERROR 1 from RNGCS.  The values of the CDF in the second
***          column of TABLE did not begin at 0.0 and end at 1.0, but they
***          have been adjusted. Prior to adjustment,
***          TABLE(1,2) = 0.000000E+00 and TABLE(NDATA,2) = 9.994079E-01.
  First few elements of the table: 0.00  0.0000
                                   0.01  0.0000
                                   0.02  0.0000
                                   0.03  0.0001
                                   0.04  0.0002
                                   0.05  0.0005
                                   0.06  0.0008
                                   0.07  0.0013
                                   0.08  0.0019
                                   0.09  0.0027
```

Comments

1. Informational error

Type	Code	
3	1	The values in TABLE(1, 2) and/or TABLE(NDATA, 2) are not exactly 0.0 and 1.0, respectively, but they are considered close enough to these values that they are set to these values.

2. The routine RNGCT (page 1354) uses the table set up by RNGCS to generate random numbers from the distribution with CDF represented in TABLE.

Description

Routine RNGCS sets up a table that routine RNGCT (page 1354) can use to generate pseudorandom deviates from a continuous distribution. The distribution is specified by its cumulative distribution function, which can be supplied either in tabular form in TABLE or by a FORTRAN function CDF. See the documentation for the routine RNGCT for a description of the method.

RNGCT

Generates pseudorandom numbers from a general continuous distribution.

Required Arguments

TABLE — NDATA by 5 table to be used for interpolation of the cumulative distribution function. (Input)
The first column of TABLE contains abscissas of the cumulative distribution function in ascending order, the second column contains the values of the CDF (which must be strictly increasing beginning with 0.0 and ending at 1.0) and the remaining columns contain values used in interpolation. This table is set up using routine RNGCS (page 1351).

R — Vector of length NR containing the random deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

NDATA — Number of points at which the cumulative distribution function is evaluated for interpolation. (Input)
NDATA must be greater than or equal to 4.
Default: NDATA = size (TABLE,1).

LDTABL — Leading dimension of TABLE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDTABL = size (TABLE,1).

FORTRAN 90 Interface

Generic: `CALL RNGCT (TABLE, R [,…])`

Specific: The specific interface names are S_RNGCT and D_RNGCT.

FORTRAN 77 Interface

Single: `CALL RNGCT (NR, NDATA, TABLE, LDTABL, R)`

Double: The double precision name is DRNGCT.

Example

In this example, RNGCS (page 1351) is used to set up a table for generation of beta pseudorandom deviates. The CDF for this distribution is computed by the routine BETDF (see Chapter 17, Probability Distribution Functions and Inverses). The table contains 100 points at which the CDF is evaluated and that are used for interpolation.

```
      USE RNGCT_INT
      USE UMACH_INT
      USE RNGCS_INT
      USE RNSET_INT
      INTEGER    LDTABL, NR
      PARAMETER  (LDTABL=100, NR=5)
!
      INTEGER    I, IOPT, ISEED, NINT, NOUT
      REAL       CDF, PIN, QIN, R(NR), TABLE(LDTABL,5), X
      COMMON     /BCOM/ PIN, QIN
      EXTERNAL   CDF
!
      CALL UMACH (2, NOUT)
      PIN  = 3.0
      QIN  = 2.0
      IOPT = 0
      NINT = 100
      X    = 0.0
!                                Fill the first column of the table
!                                with abscissas for interpolation.
      DO 10  I=1, NINT
         TABLE(I,1) = X
         X          = X + 0.01
   10 CONTINUE
      CALL RNGCS (CDF, IOPT, TABLE)
!                                Initialize seed of random number
!                                generator.
      ISEED = 123457
      CALL RNSET (ISEED)
!                                Now generate the random deviates.
      CALL RNGCT (TABLE, R)
      WRITE (NOUT,99999) R
99999 FORMAT (’  Beta (3,2) random deviates: ’, 5F7.4)
      END
!
!                                Beta distribution function
      REAL FUNCTION CDF (X)
      REAL       X
!
      REAL       BETDF, PIN, QIN
      COMMON     /BCOM/ PIN, QIN
      EXTERNAL   BETDF
!
      CDF = BETDF(X,PIN,QIN)
      RETURN
```

```
      END
```

Output

```
*** WARNING  ERROR 1 from RNGCS.  The values of the CDF in the second
***          column of TABLE did not begin at 0.0 and end at 1.0, but they
***          have been adjusted. Prior to adjustment,
***          TABLE(1,2) = 0.000000E+00 and TABLE(NDATA,2) = 9.994079E-01.
 Beta (3,2) random deviates:  0.9208 0.4641 0.7668 0.6536 0.8171
```

Comments

1. The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

2. In the interest of efficiency, this routine does only limited error checking. If TABLE is generated by the routine RNGCS (page 1351), the error checking is sufficient.

Description

Routine RNGCT generates pseudorandom numbers from a continuous distribution using the inverse CDF technique, by interpolation of points of the distribution function given in TABLE, which is set up by routine RNGCS (page 1351). A strictly monotone increasing distribution function is assumed. The interpolation is by an algorithm attributable to Akima (1970), using piecewise cubics. The use of this technique for generation of random numbers is due to Guerra, Tapia, and Thompson (1976), who give a description of the algorithm and accuracy comparisons between this method and linear interpolation. The relative errors using the Akima interpolation are generally considered very good.

RNLNL

Generates pseudorandom numbers from a lognormal distribution.

Required Arguments

XM — Mean of the underlying normal distribution. (Input)

S — Standard deviation of the underlying normal distribution. (Input)
S must be positive.

R — Vector of length NR containing the random lognormal deviates. (Output) The log of each element of R has a normal distribution with mean XM and standard deviation S.

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNLNL (XM, S, R [,…])`

Specific: The specific interface names are `S_RNLNL` and `D_RNLNL`.

FORTRAN 77 Interface

Single: `CALL RNLNL (NR, XM, S, R)`

Double: The double precision name is `DRNLNL`.

Example

In this example, `RNLNL` is used to generate five pseudorandom lognormal deviates with $\mu = 0$ and $\sigma = 1$.

```
      USE RNLNL_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    ISEED, NOUT
      REAL       R(NR), S, XM
!
      CALL UMACH (2, NOUT)
      XM    = 0.0
      S     = 1.0
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNLNL (XM, S, R)
      WRITE (NOUT,99999) R
99999 FORMAT (’  Lognormal random deviates: ’, 5F8.4)
      END
```

Output

```
Lognormal random deviates:   7.7801  2.9543  1.0861  3.5885  0.2935
```

Comments

The routine `RNSET` (page 1311) can be used to initialize the seed of the random number generator. The routine `RNOPT` (page 1311) can be used to select the form of the generator.

Description

Routine `RNLNL` generates pseudorandom numbers from a lognormal distribution with parameters `XM` and `S`. The scale parameter in the underlying normal distribution, `S`, must be positive. The method is to generate normal deviates with mean `XM` and standard deviation `S` and then to exponentiate the normal deviates.

With μ = XM and σ = S, the probability density function for the lognormal distribution is

$$f(x) = \frac{1}{\sigma x\sqrt{2\pi}} \exp\left[-\frac{1}{2\sigma^2}(\ln x - \mu)^2\right] \quad \text{for } x > 0$$

The mean and variance of the lognormal distribution are $\exp(\mu + \sigma^2/2)$ and $\exp(2\mu + 2\sigma^2) - \exp(2\mu + \sigma^2)$, respectively.

RNNOA

Generates pseudorandom numbers from a standard normal distribution using an acceptance/rejection method.

Required Arguments

R — Vector of length NR containing the random standard normal deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNNOA (R [,…])`

Specific: The specific interface names are `S_RNNOA` and `D_RNNOA`.

FORTRAN 77 Interface

Single: `CALL RNNOA (NR, R)`

Double: The double precision name is `DRNNOA`.

Example

In this example, RNNOA is used to generate five pseudorandom deviates from a standard normal distribution.

```
      USE RNNOA_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    ISEED, NOUT, NR
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      NR    = 5
      ISEED = 123457
      CALL RNSET (ISEED)
```

```
      CALL RNNOA (R)
      WRITE (NOUT,99999) R
99999 FORMAT (’  Standard normal random deviates: ’, 5F8.4)
      END
```

Output

```
Standard normal random deviates:   2.0516  1.0833  0.0826  1.2777 -1.2260
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNNOA generates pseudorandom numbers from a standard normal (Gaussian) distribution using an acceptance/rejection technique due to Kinderman and Ramage (1976). In this method, the normal density is represented as a mixture of densities over which a variety of acceptance/rejection methods due to Marsaglia (1964), Marsaglia and Bray (1964), and Marsaglia, MacLaren, and Bray (1964) are applied. This method is faster than the inverse CDF technique used in RNNOR (page 1361) to generate standard normal deviates.

Deviates from the normal distribution with mean XM and standard deviation XSTD can be obtained by scaling the output from RNNOA. The following statements (in single precision) would yield random deviates from a normal (XM, XSTD**2) distribution.

```
CALL RNNOA (NR, R)
CALL SSCAL (NR, XSTD, R, 1)
CALL SADD (NR, XM, R, 1)
```

RNNOF

This function generates a pseudorandom number from a standard normal distribution.

Function Return Value

RNNOF — Function value, a random standard normal deviate. (Output)
See Comment 1.

Required Arguments

None.

FORTRAN 90 Interface

Generic: `RNNOF ()`

Specific: The specific interface names are `S_RNNOF` and `D_RNNOF`.

FORTRAN 77 Interface

Single: `RNNOF ()`

Double: The double precision name is `DRNNOF`.

Example

In this example, `RNNOF` is used to generate five pseudorandom standard normal numbers.

```
      USE UMACH_INT
      USE RNSET_INT
      USE RNNOF_INT

      INTEGER    I, ISEED, NOUT, NR
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      ISEED = 123457
      CALL RNSET (ISEED)
      NR=5
      DO 10  I=1, NR
         R(I) = RNNOF()
   10 CONTINUE
      WRITE (NOUT,99999) R
99999 FORMAT (’  Standard normal random deviates: ’, 5F8.4)
      END
```

Output

```
Standard normal random deviates:   1.8279  -0.6412  0.7266  0.1747  1.0145
```

Comments

1. If the generic version of this function is used, the immediate result must be stored in a variable before use in an expression. For example:

   ```
   X = RNNOF()
   Y = SQRT(X)
   ```

   must be used rather than

   ```
   Y = SQRT(RNNOF())
   ```

   If this is too much of a restriction on the programmer, then the specific name can be used without this restriction.

2. The routine `RNSET` (page 1311) can be used to initialize the seed of the random number generator. The routine `RNOPT` (page 1311) can be used to select the form of the generator.

3. This function has a side effect: it changes the value of the seed, which is passed through a common block.

Description

Routine RNNOF is the function form of RNNOR. If several standard normal deviates are needed, it may be more efficient to obtain them all at once by a call to RNNOR, rather than by several references to RNNOF.

RNNOR

Generates pseudorandom numbers from a standard normal distribution using an inverse CDF method.

Required Arguments

R — Vector of length NR containing the random standard normal deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNNOR (R [,…])`

Specific: The specific interface names are S_RNNOR and D_RNNOR.

FORTRAN 77 Interface

Single: `CALL RNNOR (NR, R)`

Double: The double precision name is DRNNOR.

Example

In this example, RNNOR is used to generate five pseudorandom deviates from a standard normal distribution.

```
      USE RNNOR_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    ISEED, NOUT, NR
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      NR    = 5
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNNOR (R)
```

```
      WRITE (NOUT,99999) R
99999 FORMAT ('  Standard normal random deviates: ', 5F8.4)
      END
```

Output

```
Standard normal random deviates:   1.8279 -0.6412  0.7266  0.1747  1.0145
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNNOR generates pseudorandom numbers from a standard normal (Gaussian) distribution using an inverse CDF technique. In this method, a uniform (0,1) random deviate is generated and then the inverse of the normal distribution function is evaluated at that point, using the routine ANORIN (see Chapter 17, Probability Distribution Functions and Inverses). This method is slower than the acceptance/rejection technique used in the routine RNNOA (page 1358) to generate standard normal deviates. Deviates from the normal distribution with mean XM and standard deviation XSTD can be obtained by scaling the output from RNNOR. The following statements (in single precision, using the routines SSCAL (IMSL MATH/LIBRARY) and SADD (IMSL MATH/LIBRARY).) would yield random deviates from a normal (XM, XSTD**2) distribution.

```
USE IMSL_LIBRARIES

        ⋮

CALL RNNOR (R, NR)
CALL SSCAL (NR, XSTD, R, 1)
CALL SADD (NR, XM, R, 1)
```

RNSTA

Generates pseudorandom numbers from a stable distribution.

Required Arguments

ALPHA — Characteristic exponent of the stable distribution. (Input)
This parameter must be positive and less than or equal to 2.

BPRIME — Skewness parameter of the stable distribution. (Input)
When BPRIME = 0, the distribution is symmetric. Unless ALPHA = 1, BPRIME is not the usual skewness parameter of the stable distribution. BPRIME must be greater than or equal to – 1 and less than or equal to 1.

R — Vector of length NR containing the random stable deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNSTA (ALPHA, BPRIME, R [,…])`

Specific: The specific interface names are `S_RNSTA` and `D_RNSTA`.

FORTRAN 77 Interface

Single: `CALL RNSTA (NR, ALPHA, BPRIME, R)`

Double: The double precision name is `DRNSTA`.

Example

In this example, `RNSTA` is used to generate five pseudorandom symmetric stable variates with characteristic exponent 1.5. The tails of this distribution are heavier than those of a normal distribution, but not so heavy as those of a Cauchy distribution. The variance of this distribution does not exist, however. (This is the case for any stable distribution with characteristic exponent less than 2.)

```
      USE RNSTA_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    ISEED, NOUT
      REAL       ALPHA, BPRIM, R(NR)
!
      CALL UMACH (2, NOUT)
      ALPHA = 1.5
      BPRIM = 0.0
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNSTA (ALPHA, BPRIM, R)
      WRITE (NOUT,99999) R
99999 FORMAT (' Stable random deviates: ', 5F9.4)
      END
```

Output

```
Stable random deviates:    4.4091   1.0564   2.5463   5.6724   2.1656
```

Comments

The routine `RNSET` (page 1311) can be used to initialize the seed of the random number generator. The routine `RNOPT` (page 1311) can be used to select the form of the generator.

Description

Routine RNSTA generates pseudorandom numbers from a stable distribution with parameters ALPHA and BPRIME. ALPHA is the usual characteristic exponent parameter α and BPRIME is related to the usual skewness parameter β of the stable distribution. With the restrictions $0 < \alpha \leq 2$ and $-1 \leq \beta \leq 1$, the characteristic function of the distribution is

$$\varphi(t) = \exp[-|t|^{\alpha} \exp(-\pi i \beta(1 - |1 - \alpha|)\text{sign}(t)/2)] \qquad \text{for } \alpha \neq 1$$

and

$$\varphi(t) = \exp[-|t|(1 + 2i\beta \ln|t|)\text{sign}(t)/\pi)] \qquad \text{for } \alpha = 1$$

When $\beta = 0$, the distribution is symmetric. In this case, if $\alpha = 2$, the distribution is normal with mean 0 and variance 2; and if $\alpha = 1$, the distribution is Cauchy.

The parameterization using BPRIME and the algorithm used here are due to Chambers, Mallows, and Stuck (1976). The relationship between BPRIME = β' and the standard β is

$$\beta' = -\tan(\pi(1 - \alpha)/2) \tan(-\pi\beta(1 - |1 - \alpha|)/2) \qquad \text{for } \alpha \neq 1$$

and

$$\beta' = \beta \qquad \text{for } \alpha = 1$$

The algorithm involves formation of the ratio of a uniform and an exponential random variate.

RNSTT

Generates pseudorandom numbers from a Student's *t* distribution.

Required Arguments

DF — Degrees of freedom. (Input)
DF must be positive.

R — Vector of length NR containing the random Student's *t* deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNSTT (DF, R [,…])`

Specific: The specific interface names are `S_RNSTT` and `D_RNSTT`.

FORTRAN 77 Interface

Single: `CALL RNSTT (NR, DF, R)`

Double: The double precision name is DRNSTT.

Example

In this example, RNSTT is used to generate 5 pseudo-random *t* variates with 10 degrees of freedom.

```
      USE RNSTT_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    ISEED, NOUT
      REAL       DF, R(NR)
!
      CALL UMACH(2, NOUT)
      DF = 10.0
      ISEED = 123457
      CALL RNSET(ISEED)
      CALL RNSTT(DF, R)
      WRITE(NOUT, 99999) R
99999 FORMAT (’  t (10) random deviates: ’, 5F8.4)
      END
```

Output

```
t (10) random deviates:   0.6152  1.1528  0.0881  1.3382 -0.9893
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNSTT generates pseudo-random numbers from a Student’s *t* distribution with DF degrees of freedom, using a method suggested by Kinderman, Monahan, and Ramage (1977). The method (“TMX” in the reference) involves a representation of the *t* density as the sum of a triangular density over (–2, 2) and the difference of this and the *t* density. The mixing probabilities depend on the degrees of freedom of the *t* distribution. If the triangular density is chosen, the variate is generated as the sum of two uniforms; otherwise, an acceptance/rejection method is used to generate a variate from the difference density.

For degrees of freedom less than 100, RNSTT requires approximately twice the execution time as routine RNNOA (page 1358) which generates pseudorandom normal deviates. The execution time of RNSTT increases very slowly as the degrees of freedom increase. Since for very large degrees of freedom the normal distribution and the *t* distribution are very similar, the user may find that the difference in the normal and the *t* does not warrant the additional generation time required to use RNSTT instead of RNNOA.

RNTRI

Generates pseudorandom numbers from a triangular distribution on the interval (0, 1).

Required Arguments

R — Vector of length NR containing the random triangular deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNTRI (R [,…])`

Specific: The specific interface names are `S_RNTRI` and `D_RNTRI`.

FORTRAN 77 Interface

Single: `CALL RNTRI (NR, R)`

Double: The double precision name is `DRNTRI`.

Example

In this example, `RNTRI` is used to generate five pseudorandom deviates from a triangular distribution.

```
      USE RNTRI_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    ISEED, NOUT, NR
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      NR    = 5
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNTRI (R)
      WRITE (NOUT,99999) R
99999 FORMAT (’    Triangular random deviates: ’, 5F8.4)
      END
```

Output

```
Triangular random deviates:   0.8700  0.3610  0.6581  0.5360  0.7215
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNTRI generates pseudorandom numbers from a triangular distribution over the unit interval. The probability density function is $f(x) = 4x$, for $0 \le x \le .5$, and $f(x) = 4(1 - x)$, for $.5 < x \le 1$. RNTRI uses an inverse CDF technique.

RNVMS

Generates pseudorandom numbers from a von Mises distribution.

Required Arguments

C — Parameter of the von Mises distribution. (Input)
This parameter must be greater than one half of machine epsilon. (On many machines, the lower bound for C is 10^{-3}.)

R — Vector of length NR containing the random von Mises deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNVMS (C, R [,…])`

Specific: The specific interface names are S_RNVMS and D_RNVMS.

FORTRAN 77 Interface

Single: `CALL RNVMS (NR, C, R)`

Double: The double precision name is DRNVMS.

Example

In this example, RNVMS is used to generate five pseudorandom von Mises variates with $c = 1$.

```
USE RNVMS_INT
USE UMACH_INT
USE RNSET_INT
INTEGER    NR
PARAMETER  (NR=5)
```

```
!
      INTEGER    ISEED, NOUT
      REAL       C, R(NR)
!
      CALL UMACH (2, NOUT)
      C     = 1.0
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNVMS (C, R)
      WRITE (NOUT,99999) R
99999 FORMAT (’  Von Mises random deviates: ’, 5F8.4)
      END
```

Output

```
Von Mises random deviates:   0.2472 -2.4326 -1.0216 -2.1722 -0.5029
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNVMS generates pseudorandom numbers from a von Mises distribution with parameter C, which must be positive. With c = C, the probability density function is

$$f(x) = \frac{1}{2\pi I_0(c)} \exp\left[c\cos(x)\right] \text{ for } -\pi < x < \pi$$

where $I_0(c)$ is the modified Bessel function of the first kind of order 0. The probability density equals 0 outside the interval $(-\pi, \pi)$.

The algorithm is an acceptance/rejection method using a wrapped Cauchy distribution as the majorizing distribution. It is due to Best and Fisher (1979).

RNWIB

Generates pseudorandom numbers from a Weibull distribution.

Required Arguments

A — The shape parameter of the Weibull distribution. (Input)
This parameter must be positive.

R — Vector of length NR containing the random Weibull deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNWIB (A, R [,…])`

Specific: The specific interface names are `S_RNWIB` and `D_RNWIB`.

FORTRAN 77 Interface

Single: `CALL RNWIB (NR, A, R)`

Double: The double precision name is `DRNWIB`.

Example

In this example, `RNWIB` is used to generate five pseudorandom deviates from a two-parameter Weibull distribution with shape parameter equal to 2.0 and scale parameter equal to 6.0, a Rayleigh distribution with parameter

$$\alpha = 3\sqrt{2}$$

```
      USE RNWIB_INT
      USE UMACH_INT
      USE RNSET_INT
      USE SSCAL_INT
      INTEGER    ISEED, NOUT, NR
      REAL       A, B, R(5)
!
      CALL UMACH (2, NOUT)
      A     = 2.0
      B     = 6.0
      NR    = 5
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNWIB (A, R)
      CALL SSCAL (NR, B, R, 1)
      WRITE (NOUT,99999) R
99999 FORMAT (’      Weibull(2,6) random deviates: ’, 5F8.4)
      END
```

Output

```
Weibull(2,6) random deviates:   1.1122  6.9568  3.0959  4.5031  2.4638
```

Comments

1. Informational error

Type	Code	
3	1	The value of `A` is so small that the proportion of values from the Weibull that are too large to represent is greater than machine epsilon.

2. The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNWIB generates pseudorandom numbers from a Weibull distribution with shape parameter A and unit scale parameter. The probability density function is

$$f(x) = Ax^{A-1}e^{-x^A} \quad \text{for } x \geq 0$$

Routine RNWIB uses an antithetic inverse CDF technique to generate a Weibull variate; that is, a uniform random deviate U is generated and the inverse of the Weibull cumulative distribution function is evaluated at 1.0 – U to yield the Weibull deviate.

Deviates from the two-parameter Weibull distribution with shape parameter A and scale parameter B can be generated by using RNWIB and then multiplying each entry in R by B. The following statements (using routine SSCAL (IMSL MATH/LIBARY) in single precision) would yield random deviates from a two-parameter Weibull distribution.

```
CALL RNWIB (NR, A, R)
CALL SSCAL (NR, B, R, 1)
```

The Rayleigh distribution with probability density function,

$$r(x) = \frac{1}{\alpha^2} x\, e^{-\left(x^2/2\alpha^2\right)} \quad \text{for } x \geq 0$$

is the samep as a Weibull distribution with shape parameter A equal to 2 and scale parameter B equal to

$$\sqrt{2}\alpha$$

hence, RNWIB and SSCAL (or simple multiplication) can be used to generate Rayleigh deviates.

RNCOR

Generates a pseudorandom orthogonal matrix or a correlation matrix.

Required Arguments

A — N by N random orthogonal matrix. (Output, if IOPT = 0; workspace if IOPT = 1; input/output, if IOPT = 2. If IOPT = 2, A is destroyed.)

Optional Arguments

N — The order of the matrices to be generated. (Input)
N must be at least two.
Default: N = size (A,2).

IOPT — Option indicator. (Input)
Default: IOPT = 0.

IOPT	Action
0	A random orthogonal matrix is generated in A.
1	A random correlation matrix is generated in COR. (A is used as workspace.)
2	A random correlation matrix is generated in COR using the orthogonal matrix input in A.

EV — If IOPT = 1 or 2, a vector of length N containing the eigenvalues of the correlation matrix to be generated. (Input, if IOPT = 1 or 2; not used otherwise.)
The elements of EV must be positive, they must sum to N, and they cannot all be equal.

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

COR — N by N random correlation matrix. (Output, if IOPT = 1 or 2; not used otherwise.)

LDCOR — Leading dimension of COR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOR=size(COR, 1)

FORTRAN 90 Interface

Generic: `CALL RNCOR (A [,…])`

Specific: The specific interface names are S_RNCOR and D_RNCOR.

FORTRAN 77 Interface

Single: `CALL RNCOR (N, IOPT, EV, A, LDA, COR, LDCOR)`

Double: The double precision name is DRNCOR.

Example

In this example, RNCOR is used to generate a 4 by 4 pseudorandom correlation matrix with eigenvalues in the ratio 1:2:3:4. (Note that the eigenvalues must sum to 4.) Routines MXTXF (IMSL MATH/LIBRARY) and EVCSF (IMSL MATH/LIBRARY) are used to check the output.

```
USE IMSL_LIBRARIES
INTEGER    I, IOPT, ISEED, J, LDA, LDCOR, N, NOUT
REAL       A(4,4), COR(4,4), EV(4), EVAL(4), EVEC(4,4), FLOAT, &
           SUM, XID(4,4)
```

```
      INTRINSIC  FLOAT
!
      CALL UMACH (2, NOUT)
      N     = 4
      LDA   = 4
      LDCOR = 4
      EV(1) = 1.0
      EV(2) = 2.0
      EV(3) = 3.0
      EV(4) = 4.0
!                                Scale the eigenvalues to sum to N.
      SUM = SSUM(N,EV,1)
      CALL SSCAL (N, FLOAT(N)/SUM, EV, 1)
      ISEED = 123457
      CALL RNSET (ISEED)
!                                Generate an orthogonal matrix.
      CALL RNCOR (A)
      WRITE (NOUT,99996) ((A(I,J),J=1,N),I=1,N)
99996 FORMAT (' A random orthogonal matrix: ', /, (5X,4F8.4))
!                                Check it for orthogonality.
      CALL MXTXF (A, XID)
      WRITE (NOUT,99997) ((XID(I,J),J=1,N),I=1,N)
99997 FORMAT (' The identity matrix?:       ', /, (5X,4F8.4))
!
!                                Now get a correlation matrix using
!                                the orthogonal matrix in A, which
!                                will be destroyed.
      IOPT = 2
      CALL RNCOR (A,IOPT=IOPT, EV=EV, COR=COR)
      WRITE (NOUT,99998) ((COR(I,J),J=1,N),I=1,N)
99998 FORMAT (' A random correlation matrix: ', /, (5X,4F8.4))
!                                Check the eigenvalues.
      CALL EVCSF (COR, EVAL, EVEC)
      WRITE (NOUT,99999) (EVAL(I),I=1,N)
99999 FORMAT (' The computed eigenvalues:', 4F8.4)
      END
```

Output

```
A random orthogonal matrix:
      -0.8804 -0.2417  0.4065 -0.0351
       0.3088 -0.3002  0.5520  0.7141
      -0.3500  0.5256 -0.3874  0.6717
      -0.0841 -0.7584 -0.6165  0.1941
The identity matrix?:
       1.0000  0.0000  0.0000  0.0000
       0.0000  1.0000  0.0000  0.0000
       0.0000  0.0000  1.0000  0.0000
       0.0000  0.0000  0.0000  1.0000
A random correlation matrix:
       1.0000 -0.2358 -0.3258 -0.1101
      -0.2358  1.0000  0.1906 -0.0172
      -0.3258  0.1906  1.0000 -0.4353
      -0.1101 -0.0172 -0.4353  1.0000
The computed eigenvalues:  1.6000  1.2000  0.8000  0.4000
```

Comments

1. Workspace may be explicitly provided, if desired, by use of R2COR/DR2COR. The reference is:

   ```
   CALL R2COR (N, IOPT, EV, A, LDA, COR, LDCOR, IWK, WK)
   ```

   The additional arguments are as follows:

   IWK — Work vector of length 3 * N.

   WK — Work vector of length N.

2. Informational error

Type	Code	
3	1	Considerable loss of precision occurred in the rotations used to form the correlation matrix. Some of the diagonals of COR differ from 1.0 by more than the machine epsilon.

3. The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNCOR generates a pseudorandom orthogonal matrix A from the invariant Haar measure. For each column of A, a random vector from a uniform distribution on a hypersphere is selected and then is projected onto the orthogonal complement of the columns of A already formed. The method is described by Heiberger (1978). (See also Tanner and Thisted 1982.)

A correlation matrix is formed by applying a sequence of planar rotations to the matrix $A^T DA$, where D = diag(EV(1), …,EV(N)), so as to yield ones along the diagonal. The planar rotations are applied in such an order that in the two by two matrix that determines the rotation, one diagonal element is less than 1.0 and one is greater than 1.0. This method is discussed by Bendel and Mickey (1978) and by Lin and Bendel (1985).

The distribution of the correlation matrices produced by this method is not known. Bendel and Mickey (1978) and Johnson and Welch (1980) discuss the distribution.

For larger matrices, rounding can become severe; and the double precision results may differ significantly from single precision results.

RNDAT

Generates pseudorandom numbers from a multivariate distribution determined from a given sample.

Required Arguments

X — NSAMP by K matrix containing the given sample. (Input/Output)
If IDO = 0 or 1, on output the rows of X are rearranged by routine QUADT to form a *k–d* tree.

NN — Number of nearest neighbors of the randomly selected point in X that are used to form the output point in R. (Input)

R — NR by K matrix containing the random multivariate vectors in its rows. (Output)

Optional Arguments

IDO — Generator option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of RNDAT with the sample in X and all desired pseudorandom numbers are to be generated in this call.
1	This is the first invocation, and additional calls to RNDAT will be made to generate additional random numbers using the same given sample.
2	This is an intermediate invocation of RNDAT. The work vectors have been set up in a previous call, but they are not to be released because additional calls will be made.
3	This is the final invocation of RNDAT. The work vectors have been set up in a previous call and they are to be released.

NR — Number of random multivariate vectors to generate. (Input)
If NR = 0, only initialization or wrap up operations are performed. (This would make sense only if IDO = 1 or 3.)
Default: NR = size (R,1).

K — The length of the multivariate vectors, that is, the number of dimensions. (Input)
Default: K = size (R,2).

NSAMP — Number of given data points from the distribution to be simulated. (Input)
Default: NSAMP = size (X,1).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

LDR — Leading dimension of R exactly as specified in the dimension statement of the calling program. (Input)
Default: LDR = size (R, 1).

FORTRAN 90 Interface

Generic: `CALL RNDAT (X, NN, R [,…])`

Specific: The specific interface names are S_RNDAT and D_RNDAT.

FORTRAN 77 Interface

Single: `CALL RNDAT (IDO, NR, K, NSAMP, X, LDX, NN, R, LDR)`

Double: The double precision name is DRNDAT.

Example

In this example, RNDAT is used to generate 5 pseudorandom vectors of length 4 using the initial and final systolic pressure and the initial and final diastolic pressure from Data Set A in Afifi and Azen (1979) as the fixed sample from the population to be modeled. (Values of these four variables are in the seventh, tenth, twenty-first, and twenty-fourth columns of data set number nine in routine GDATA, page 1302)

```
      USE IMSL_LIBRARIES
      INTEGER    LDR, LDRDAT, LDX, NDR, NDRDAT, NDX
      PARAMETER  (LDR=5, LDRDAT=113, LDX=113, NDR=4, NDRDAT=34, NDX=4)
!
      INTEGER    ISEED, NN, NRCOL, NRROW
      REAL       R(LDR,NDR), RDATA(LDRDAT,NDRDAT), X(LDX,NDX)
      CHARACTER * 6 NUMBER(1)
!                                Afifi and Azen Data Set A
      DATA NUMBER(1)/NUMBER/
      CALL GDATA (9, RDATA, NRROW, NRCOL)
      CALL SCOPY (NRROW, RDATA(1:,7), 1,  X(1:,1), 1)
      CALL SCOPY (NRROW, RDATA(1:,10), 1, X(1:,2), 1)
      CALL SCOPY (NRROW, RDATA(1:,21), 1, X(1:,3), 1)
      CALL SCOPY (NRROW, RDATA(1:,24), 1, X(1:,4), 1)
!
      ISEED = 123457
      CALL RNSET (ISEED)
!                                Set input values
      NN    = 5
!                                Generate random variates
      CALL RNDAT (X, NN, R)
!                                Print results
      CALL WRRRL ('Random variates', R, NUMBER, NUMBER, FMT='(F15.4)')
!
      END
```

Output

```
                         Random variates
              1              2              3              4
1      162.7668        90.5057       153.7173       104.8768
2      153.3533        78.3180       176.6643        85.2155
3       93.6958        48.1675       153.5495        71.3688
4      101.7508        54.1855       113.1215        56.2916
5       91.7403        58.7684        48.4368        28.0994
```

Comments

1. Workspace may be explicitly provided, if desired, by use of R2DAT/DR2DAT. The reference is:

   ```
   CALL R2DAT (IDO, NR, K, NSAMP, X, LDX, NN, R, LDR, IWK, WK)
   ```

   The additional arguments are as follows:

   IWK — Work vector of length equal to 2 * NSAMP + 3 * LEN + K + NN.

   WK — Work vector of length equal to 2 * NSAMP + 2 * K * LEN + K + 2 * NN.

   R2DAT allows alternating calls for two different populations (see Comment 3). **Warning:** R2DAT does no error checking.

2. The rows of X are rearranged on output from either RNDAT or R2DAT.

3. When more than one call is to be made to RNDAT to generate more than one R matrix using the same sample in X, IDO should be set to 1 for the first call, to 2 for all subsequent calls except the last one, and to 3 for the last call. If more than one population is to be simulated (that is, there is more than one sample, X), it is necessary to generate all of the observations from each population at one time because data is stored in the work vectors. If the user provides work vectors for each population to be simulated, R2DAT can be used to simulate different population alternatively.

4. The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Given a sample of size n (= NSAMP) of observations of a k-variate random variable, RNDAT generates a pseudorandom sample with approximately the same moments as the given sample. The sample obtained is essentially the same as if sampling from a Gaussian kernel estimate of the sample density. (See Thompson 1989.) Routine RNDAT uses methods described by Taylor and Thompson (1986).

Assume that the (vector-valued) observations x_i are in the rows of X. An observation, x_j, is chosen randomly; its nearest m (= NN) neighbors,

$$x_{j_1}, x_{j_2}, \ldots, x_{j_m}$$

are determined; and the mean

$$\bar{x}_j$$

of those nearest neighbors is calculated. Next, a random sample

$u_1, u_2, \ldots, u_m$ is generated from a uniform distribution with lower bound

$$\frac{1}{m} - \sqrt{\frac{3(m-1)}{m^2}}$$

and upper bound

$$\frac{1}{m} + \sqrt{\frac{3(m-1)}{m^2}}$$

The random variate delivered is

$$\sum_{l=1}^{m} u_l \left(x_{jl} - \bar{x}_j \right) + \bar{x}_j$$

The process is then repeated until NR such simulated variates are generated and stored in the rows of R.

When RNDAT is invoked for the first time for a given sample, a search tree is computed for the rows of X. During the generation process, this tree is used to find the nearest neighbors of the randomly selected row. The argument IDO is used to determine whether or not the tree must be computed and whether workspace has to be allocated to store the tree.

RNMTN

Generates pseudorandom numbers from a multinomial distribution.

Required Arguments

N — Multinomial parameter indicating the number of independent trials. (Input)

P — Vector of length K containing the probabilities of the possible outcomes. (Input)
The elements of P must be positive and must sum to 1.0.

IR — NR by K matrix containing the random multinomial vectors in its rows. (Output)

Optional Arguments

NR — Number of random multinomial vectors to generate. (Input)
Default: NR = size (IR,1).

K — The number of mutually exclusive outcomes on any trial. (Input)
K is the length of the multinomial vectors. K must be greater than or equal to 2.
Default: K = size (IR,2).

LDIR — Leading dimension of IR exactly as specified in the dimension statement of the calling program. (Input)
Default: LDIR = size (IR,1).

FORTRAN 90 Interface

Generic: `CALL RNMTN (N, P, IR [,…])`

Specific: The specific interface name is `S_RNMTN`.

FORTRAN 77 Interface

Single: `CALL RNMTN (NR, N, K, P, IR, LDIR)`

Example

In this example, RNMTN is used to generate five pseudorandom 3-dimensional multinomial variates with parameters N = 20 and P = (0.1, 0.3, 0.6).

```
      USE RNMTN_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    K, LDIR
      PARAMETER  (K=3, LDIR=5)
!
      INTEGER    I, IR(LDIR,K), ISEED, J, N, NOUT, NR
      REAL       P(K)
!
      CALL UMACH (2, NOUT)
      N     = 20
      NR    = 5
      P(1)  = 0.1
      P(2)  = 0.3
      P(3)  = 0.6
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNMTN (N, P, IR)
      WRITE (NOUT,99999) ((IR(I,J),J=1,K),I=1,NR)
99999 FORMAT (' Multinomial random deviates: ', 3I4, /, (30X,3I4))
      END
```

Output

```
Multinomial random deviates:    5   4  11
                                3   6  11
                                3   3  14
                                5   5  10
                                4   5  11
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNMTN generates pseudorandom numbers from a K-variate multinomial distribution with parameters N and P. K and N must be positive. Each element of P must be positive and the elements must sum to 1. The probability function (with n = N, k = K, and p_i = P(I)) is

$$f(x_1, x_2, \ldots, x_k) = \frac{n!}{x_1!x_2!\ldots x_k!} p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k}$$

for $x_i \geq 0$ and

$$\sum_{i=1}^{k} x_i = n$$

The deviate in each row of IR is produced by generation of the binomial deviate x_1 with parameters n and p_i and then by successive generations of the conditional binomial deviates x_j given $x_1, x_2, \ldots, x_{j-1}$ with parameters $n - x_1 - x_2 - \ldots - x_{j-1}$ and $p_j/(1 - p_1 - p_2 - \ldots - p_{j-1})$.

RNMVN

Generates pseudorandom numbers from a multivariate normal distribution.

Required Arguments

RSIG — Upper triangular matrix, K by K, containing the Cholesky factor of the variance-covariance matrix. (Input)
The variance-covariance matrix is equal to the product of the transpose of RSIG and RSIG. RSIG can be obtained from the variance-covariance matrix using routine CHFAC.

R — NR by K matrix containing the random multivariate normal vectors in its rows. (Output)

Optional Arguments

NR — Number of random multivariate normal vectors to generate. (Input)
Default: NR = size (R,1).

K — Length of the multivariate normal vectors. (Input)
Default: K = size (R,2).

LDRSIG — Leading dimension of RSIG exactly as specified in the dimension statement in the calling program. (Input)
Default: LDRSIG = size (RSIG,1).

LDR — Leading dimension of R exactly as specified in the dimension statement of the calling program. (Input)
Default: LDR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL RNMVN (RSIG, R [,…])`

Specific: The specific interface names are S_RNMVN and D_RNMVN.

FORTRAN 77 Interface

Single: `CALL RNMVN (NR, K, RSIG, LDRSIG, R, LDR)`

Double: The double precision name is DRNMVN.

Example

In this example, RNMVN is used to generate five pseudorandom multivariate normal vectors of length 2 with variance-covariance matrix equal to

$$\begin{matrix} 0.500 & 0.375 \\ 0.375 & 0.500 \end{matrix}$$

The routine CHFAC is first called to compute the Cholesky factorization of the variance-covariance matrix.

```
      USE RNMVN_INT
      USE UMACH_INT
      USE CHFAC_INT
      USE RNSET_INT
      INTEGER    I, IRANK, ISEED, J, K, LDR, LDRSIG, NOUT, NR
      REAL       COV(2,2), R(5,2), RSIG(2,2)
!
      CALL UMACH (2, NOUT)
      NR       = 5
      K        = 2
      LDRSIG   = 2
      LDR      = 5
      COV(1,1) = 0.5
      COV(1,2) = 0.375
      COV(2,1) = 0.375
      COV(2,2) = 0.5
!                                Obtain the Cholesky factorization.
      CALL CHFAC (COV, IRANK, RSIG)
!                                Initialize seed of random number
!                                generator.
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNMVN (RSIG, R)
      WRITE (NOUT,99999) ((R(I,J),J=1,K),I=1,NR)
```

```
99999 FORMAT ('    Multivariate normal random deviates: ', /, &
             (1X,2F8.4))
      END
```

Output

```
 Multivariate normal random deviates:
 1.4507  1.2463
 0.7660 -0.0429
 0.0584 -0.6692
 0.9035  0.4628
-0.8669 -0.9334
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNMVN generates pseudorandom numbers from a multivariate normal distribution with mean vector consisting of all zeroes and variance-covariance matrix whose Cholesky factor (or "square root") is RSIG; that is, RSIG is an upper triangular matrix such that the transpose of RSIG times RSIG is the variance-covariance matrix. First, independent random normal deviates with mean 0 and variance 1 are generated, and then the matrix containing these deviates is postmultiplied by RSIG. The independent normals are generated into the columns of a matrix, which has NR rows; hence, if RNSET is called with different values of NR, the output is different even if the seed is the same in the calls.

Deviates from a multivariate normal distribution with means other than zero can be generated by using RNMVN and then by adding the vector of means to each row of R.

RNSPH

Generates pseudorandom points on a unit circle or K-dimensional sphere.

Required Arguments

Z — NR by K matrix containing the random Cartesian coordinates on the unit circle or sphere. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (Z,1).

K — Dimension of the circle (K = 2) or of the sphere. (Input)
Default: K = size (Z,2).

LDZ — Leading dimension of Z exactly as specified in the dimension statement of the calling program. (Input)
Default: LDZ = size (Z,1).

FORTRAN 90 Interface

Generic: `CALL RNSPH (Z [,…])`

Specific: The specific interface names are `S_RNSPH` and `D_RNSPH`.

FORTRAN 77 Interface

Single: `CALL RNSPH (NR, K, Z, LDZ)`

Double: The double precision name is `DRNSPH`.

Example

In this example, `RNSPH` is used to generate two uniform random deviates from the surface of the unit sphere in three space.

```
      USE UMACH_INT
      USE RNSET_INT
      USE RNSPH_INT
      INTEGER    K, LDZ
      PARAMETER  (K=3, LDZ=2)
!
      INTEGER    I, ISEED, J, NOUT
      REAL       Z(LDZ,K)
!
      CALL UMACH (2, NOUT)
      NR    = 2
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNSPH (Z)
      WRITE (NOUT,99999) ((Z(I,J),J=1,K),I=1,NR)
99999 FORMAT ('      Coordinates of first point: ', 3F8.4, /, &
             '      Coordinates of second point:', 3F8.4)
      END
```

Output

```
Coordinates of first point:   0.8893  0.2316  0.3944
Coordinates of second point:  0.1901  0.0396 -0.9810
```

Comments

The routine `RNSET` (page 1311) can be used to initialize the seed of the random number generator. The routine `RNOPT` (page 1311) can be used to select the form of the generator.

Description

Routine RNSPH generates pseudorandom coordinates of points that lie on a unit circle or a unit sphere in K-dimensional space. For points on a circle (K = 2), pairs of uniform (– 1, 1) points are generated and accepted only if they fall within the unit circle (the sum of their squares is less than 1), in which case they are scaled so as to lie on the circle.

For spheres in three or four dimensions, the algorithms of Marsaglia (1972) are used. For three dimensions, two independent uniform (– 1, 1) deviates U_1 and U_2 are generated and accepted only if the sum of their squares S_1 is less than 1. Then, the coordinates

$$Z_1 = 2U_1\sqrt{1-S_1},\ Z_2 = 2U_2\sqrt{1-S_1},\ \text{and}\ Z_3 = 1-2S_1$$

are formed. For four dimensions, U_1, U_2, and S_1 are produced as described above. Similarly, U_3, U_4, and S_2 are formed. The coordinates are then

$$Z_1 = U_1,\ Z_2 = U_2,\ Z_3 = U_3\sqrt{(1-S_1)/S_2}$$

and

$$Z_4 = U_4\sqrt{(1-S_1)/S_2}$$

For spheres in higher dimensions, K independent normal deviates are generated and scaled so as to lie on the unit sphere in the manner suggested by Muller (1959).

RNTAB

Generates a pseudorandom two-way table.

Required Arguments

NRTOT — Vector of length NROW containing the row totals. (Input)

NCTOT — Vector of length NCOL containing the column totals. (Input)
The elements of NRTOT and NCTOT must be nonnegative and must sum to the same quantity.

ITAB — NROW by NCOL random matrix with the given row and column totals. (Output)

Optional Arguments

IDO — Generator option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of RNTAB with these input specifications of the two-way table.
1	This is the first invocation, and additional calls to RNTAB will be made to generate random tables with the same specifications.
2	This is an intermediate invocation of RNTAB. The work vectors have been set up in a previous call, but they are not to be released because additional calls will be made.
3	This is the final invocation of RNTAB. The work vectors have been set up in a previous call and they are to be released.

NROW — Number of rows in the table. (Input)
Default: NROW = size (ITAB,1).

NCOL — Number of columns in the table. (Input)
Default: NCOL = size (ITAB,2).

LDITAB — Leading dimension of ITAB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDITAB = size (ITAB,1).

FORTRAN 90 Interface

Generic: `CALL RNTAB (NRTOT, NCTOT, ITAB [,…])`

Specific: The specific interface name is S_RNTAB.

FORTRAN 77 Interface

Single: `CALL RNTAB (IDO, NROW, NCOL, NRTOT, NCTOT, ITAB, LDITAB)`

Example

In this example, RNTAB is used to generate a two by three table with row totals 3 and 5, and column totals 2, 4, and 2.

```
      USE RNTAB_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    I, ISEED, ITAB(2,3), IWK, J, NCTOT(3), NOUT,&
                 NRTOT(2)
!
      CALL UMACH (2, NOUT)
      NROW = 2
      NCOL = 3
```

```
      NRTOT(1) = 3
      NRTOT(2) = 5
      NCTOT(1) = 2
      NCTOT(2) = 4
      NCTOT(3) = 2
      ISEED    = 123457
      CALL RNSET (ISEED)
      CALL RNTAB (NRTOT, NCTOT, ITAB)
      WRITE (NOUT,99999) ((ITAB(I,J),J=1,NCOL),I=1,NROW)
99999 FORMAT (' A random contingency table with fixed marginal totals:' &
            , /, (5X,3I5))
      END
```

Output

```
A random contingency table with fixed marginal totals:
        0    2    1
        2    2    1
```

Comments

1. Let IRSUM = the sum of the elements in NRTOT. If IRSUM + 1 is less than 2 * NROW * NCOL, automatic workspace usage is IRSUM; otherwise, automatic workspace usage is 2 * IRSUM + 1 because a different algorithm is used. Workspace may be explicitly provided, if desired, by use of R2TAB. R2TAB allows selection of the algorithm to be used and it allows alternating calls for two different problems (see Comment 3). The reference is:

   ```
   CALL R2TAB (IDO, NROW, NCOL, NRTOT, NCTOT, ITAB, LDITAB, IOPT,
   IRSUM, IWK, WK).
   ```

   The additional arguments are as follows:

   IOPT — Option indicator. (Input)
   If IOPT = 1, Boyette's method is used.
   If IOPT = 2, Patefield's method is used.

   IRSUM — Sum of the elements in NRTOT. (Output)

   IWK — Work vector of length equal to the sum of the elements in NRTOT.

   WK — Work vector of length equal to the sum of the elements in NRTOT plus one, used only if IOPT = 2.
   WARNING: R2TAB does no error checking.

2. Informational error

Type	Code	
3	1	The values of NRTOT and/or of NCTOT are such that the probability distribution of tables is degenerate, that is, only one such table is possible.

3. When more than one table with the same marginal totals is to be generated, IDO should be set to 1 for the first call, to 2 for all subsequent calls except the last one, and to 3 for the last call. If several tables of different sizes or with different marginal totals are to be generated, it is necessary to generate all of each type together because of the data stored in the work vectors. If the user provides work vectors for each type of table to be generated, R2TAB can be used to generate different types of tables alternatively.

4. The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNTAB generates pseudorandom entries for a two-way contingency table with fixed row and column totals. The method depends on the size of the table and the total number of entries in the table. If the total number of entries is less than twice the product of the number of rows and columns, the method described by Boyette (1979) and by Agresti, Wackerly, and Boyette (1979) is used. In this method, a work vector is filled with row indices so that the number of times each index appears equals the given row total. This vector is then randomly permuted and used to increment the entries in each row so that the given row total is attained.

For tables with larger numbers of entries, the method of Patefield (1981) is used. This method can be considerably faster in these cases. The method depends on the conditional probability distribution of individual elements, given the entries in the previous rows. The probabilities for the individual elements are computed starting from their conditional means.

On the first call to RNTAB with a given set of row and column totals, certain checking is done, and the work vector is allocated and initialized. On the final call, the work vector is released. The argument IDO indicates the nature of the call. In a simulation study, RNTAB would typically be called first with IDO = 1, then would be called several times with IDO = 2, and then finally would be called with IDO = 3. If only one table is needed, IDO should be set to 0.

RNNOS

Generates pseudorandom order statistics from a standard normal distribution.

Required Arguments

IFIRST — First order statistic to generate. (Input)

ILAST — Last order statistic to generate. (Input)
ILAST must be greater than or equal to IFIRST. The full set of order statistics from IFIRST to ILAST is generated. If only one order statistic is desired, set ILAST = IFIRST.

N — Size of the sample from which the order statistics arise. (Input)

R — Vector of length ILAST + 1 – IFIRST containing the random order statistics in ascending order. (Output)
The first element of R is the IFIRST-th order statistic in a random sample of size N from the standard normal distribution.

FORTRAN 90 Interface

Generic: `CALL RNNOS (IFIRST, ILAST, N, R)`

Specific: The specific interface names are S_RNNOS and D_RNNOS.

FORTRAN 77 Interface

Single: `CALL RNNOS (IFIRST, ILAST, N, R)`

Double: The double precision name is DRNNOS.

Example

In this example, RNNOS is used to generate the fifteenth through the nineteenth order statistics from a sample of size twenty.

```
      USE RNNOS_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    IFIRST, ILAST, ISEED, N, NOUT
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      IFIRST = 15
      ILAST  = 19
      N      = 20
!                                Initialize seed of random number
!                                generator.
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNNOS (IFIRST, ILAST, N, R)
      WRITE (NOUT,99999) R
99999 FORMAT (’  The 15th through the 19th order statistics from a’, &
            /, ’  random sample of size 20 from a normal distribution’ &
            , /, 5F8.4)
      END
```

Output

```
The 15th through the 19th order statistics from a
random sample of size 20 from a normal distribution
0.4056  0.4681  0.4697  0.9067  0.9362
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNNOS generates the IFIRST through the ILAST order statistics from a pseudorandom sample of size N from a normal (0, 1) distribution. Routine RNNOS uses the routine RNUNO (page 1388) to generate order statistics from the uniform (0, 1) distribution and then obtains the normal order statistics using the inverse CDF transformation.

Each call to RNNOS yields an independent event so order statistics from different calls may not have the same order relations with each other.

RNUNO

Generates pseudorandom order statistics from a uniform (0, 1) distribution.

Required Arguments

IFIRST — First order statistic to generate. (Input)

ILAST — Last order statistic to generate. (Input)
ILAST must be greater than or equal to IFIRST. The full set of order statistics from IFIRST to ILAST is generated. If only one order statistic is desired, set ILAST = IFIRST.

N — Size of the sample from which the order statistics arise. (Input)

R — Vector of length ILAST + 1 – IFIRST containing the random order statistics in ascending order. (Output)
The first element of R is the IFIRST-th order statistic in a random sample of size N from the uniform (0, 1) distribution.

FORTRAN 90 Interface

Generic: `CALL RNUNO (IFIRST, ILAST, N, R)`

Specific: The specific interface names are S_RNUNO and D_RNUNO.

FORTRAN 77 Interface

Single: `CALL RNUNO (IFIRST, ILAST, N, R)`

Double: The double precision name is DRNUNO.

Example

In this example, RNUNO is used to generate the fifteenth through the nineteenth order statistics from a sample of size twenty.

```
      USE RNUNO_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    IFIRST, ILAST, ISEED, N, NOUT
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      IFIRST = 15
      ILAST  = 19
      N      = 20
!                                Initialize seed of random number
!                                generator.
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNUNO (IFIRST, ILAST, N, R)
      WRITE (NOUT,99999) R
99999 FORMAT (’  The 15th through the 19th order statistics from a’, &
            /, ’  random sample of size 20 from a uniform ’, &
            ’distribution’, /, 5F8.4)
      END
```

Output

```
  The 15th through the 19th order statistics from a
  random sample of size 20 from a uniform distribution
  0.6575  0.6802  0.6807  0.8177  0.8254
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNUNO generates the IFIRST through the ILAST order statistics from a pseudorandom sample of size N from a uniform (0, 1) distribution. Depending on the values of IFIRST and ILAST, different methods of generation are used to achieve greater efficiency. If IFIRST = 1 and ILAST = N, that is, if the full set of order statistics are desired, the spacings between successive order statistics are generated as ratios of exponential variates. If the full set is not desired, a beta variate is generated for one of the order statistics, and the others are generated as extreme order statistics from conditional uniform distributions. Extreme order statistics from a uniform distribution can be obtained by raising a uniform deviate to an appropriate power.

Each call to RNUNO yields an independent event. This means, for example, that if on one call the fourth order statistic is requested and on a second call the third order statistic is requested, the “fourth” may be smaller than the “third”. If both the third and fourth order statistics from a given sample are desired, they should be obtained from a single call to RNUNO (by specifying IFIRST less than or equal to 3 and ILAST greater than or equal to 4).

RNARM

Generates a time series from a specified ARMA model.

Required Arguments

CNST — Overall constant. (Input)
See Comments.

PAR — Vector of length NPAR containing the autoregressive parameters. (Input)

LAGAR — Vector of length NPAR containing the order of the autoregressive parameters. (Input)
The elements of LAGAR must be greater than or equal to one.

PMA — Vector of length NPMA containing the moving average parameters. (Input)

LAGMA — Vector of length NPMA containing the order of the moving average parameters. (Input)
The elements of LAGMA must be greater than or equal to one.

IADIST — Option for normally distributed innovations. (Input)

IADIST	Action
0	Innovations are generated from a normal distribution (white noise) with mean 0 and variance AVAR.
1	Innovations are specifed by the user.

AVAR — Variance of the normal distribution, if used. (Input)
For IADIST = 0, AVAR is input; and for IADIST = 1, AVAR is unused.

A — Vector of length NW + max(LAGMA(*j*)) containing the innovations. (Input or output)
For IADIST = 1, A is input; and for IADIST = 0, A is output.

WI — Vector of length max(LAGAR(*i*)) containing the initial values of the time series. (Input)

W — Vector of length NW containing the generated time series. (Output)

Optional Arguments

NW — Number of observations of the time series to generate. (Input)
NW must be greater than or equal to one.
Default: NW = size (W,1).

NPAR — Number of autoregressive parameters. (Input)
NPAR must be greater than or equal to zero.
Default: NPAR = size (PAR,1).

NPMA — Number of moving average parameters. (Input)
NPMA must be greater than or equal to zero.
Default: NPMA = size (PMA,1).

FORTRAN 90 Interface

Generic: `CALL RNARM (CNST, PAR, LAGAR, PMA, LAGMA, IADIST, AVAR, A, WI, W [,…])`

Specific: The specific interface names are S_RNARM and D_RNARM.

FORTRAN 77 Interface

Single: `CALL RNARM (NW, CNST, NPAR, PAR, LAGAR, NPMA, PMA, LAGMA, IADIST, AVAR, A, WI, W)`

Double: The double precision name is DRNARM.

Example 1

In this example, RNARM is used to generate a time series of length five, using an ARMA model with three autoregressive parameters and two moving average parameters. The start values are 0.1000, 0.0500, and 0.0375.

```
      USE RNARM_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    NPAR, NPMA, NW
      PARAMETER  (NPAR=3, NPMA=2, NW=5)
!
      INTEGER    I, IADIST, ISEED, LAGAR(NPAR), LAGMA(NPMA), NOUT
      REAL       A(NW+2), AVAR, CNST, PAR(NPAR), PMA(NPMA), W(NW), &
                 WI(3)
!
      CALL UMACH (2, NOUT)
      LAGAR(1) = 1
      LAGAR(2) = 2
      LAGAR(3) = 3
      PAR(1)   = 0.500
      PAR(2)   = 0.250
      PAR(3)   = 0.125
```

```
      LAGMA(1) = 1
      LAGMA(2) = 2
      PMA(1)   = -0.500
      PMA(2)   = -0.250
      IADIST   = 0
      CNST    = 1.0
      AVAR     = 0.1
      WI(1)    = 0.1
      WI(2)    = 0.05
      WI(3)    = 0.0375
      ISEED    = 123457
      CALL RNSET (ISEED)
      CALL RNARM (CNST, PAR, LAGAR, PMA, LAGMA, &
                  IADIST, AVAR, A, WI, W)
      WRITE (NOUT,99999) (W(I),I=1,NW)
99999 FORMAT ('   Simulated ARMA(3,2) series ', 5F7.4)
      END
```

Output

```
Simulated ARMA(3,2) series  1.4033 2.2200 2.2864 2.8878 2.8322
```

Comments

1. The time series is generated according to the following model:

   $X(i) = CNST + PAR(1) * X(i - LAGAR(1)) + \ldots + PAR(NPAR) * X(i - LAGAR(NPAR)) + A(i) - PMA(1) * A(i - LAGMA(1)) - \ldots - PMA(NPMA) * A(i - LAGAR(NPMA))$

   where

   $X(t) = W(t), t = 1, 2, \ldots, NW$

   and

   $W(t) = WI(t + p), \quad t = 1 - p, 2 - p, \ldots, -1, 0$

   with $p = \max(LAGAR(k))$.

   The constant is related to the mean of the series, WMEAN, as follows:

   ```
   CNST = WMEAN * (1 - PAR(1)- ... - PAR(NPAR))
   ```

2. Time series whose innovations have a nonnormal distribution may be simulated by setting IADIST = 1 and by providing the appropriate innovations in A and start values in WI.

3. The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNARM simulates an ARMA(p, q) process, $\{W_t\}$ for $t = 1, 2, \ldots, n$ (with n = NW, p = NPAR, and q = NPMA). The model is

$$\phi(B)W_t = \theta_0 + \theta(B)A_t \qquad t \in ZZ$$

where B is the backward shift operator,

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p$$

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q$$

Let μ be the mean of the time series $\{W_t\}$. The overall constant θ_0 (CNST) is

$$\theta_0 = \begin{cases} \mu & p = 0 \\ \mu\left(1 - \sum_{i=1}^{p} \phi_i\right) & p > 0 \end{cases}$$

Additional Example

Example 2

In this example, 500 observations from an ARMA(2, 2) process are simulated using RNARM; and then routine NSPE (page 664) is used to estimate the parameters of the model. The model is used as an example by Priestley (1981), page 139.

```
      USE RNARM_INT
      USE RNSET_INT
      USE NSPE_INT
      INTEGER    NPAR, NPMA, NW
      PARAMETER  (NPAR=2, NPMA=2, NW=500)
!
      INTEGER    IADIST, ISEED, LAGAR(NPAR), LAGMA(NPMA)
      REAL       A(NW+2), AVAR, AVAR1, CNST, CNST1, PAR(NPAR), &
                 PAR1(NPAR), PMA(NPMA), PMA1(NPMA), W(NW), WI(2), WMEAN
!
      LAGAR(1) = 1
      LAGAR(2) = 2
      PAR(1)   = -1.4
      PAR(2)   = -0.5
      LAGMA(1) = 1
      LAGMA(2) = 2
      PMA(1)   = 0.2
      PMA(2)   = 0.1
      IADIST   = 0
      CNST    = 0.0
      AVAR     = 1.0
      WI(1)    = 0.0
      WI(2)    = 0.0
      ISEED    = 123457
      CALL RNSET (ISEED)
      CALL RNARM (CNST, PAR, LAGAR, PMA, LAGMA, &
```

```
                IADIST, AVAR, A, WI, W)
      CALL NSPE (W, CNST1, PAR1, PMA1, AVAR1, IPRINT=1)
      END
```

Output

```
Results from NSPE/N2PE
WMEAN =     .02192622
CONST =      .0695866
AVAR  =    1.0936457
       PAR
      1       2
 -1.533  -0.641
        PMA
       1        2
  0.0560   0.1294
```

RNNPP

Generates pseudorandom numbers from a nonhomogeneous Poisson process.

Required Arguments

TIMBEG — Lower endpoint of the time interval of the process. (Input)
TIMBEG must be nonnegative. Usually, TIMBEG = 0.

TIMEND — Upper endpoint of the time interval of the process. (Input)
TIMEND must be greater than TIMBEG.

FTHETA — User-supplied FUNCTION to provide the value of the rate of the process as a function of time. This function must be defined over the interval from TIMBEG to TIMEND and must be nonnegative in that interval. The form is FTHETA(TIME), w:here

TIME — Time at which the rate function is evaluated. (Input)

FTHETA — Value of the rate function. (Output)

FTHETA must be declared EXTERNAL in the calling program.

THEMIN — Minimum value of the rate function FTHETA in the interval (TIMBEG, TIMEND). (Input)
If the actual minimum is unknown, set THEMIN = 0.0.

THEMAX — Maximum value of the rate function FTHETA in the interval (TIMBEG, TIMEND). (Input)
If the actual maximum is unknown, set THEMAX to a known upper bound of the maximum. The efficiency of RNNPP is less the greater THEMAX exceeds the true maximum.

NEUB — Upper bound on the number of events to be generated. (Input)
In order to be reasonably sure that the full process through time TIMEND is generated, calculate NEUB as NEUB = X + 10.0 * SQRT(X), where X = THEMAX * (TIMEND − TIMBEG). The only penalty in setting NEUB too large is that the output vector must be dimensioned of length NEUB.

NE — Number of events actually generated. (Output)
If NE is less that NEUB, the time TIMEND is reached before NEUB events are realized.

R — Vector of length NE containing the times to events. (Output)
R must be dimensioned to be of length NEUB.

FORTRAN 90 Interface

Generic: `CALL RNNPP (TIMBEG, TIMEND, FTHETA, THEMIN, THEMAX, NEUB, NE, R)`

Specific: The specific interface names are S_RNNPP and D_RNNPP.

FORTRAN 77 Interface

Single: `CALL RNNPP (TIMBEG, TIMEND, FTHETA, THEMIN, THEMAX, NEUB, NE, R)`

Double: The double precision name is DRNNPP.

Example 1

In this example, RNNPP is used to generate the first five events in the time 0 to 20 (if that many events are realized) in a nonhomogeneous process with rate function

$$\lambda(t) = 0.6342\, e^{0.001427t}$$

for $0 < t \leq 20$.

Since this is a monotonically increasing function of t, the minimum is at $t = 0$ and is 0.6342, and the maximum is at $t = 20$ and is $0.6342\, e^{0.02854} = 0.652561$.

```
      USE RNNPP_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    NEUB
      PARAMETER  (NEUB=5)
!
      INTEGER    I, ISEED, NE, NOUT
      REAL       FTHETA, R(NEUB), THEMAX, THEMIN, TIMBEG, TIMEND
      EXTERNAL   FTHETA
!
      CALL UMACH (2, NOUT)
```

```
      TIMBEG = 0.0
      TIMEND = 20.0
      THEMIN = 0.6342
      THEMAX = 0.652561
      ISEED  = 123457
      CALL RNSET (ISEED)
      CALL RNNPP (TIMBEG, TIMEND, FTHETA, THEMIN, THEMAX, NEUB, NE, R)
      WRITE (NOUT,99999) NE, (R(I),I=1,NE)
99999 FORMAT (’   Inter-event times for the first ’, I1, ’ events’, /, &
            ’   in the process: ’, 5F7.4)
      END
!
      REAL FUNCTION FTHETA (T)
      REAL       T
!
      REAL       EXP
      INTRINSIC  EXP
!
      FTHETA = 0.6342*EXP(0.001427*T)
      RETURN
      END
```

Output

```
Inter-event times for the first 5 events
in the process:  0.0527 0.4080 0.2584 0.0198 0.1676
```

Comments

The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. The routine RNOPT (page 1311) can be used to select the form of the generator.

Description

Routine RNNPP simulates a one-dimensional nonhomogeneous Poisson process with rate function THETA in a fixed interval (TIMBEG, TIMEND].

Let $\lambda(t)$ be the rate function and t_0 = TIMBEG and t_1 = TIMEND. Routine RNNPP uses a method of thinning a nonhomogeneous Poisson process $\{N^*(t), t \geq t_0\}$ with rate function $\lambda^*(t) \geq \lambda(t)$ in $(t_0, t_1]$, where the number of events, N^*, in the interval $(t_0, t_1]$ has a Poisson distribution with parameter

$$\mu_0 = \int_{t_0}^{t_1} \lambda(t)\, dt$$

The function

$$\Lambda(t) = \int_0^t \lambda(t)\, dt$$

is called the *integrated rate function*.) In RNNPP, $\lambda^*(t)$ is taken to be a constant λ^*(= THEMAX) so that at time t_i, the time of the next event t_{i+1} is obtained by generating and cumulating exponential random numbers

$$E_{1,i}^*, E_{2,i}^*, \ldots,$$

with parameter λ^*, until for the first time

$$u_{j,i} \le \left(t_i + E_{1,i}^* + \cdots + E_{j,i}^*\right)/\lambda^*$$

where the $u_{j,i}$ are independent uniform random numbers between 0 and 1. This process is continued until the specified number of events, NEUB, is realized or until the time, TIMEND, is exceeded. This method is due to Lewis and Shedler (1979), who also review other methods. The most straightforward (and most efficient) method is by inverting the integrated rate function, but often this is not possible.

If THEMAX is actually greater than the maximum of $\lambda(t)$ in $(t_0, t_1]$, the routine will work, but less efficiently. Also, if $\lambda(t)$ varies greatly within the interval, the efficiency is reduced. In that case, it may be desirable to divide the time interval into subintervals within which the rate function is less variable. This is possible because the process is without memory.

If no time horizon arises naturally, TIMEND must be set large enough to allow for the required number of events to be realized. Care must be taken, however, that FTHETA is defined over the entire interval.

After simulating a given number of events, the next event came be generated by setting TIMBEG to the time of the last event (the sum of the elements in R) and calling RNNPP again. Cox and Lewis (1966) discuss modeling applications of nonhomogeneous Poisson processes.

Additional Example

Example 2

As it turns out in the simulation above, the first five events are realized before time equals 20. If it is desired to continue the simulation to time equals 20, setting NEUB to 49 (that is,

$$\lambda^*\left(t_1 - t_0\right) + 10\sqrt{\lambda^*\left(t_1 - t_0\right)}$$

would likely ensure that the time is reached. In the following example, we see that there are twelve events realized by time equals 20.

```
      USE UMACH_INT
      USE RNSET_INT
      USE RNNPP_INT
      USE SSUM_INT
      INTEGER    NEUB
      PARAMETER  (NEUB=49)
!
      INTEGER    ISEED, NE, NOUT
      REAL       FTHETA, R(NEUB), T, THEMAX, THEMIN, TIMBEG, TIMEND
      EXTERNAL   FTHETA
!
      CALL UMACH (2, NOUT)
      TIMBEG = 0.0
      TIMEND = 20.0
      THEMIN = 0.6342
      THEMAX = 0.652561
```

```
      ISEED  = 123457
      CALL RNSET (ISEED)
      CALL RNNPP (TIMBEG, TIMEND, FTHETA, THEMIN, THEMAX, NEUB, NE, R)
      T = TIMBEG + SSUM(NE,R,1)
      IF (NE .LT. NEUB) THEN
         WRITE (NOUT,99998) NE, T
99998    FORMAT (’   Only ’, I2, ’ events occurred before the time’, &
               /, ’   limit expired.  The last event occurred at’, /, &
               ’   time = ’, F6.3)
      ELSE
         WRITE (NOUT,99999) NE, T
99999    FORMAT (’   Possibly more than ’, I2, ’ events would have’, &
               /, ’   occurred before the time limit expired.’, /, &
               ’   The last event occurred at time = ’, F6.3)
      END IF
      END
!
      REAL FUNCTION FTHETA (T)
      REAL       T
!
      REAL       EXP
      INTRINSIC  EXP
!
      FTHETA = 0.6342*EXP(0.001427*T)
      RETURN
      END
```

Output

```
Only 12 events occurred before the time
limit expired.  The last event occurred at
time = 18.809
```

RNPER

Generates a pseudorandom permutation.

Required Arguments

IPER — Vector of length K containing the random permutation of the integers from 1 to K. (Output)

Optional Arguments

K — Number of integers to be permuted. (Input)
Default: K = size (IPER,1).

FORTRAN 90 Interface

Generic: `CALL RNPER (IPER [,…])`

Specific: The specific interface name is S_RNPER.

FORTRAN 77 Interface

Single: `CALL RNPER (K, IPER)`

Example

In this example, `RNPER` is called to produce a pseudorandom permutation of the integers from 1 to 10.

```
      USE RNPER_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    IPER(10), ISEED, NOUT
!
      CALL UMACH (2, NOUT)
!                                Initialize seed of random number
!                                generator.
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNPER (IPER)
      WRITE (NOUT,99999) IPER
99999 FORMAT ('   Random permutation of the integers from 1 to 10', /, &
            10I5)
      END
```

Output

```
Random permutation of the integers from 1 to 10
 5    9    2    8    1    6    4    7    3   10
```

Comments

The routine `RNSET` (page 1311) can be used to initialize the seed of the random number generator. The routine `RNOPT` (page 1311) can be used to select the form of the generator.

Description

Routine `RNPER` generates a pseudorandom permutation of the integers from 1 to `K`. It begins by filling a vector of length `K` with the consecutive integers 1 to `K`. Then, with M initially equal to `K`, a random index J between 1 and M (inclusive) is generated. The element of the vector with the index M and the element with index J swap places in the vector. M is then decremented by 1 and the process repeated until $M = 1$.

RNSRI

Generates a simple pseudorandom sample of indices.

Required Arguments

NPOP — Number of items in the population. (Input)

INDEX — Vector of length NSAMP containing the indices of the sample. (Output)
INDEX is a random sample (without replacement) of the integers from 1 to NPOP, in increasing order.

Optional Arguments

NSAMP — Sample size desired. (Input)
Default: NSAMP = size (INDEX,1).

FORTRAN 90 Interface

Generic: `CALL RNSRI (NPOP, INDEX [,…])`

Specific: The specific interface name is S_RNSRI.

FORTRAN 77 Interface

Single: `CALL RNSRI (NSAMP, NPOP, INDEX)`

Example

In this example, RNSRI is used to generate the indices of a pseudorandom sample of size 5 from a population of size 100.

```
      USE RNSRI_INT
      USE UMACH_INT
      USE RNSET_INT
      INTEGER    INDEX(5), ISEED, NOUT, NPOP
!
      CALL UMACH (2, NOUT)
      NPOP  = 100
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNSRI (NPOP, INDEX)
      WRITE (NOUT,99999) INDEX
99999 FORMAT (’      Random sample: ’, 5I4)
      END
```

Output

```
Random sample:    2   22   53   61  79
```

Comments

1. The routine RNSET (page 1311) can be used to initialize the seed of the random number generator. If NSAMP is greater than NPOP/2, RNSRI uses two different generators in an algorithm due to Ahrens and Dieter (1985). The routine RNOPT (page 1311) can be used to select the form of the generator used for uniform deviates in the algorithm. The generator used for exponential deviates in the algorithm is a nonshuffled generator that is different from the one for the uniform. If IOPTU is the option indicator for the

uniform generator (see documentation for RNOPT, page 1311), then the option indicator for the exponential generator is MOD((2 * INT((IOPTU + 1)/2) + 1), 6).

2. The routine RNSRS (page 1401) can be used to select a sample from a population of unknown size.

Description

Routine RNSRI generates the indices of a pseudorandom sample,without replacement, of size NSAMP numbers from a population of size NPOP. If NSAMP is greater than NPOP/2, the integers from 1 to NPOP are selected sequentially with a probability conditional on the number selected and the number remaining to be considered. If, when the i-th population index is considered, j items have been included in the sample, then the index i is included with probability $(\text{NSAMP} - j)/(\text{NPOP} + 1 - i)$.

If NSAMP is not greater than NPOP/2, a $O(\text{NSAMP})$ algorithm due to Ahrens and Dieter (1985) is used. Of the methods discussed by Ahrens and Dieter, the one called SG* is used in RNSRI. It involves a preliminary selection of q indices using a geometric distribution for the distances between each index and the next one. If the preliminary sample size q is less than NSAMP, a new preliminary sample is chosen, and this is continued until a preliminary sample greater in size than NSAMP is chosen. This preliminary sample is then thinned using the same kind of sampling as described above for the case in which the sample size is greater than half of the population size. Routine RNSRI does not store the preliminary sample indices, but rather restores the state of the generator used in selecting the sample initially, and then passes through once again, making the final selection as the preliminary sample indices are being generated.

RNSRS

Generates a simple pseudorandom sample from a finite population.

Required Arguments

POP — NROW by NVAR matrix containing the population to be sampled. (Input) If IDO = 0, POP contains the entire population; otherwise, POP contains a different part of the population on each invocation of RNSRS.

NPOP — The number of items in the population. (Output, if IDO = 0 or 1; input/output, if IDO = 2.)
If IDO = 0, NPOP = NROW on output. If the population is input a few items at a time, it is not necessary to know the number of items in the population in advance. NPOP is used to cumulate the population size and should not be changed between calls to RNSRS. If, on output, NPOP is greater than or equal to NSAMP, the sampling can be considered complete for a population of size NPOP.

SAMP — NSAMP by NVAR matrix containing the sample. (Output, if IDO = 0 or 1; input/output, if IDO = 2.)

INDEX — Vector of length NSAMP containing the indices of the sample in the population. (Output, if IDO = 0 or 1; input/output, if IDO = 2.) The INDEX(I)-th item in the population is the I-th item in the sample. INDEX is not necessarily in increasing order.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of RNSRS for this data set, and the entire population is input at once.
1	This is the first invocation, and additional calls to RNSRS will be made. Initialization and updating for the subpopulation in POP are performed.
2	This is an additional invocation of RNSRS, and updating for the subpopulation in POP is performed.

NROW — Number of rows of data currently input in POP. (Input)
NROW must be nonnegative.
Default: NROW = size (POP,1).

NVAR — Number of variables in the population and in the sample. (Input)
Default: NVAR = size (POP,2).

LDPOP — Leading dimension of POP exactly as specified in the dimension statement in the calling program. (Input)
Default: LDPOP = size (POP,1).

NSAMP — The sample size desired. (Input)
Default: NSAMP = size (SAMP,1).

LDSAMP — Leading dimension of SAMP exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSAMP = size (SAMP,1).

FORTRAN 90 Interface

Generic: `CALL RNSRS (POP, NPOP, SAMP, INDEX [,…])`

Specific: The specific interface names are S_RNSRS and D_RNSRS.

FORTRAN 77 Interface

Single: `CALL RNSRS (IDO, NROW, NVAR, POP, LDPOP, NSAMP, NPOP, SAMP, LDSAMP, INDEX)`

Double: The double precision name is `DRNSRS`.

Example 1

In this example, `RNSRS` is used to generate a sample of size 5 from a population stored in the matrix `POP`. All of the data are available at once, so default `IDO` = 0 is used.

```
      USE RNSRS_INT
      USE UMACH_INT
      USE GDATA_INT
      USE RNSET_INT
      INTEGER    I, INDEX(5), ISEED, J, NOUT, NPOP, NROW, NVAR

      REAL       POP(176,2), SAMP(5,2)
!
      CALL UMACH (2, NOUT)
!                                Get Wolfer sunspot data to use
!                                as "population".
      CALL GDATA (2, POP, NROW, NVAR)
!                                Initialize seed of random number
!                                generator.
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNSRS (POP, NPOP, SAMP, INDEX)
      WRITE (NOUT,99999) NPOP, INDEX, ((SAMP(I,J),I=1,5),J=1,2)
99999 FORMAT ('     The population size is ', I5, /, '   Indices of ', &
            'random sample: ', 5I8, /, '                 The sample: ' &
            , 5F8.0, /, '                              ', 5F8.0)
      END
```

Output

```
The population size is   176
Indices of random sample:       16      80     175      25      21
              The sample:    1764.   1828.   1923.   1773.   1769.
                               36.     62.      6.     35.    106.
```

Comments

1. The routine `RNSET` (page 1311) can be used to initialize the seed of the random number generator. The routine `RNOPT` (page 1311) can be used to select the form of the generator.

2. The routine `RNSRI` (page 1399) can be used to select a sample of indices in increasing order.

Description

Routine RNSRS generates a pseudorandom sample from a given population, without replacement, using an algorithm due to McLeod and Bellhouse (1983).

The first NSAMP items in the population are included in the sample. Then, for each successive item from the population, a random item in the sample is replaced by that item from the population with probability equal to the sample size divided by the number of population items that have been encountered at that time.

Additional Example

Example 2

Routine RNSRS is now used to generate a sample of size 5 from the same population as in the example above except the data are input to RNSRS one observation at a time. This is the way RNSRS may be used to sample from a file on disk or tape. Notice that the number of records need not be known in advance.

```
      USE RNSRS_INT
      USE UMACH_INT
      USE GDATA_INT
      USE RNSET_INT
      INTEGER    ISEED, NOUT, IDO, NROW, NVAR, NPOP, INDEX(5)
      REAL       POP(176,2), SAMP(5,2), X(2, 1)
      CALL UMACH(2, NOUT)
!                                Get Wolfer sunspot data to use
!                                as "population".
      CALL GDATA (2, POP, NROW, NVAR)
!                                Initialize seed of random number
!                                generator.
      ISEED = 123457
      CALL RNSET(ISEED)
      IDO = 1
      DO 10 I=1,176
!                                In this DO-loop, the data would
!                                generally be read from a file,
!                                one observation at a time.  This
!                                program simulates this by copying
!                                the observations one at a time into
!                                X from POP.
         X(1,1) = POP(I,1)
         X(2,1) = POP(I,2)
         CALL RNSRS (X, NPOP, SAMP, INDEX, IDO=IDO, &
                     NROW=1, NVAR=NVAR, LDPOP=1)
         IDO = 2
   10 CONTINUE
      WRITE(NOUT, 20) NPOP, INDEX, ((SAMP(I,J),I=1,5),J=1,2)
   20 FORMAT ('     The population size is ', I5,/, &
             '   Indices of random sample: ', 5I8,/, &
             '                The sample: ', 5F8.0,/, &
             '                           ', 5F8.0)
      END
```

Output

```
The population size is   176
Indices of random sample:       16      80     175      25      21
             The sample:    1764.   1828.   1923.   1773.   1769.
                              36.     62.      6.     35.    106.
```

FAURE_INIT

Shuffled Faure sequence initialization.

Required Arguments

NDIM — The dimension of the hyper-rectangle. (Input)

STATE — An `IMSL_FAURE` pointer for the derived type created by the call to `FAURE_INIT`. The output contains information about the sequence. Use `?_IMSL_FAURE` as the type, where `?_` is `S_` or `D_` depending on precision. (Output)

Optional Arguments

NBASE — The base of the Faure sequence. (Input)

Default: The smallest prime number greater than or equal to `NDIM`.

NSKIP — The number of points to be skipped at the beginning of the Faure sequence. (Input)

Default: $\lfloor \texttt{base}^{m/2-1} \rfloor$, where $m = \lfloor \log B / \log \texttt{base} \rfloor$ and B is the largest machine representable integer.

FORTRAN 90 Interface

Generic: `CALL FAURE_INIT (NDIM, STATE [,…])`

Specific: The specific interface names are `S_FAURE_INIT` and `D_FAURE_INIT`.

FAURE_FREE

Frees the structure containing information about the Faure sequence.

Required Arguments

STATE — An `IMSL_FAURE` pointer containing the structure created by the call to `FAURE_INIT`. (Input/Output)

FORTRAN 90 Interface

Generic: `CALL FAURE_FREE (STATE)`

Specific: The specific interface names are `S_FAURE_FREE` and `D_FAURE_FREE`.

FAURE_NEXT

Computes a shuffled Faure sequence.

Required Arguments

STATE — An `IMSL_FAURE` pointer containing the structure created by the call to `FAURE_INIT`. The structure contains information about the sequence. The structure should be freed using `FAURE_FREE` after it is no longer needed. (Input/Output)

NEXT_PT — Vector of length `NDIM` containing the next point in the shuffled Faure sequence, where `NDIM` is the dimension of the hyper-rectangle specified in `FAURE_INIT`. (Output)

Optional Arguments

IMSL_RETURN_SKIP — Returns the current point in the sequence. The sequence can be restarted by calling `FAURE_INIT` using this value for `NSKIP`, and using the same value for `NDIM`. (Input)

FORTRAN 90 Interface

Generic: `CALL FAURE_NEXT (STATE, NEXT_PT [,…])`

Specific: The specific interface names are `S_FAURE_NEXT` and `D_FAURE_NEXT`.

Example

In this example, five points in the Faure sequence are computed. The points are in the three-dimensional unit cube.

Note that `FAURE_INIT` is used to create a structure that holds the state of the sequence. Each call to `FAURE_NEXT` returns the next point in the sequence and updates the `IMSL_FAURE` structure. The final call to `FAURE_FREE` frees data items, stored in the structure, that were allocated by `FAURE_INIT`.

```
use faure_int
implicit none
type (s_imsl_faure), pointer  :: state
real(kind(1e0))          :: x(3)
integer,parameter :: ndim=3
integer           :: k
```

```
!                                  CREATE THE STRUCTURE THAT HOLDS
!                                  THE STATE OF THE SEQUENCE.
      call faure_init(ndim, state)
!                                  GET THE NEXT POINT IN THE SEQUENCE
      do k=1,5
         call faure_next(state, x)
         write(*,'(3F15.3)') x(1), x(2) , x(3)
      enddo
!                                   FREE DATA ITEMS STORED IN
!                                   state STRUCTURE
      call faure_free(state)
      end
```

Output

```
  0.334       0.493       0.064
  0.667       0.826       0.397
  0.778       0.270       0.175
  0.111       0.604       0.509
  0.445       0.937       0.842
```

Description

Discrepancy measures the deviation from uniformity of a point set.

The discrepancy of the point set $x_1,...,x_n \in [0,1]^d$, $d \geq 1$, is defined

$$D_n^{(d)} = \sup_E \left| \frac{A(E;n)}{n} - \lambda(E) \right|,$$

where the supremum is over all subsets of $[0, 1]^d$ of the form

$$E = \left[0,t_1\right) \times ... \times \left[0,t_d\right),\ 0 \leq t_j \leq 1,\ 1 \leq j \leq d\ ,$$

λ is the Lebesque measure, and $A(E;n)$ is the number of the x_j contained in E.

The sequence $x_1, x_2, \ldots$ of points $[0,1]^d$ is a low-discrepancy sequence if there exists a constant $c(d)$, depending only on d, such that

$$D_n^{(d)} \leq c(d) \frac{(\log n)^d}{n}$$

for all $n>1$.

Generalized Faure sequences can be defined for any prime base $b \geq d$. The lowest bound for the discrepancy is obtained for the smallest prime $b \geq d$, so the optional argument `NBASE` defaults to the smallest prime greater than or equal to the dimension.

The generalized Faure sequence $x_1, x_2, \ldots$, is computed as follows:

Write the positive integer n in its b-ary expansion,

$$n = \sum_{i=0}^{\infty} a_i(n) b^i$$

where $a_i(n)$ are integers, $0 \le a_i(n) < b$.

The j-th coordinate of x_n is

$$x_n^{(j)} = \sum_{k=0}^{\infty} \sum_{d=0}^{\infty} c_{kd}^{(j)} \; a_d(n) \; b^{-k-1}, \qquad 1 \le j \le d$$

The generator matrix for the series, $c_{k\,d}^{(j)}$, is defined to be

$$c_{kd}^{(j)} = j^{\,d-k} c_{k\,d}$$

and $c_{k\,d}$ is an element of the Pascal matrix,

$$c_{k\,d} = \begin{cases} \dfrac{d!}{c!(d-c)!} & k \le d \\ 0 & k > d \end{cases}$$

It is faster to compute a shuffled Faure sequence than to compute the Faure sequence itself. It can be shown that this shuffling preserves the low-discrepancy property.

The shuffling used is the b-ary Gray code. The function $G(n)$ maps the positive integer n into the integer given by its b-ary expansion.

The sequence computed by this function is $x(G(n))$, where x is the generalized Faure sequence.

Chapter 19: Utilities

Routines

WRRRN

Prints a real rectangular matrix with integer row and column labels.

Required Arguments

TITLE — Character string specifying the title. (Input)
TITLE set equal to a blank character(s) suppresses printing of the title. Use "% /" within the title to create a new line. Long titles are automatically wrapped.

A — NRA by NCA matrix to be printed. (Input)

Optional Arguments

NRA — Number of rows. (Input)
Default: NRA = size (A, 1).

NCA — Number of columns. (Input)
Default: NCA = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

ITRING — Triangle option. (Input)
Default: ITRING = 0.

ITRING	Action
0	Full matrix is printed.
1	Upper triangle of A is printed, including the diagonal.

ITRING	Action
2	Upper triangle of A excluding the diagonal of A is printed.
–1	Lower triangle of A is printed, including the diagonal.
–2	Lower triangle of A excluding the diagonal of A is printed.

FORTRAN 90 Interface

Generic: `CALL WRRRN (TITLE, A [,…])`

Specific: The specific interface names are `S_WRRRN` and `D_WRRRN` for two dimensional arrays, and `S_WRRRN1D` and `D_WRRRN1D` for one dimensional arrays.

FORTRAN 77 Interface

Single: `CALL WRRRN (TITLE, NRA, NCA, A, LDA, ITRING)`

Double: The double precision name is `DWRRRN`.

Example

The following example prints all of a 3×4 matrix A where $a_{ij} = i + j/10$.

```
      USE WRRRN_INT
      INTEGER    ITRING, LDA, NCA, NRA
      PARAMETER  (ITRING=0, LDA=10, NCA=4, NRA=3)
!
      INTEGER    I, J
      REAL       A(LDA,NCA)
!
      DO 20  I=1, NRA
         DO 10  J=1, NCA
            A(I,J) = I + J*0.1
   10    CONTINUE
   20 CONTINUE
!                                  Write A matrix.
      CALL WRRRN ('A', A, NRA=NRA)
      END
```

Output

```
                  A
         1       2       3       4
1    1.100   1.200   1.300   1.400
2    2.100   2.200   2.300   2.400
3    3.100   3.200   3.300   3.400
```

Comments

1. A single D, E, or F format is chosen automatically in order to print 4 significant digits for the largest element of A in absolute value. Routine WROPT can be used to change the default format.

2. Horizontal centering, a method for printing large matrices, paging, printing a title on each page, and many other options can be selected by invoking WROPT.

3. A page width of 78 characters is used. Page width and page length can be reset by invoking PGOPT.

4. Output is written to the unit specified by UMACH (see the Reference Material).

Description

Routine WRRRN prints a real rectangular matrix with the rows and columns labeled 1, 2, 3, and so on. WRRRN can restrict printing to the elements of the upper or lower triangles of matrices via the ITRING option. Generally, ITRING $\neq 0$ is used with symmetric matrices.

In addition, one-dimensional arrays can be printed as column or row vectors. For a column vector, set NRA to the length of the array and set NCA = 1. For a row vector, set NRA = 1 and set NCA to the length of the array. In both cases, set LDA = NRA and set ITRING = 0.

WRRRL

Print a real rectangular matrix with a given format and labels.

Required Arguments

TITLE — Character string specifying the title. (Input)
TITLE set equal to a blank character(s) suppresses printing of the title.

A — NRA by NCA matrix to be printed. (Input)

RLABEL — CHARACTER * (*) vector of labels for rows of A. (Input)
If rows are to be numbered consecutively 1, 2, …, NRA, use RLABEL(1) = 'NUMBER'. If no row labels are desired, use RLABEL(1) = 'NONE'. Otherwise, RLABEL is a vector of length NRA containing the labels.

CLABEL — CHARACTER * (*) vector of labels for columns of A. (Input)
If columns are to be numbered consecutively 1, 2, …, NCA, use CLABEL(1) = 'NUMBER'. If no column labels are desired, use CLABEL(1) = 'NONE'. Otherwise, CLABEL(1) is the heading for the row labels, and either CLABEL(2) must be 'NUMBER' or 'NONE', or CLABEL must be a vector of length NCA + 1 with CLABEL(1 + *j*) containing the column heading for the *j*-th column.

Optional Arguments

NRA — Number of rows. (Input)
Default: NRA = size (A,1).

NCA — Number of columns. (Input)
Default: NCA = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

ITRING — Triangle option. (Input)
Default: ITRING = 0.

ITRING	Action
0	Full matrix is printed.
1	Upper triangle of A is printed, including the diagonal.
2	Upper triangle of A excluding the diagonal of A is printed.
–1	Lower triangle of A is printed, including the diagonal.
–2	Lower triangle of A excluding the diagonal of A is printed.

FMT — Character string containing formats. (Input)
If FMT is set to a blank character(s), the format used is specified by WROPT. Otherwise, FMT must contain exactly one set of parentheses and one or more edit descriptors. For example, FMT = '(F10.3)' specifies this F format for the entire matrix. FMT = '(2E10.3, 3F10.3)' specifies an E format for columns 1 and 2 and an F format for columns 3, 4 and 5. If the end of FMT is encountered and if some columns of the matrix remain, format control continues with the first format in FMT. Even though the matrix A is real, an I format can be used to print the integer part of matrix elements of A. The most useful formats are special formats, called the "V and W formats," that can be used to specify pretty formats automatically. Set FMT = '(V10.4)' if you want a single D, E, or F format selected automatically with field width 10 and with 4 significant digits. Set FMT = '(W10.4)' if you want a single D, E, F, or I format selected automatically with field width 10 and with 4 significant digits. While the V format prints trailing zeroes and a trailing decimal point, the W format does not. See Comment 4 for general descriptions of the V and W formats. FMT may contain only D, E, F, G, I, V, or W edit descriptors, e.g., the X descriptor is not allowed.
Default: FMT = ' '.

FORTRAN 90 Interface

Generic: CALL WRRRL (TITLE, A, RLABEL, CLABEL [,…])

Specific: The specific interface names are `S_WRRRL` and `D_WRRRL` for two dimensional arrays, and `S_WRRRL1D` and `D_WRRRL1D` for one dimensional arrays.

FORTRAN 77 Interface

Single: `CALL WRRRL (TITLE, NRA, NCA, A, LDA, ITRING, FMT, RLABEL, CLABEL)`

Double: The double precision name is `DWRRRL`.

Example

The following example prints all of a 3×4 matrix A where $a_{ij} = (i + j/10)10^{j-3}$.

```
      USE WRRRL_INT
      INTEGER    ITRING, LDA, NCA, NRA
      PARAMETER  (ITRING=0, LDA=10, NCA=4, NRA=3)
!
      INTEGER    I, J
      REAL       A(LDA,NCA)
      CHARACTER  CLABEL(5)*5, FMT*8, RLABEL(3)*5
!
      DATA FMT/'(W10.6)'/
      DATA CLABEL/'   ', 'Col 1', 'Col 2', 'Col 3', 'Col 4'/
      DATA RLABEL/'Row 1', 'Row 2', 'Row 3'/
!
      DO 20  I=1, NRA
         DO 10  J=1, NCA
            A(I,J) = (I+J*0.1)*10.0**(J-3)
   10    CONTINUE
   20 CONTINUE
!                                Write A matrix.
      CALL WRRRL ('A', A, RLABEL, CLABEL, NRA=NRA, FMT=FMT)
      END
```

Output

```
                              A
            Col 1       Col 2       Col 3       Col 4
Row 1       0.011       0.120       1.300      14.000
Row 2       0.021       0.220       2.300      24.000
Row 3       0.031       0.320       3.300      34.000
```

Comments

1. Workspace may be explicitly provided, if desired, by use of W2RRL/DW2RRL. The reference is:

   ```
   CALL W2RRL (TITLE, NRA, NCA, A, LDA, ITRING, FMT,
   RLABEL, CLABEL, CHWK)
   ```

   The additional argument is:

   CHWK — CHARACTER * 10 work vector of length NCA. This workspace is referenced only if all three conditions indicated at the beginning of this comment are met. Otherwise, CHWK is not referenced and can be a CHARACTER * 10 vector of length one.

2. The output appears in the following form:

	TITLE		
CLABEL(1)	CLABEL(2)	CLABEL(3)	CLABEL(4)
RLABEL(1)	Xxxxx	xxxxx	Xxxxx
RLABEL(2)	Xxxxx	xxxxx	Xxxxx

3. Use "% /" within titles or labels to create a new line. Long titles or labels are automatically wrapped.

4. For printing numbers whose magnitudes are unknown, the G format in FORTRAN is useful; however, the decimal points will generally not be aligned when printing a column of numbers. The V and W formats are special formats used by this routine to select a D, E, F, or I format so that the decimal points will be aligned. The V and W formats are specified as *Vn.d* and *Wn.d*. Here, *n* is the field width and *d* is the number of significant digits generally printed. Valid values for *n* are 3, 4,…, 40. Valid values for *d* are 1, 2, …, $n - 2$. If FMT specifies one format and that format is a V or W format, all elements of the matrix A are examined to determine one FORTRAN format for printing. If FMT specifies more than one format, FORTRAN formats are generated separately from each V or W format.

5. A page width of 78 characters is used. Page width and page length can be reset by invoking PGOPT.

6. Horizontal centering, method for printing large matrices, paging, method for printing NaN (not a number), printing a title on each page, and many other options can be selected by invoking WROPT.

7. Output is written to the unit specified by UMACH (see the Reference Material).

Description

Routine WRRRL prints a real rectangular matrix (stored in *A*) with row and column labels (specified by RLABEL and CLABEL, respectively) according to a given format (stored in FMT). WRRRL can restrict printing to the elements of upper or lower triangles of matrices via the ITRING option. Generally, ITRING $\neq 0$ is used with symmetric matrices.

In addition, one-dimensional arrays can be printed as column or row vectors. For a column vector, set NRA to the length of the array and set NCA = 1. For a row vector, set NRA = 1 and set NCA to the length of the array. In both cases, set LDA = NRA, and set ITRING = 0.

WRIRN

Prints an integer rectangular matrix with integer row and column labels.

Required Arguments

TITLE — Character string specifying the title. (Input)
TITLE set equal to a blank character(s) suppresses printing of the title. Use "% /" within the title to create a new line. Long titles are automatically wrapped.

MAT — NRMAT by NCMAT matrix to be printed. (Input)

Optional Arguments

NRMAT — Number of rows. (Input)
Default: NRMAT = size (MAT,1).

NCMAT — Number of columns. (Input)
Default: NCMAT = size (MAT,2).

LDMAT — Leading dimension of MAT exactly as specified in the dimension statement in the calling program. (Input)
Default: LDMAT = size (MAT,1).

ITRING — Triangle option. (Input)
Default: ITRING = 0.

ITRING	Action
0	Full matrix is printed.
1	Upper triangle of MAT is printed, including the diagonal.
2	Upper triangle of MAT excluding the diagonal of MAT is printed.
–1	Lower triangle of MAT is printed, including the diagonal.

ITRING	Action
–2	Lower triangle of MAT excluding the diagonal of MAT is printed.

FORTRAN 90 Interface

Generic: `CALL WRIRN (TITLE, MAT [,…])`

Specific: The specific interface name is `S_WRIRN`.

FORTRAN 77 Interface

Single: `CALL WRIRN (TITLE, NRMAT, NCMAT, MAT, LDMAT, ITRING)`

Example

The following example prints all of a 3×4 matrix A = MAT where $a_{ij} = 10i + j$.

```
      USE WRIRN_INT
      INTEGER    ITRING, LDMAT, NCMAT, NRMAT
      PARAMETER  (ITRING=0, LDMAT=10, NCMAT=4, NRMAT=3)
!
      INTEGER    I, J, MAT(LDMAT,NCMAT)
!
      DO 20  I=1, NRMAT
         DO 10  J=1, NCMAT
            MAT(I,J) = I*10 + J
   10    CONTINUE
   20 CONTINUE
!                                  Write MAT matrix.
      CALL WRIRN ('MAT', MAT, NRMAT=NRMAT)
      END
```

Output

```
          MAT
       1    2    3    4
1     11   12   13   14
2     21   22   23   24
3     31   32   33   34
```

Comments

1. All the entries in MAT are printed using a single `I` format. The field width is determined by the largest absolute entry.

2. Horizontal centering, a method for printing large matrices, paging, printing a title on each page, and many other options can be selected by invoking `WROPT`.

3. A page width of 78 characters is used. Page width and page length can be reset by invoking `PGOPT`.

4. Output is written to the unit specified by UMACH (see the Reference Material).

Description

Routine WRIRN prints an integer rectangular matrix with the rows and columns labeled 1, 2, 3, and so on. WRIRN can restrict printing to elements of the upper and lower triangles of matrices via the ITRING option. Generally, ITRING $\neq 0$ is used with symmetric matrices.

In addition, one-dimensional arrays can be printed as column or row vectors. For a column vector, set NRMAT to the length of the array and set NCMAT = 1. For a row vector, set NRMAT = 1 and set NCMAT to the length of the array. In both cases, set LDMAT = NRMAT and set ITRING = 0:

WRIRL

Print an integer rectangular matrix with a given format and labels.

Required Arguments

TITLE — Character string specifying the title. (Input)
TITLE set equal to a blank character(s) suppresses printing of the title.

MAT — NRMAT by NCMAT matrix to be printed. (Input)

RLABEL — CHARACTER * (*) vector of labels for rows of MAT. (Input)
If rows are to be numbered consecutively 1, 2, …, NRMAT, use RLABEL(1) = 'NUMBER'. If no row labels are desired, use RLABEL(1) = 'NONE'. Otherwise, RLABEL is a vector of length NRMAT containing the labels.

CLABEL — CHARACTER * (*) vector of labels for columns of MAT. (Input)
If columns are to be numbered consecutively 1, 2, …, NCMAT, use CLABEL(1) = 'NUMBER'. If no column labels are desired, use CLABEL(1) = 'NONE'. Otherwise, CLABEL(1) is the heading for the row labels, and either CLABEL(2) must be 'NUMBER' or 'NONE', or CLABEL must be a vector of length

NCMAT + 1 with CLABEL(1 + *j*) containing the column heading for the *j*-th column.

Optional Arguments

NRMAT — Number of rows. (Input)
Default: NRMAT = size (MAT,1).

NCMAT — Number of columns. (Input)
Default: NCMAT = size (MAT,2).

LDMAT — Leading dimension of MAT exactly as specified in the dimension statement in the calling program. (Input)
Default: LDMAT = size (MAT,1).

ITRING — Triangle option. (Input)
Default: ITRING = 0.

ITRING	Action
0	Full matrix is printed.
1	Upper triangle of MAT is printed, including the diagonal.
2	Upper triangle of MAT excluding the diagonal of MAT is printed.
–1	Lower triangle of MAT is printed, including the diagonal.
–2	Lower triangle of MAT excluding the diagonal of MAT is printed.

FMT — Character string containing formats. (Input)
If FMT is set to a blank character(s), the format used is a single I format with field width determined by the largest absolute entry. Otherwise, FMT must contain exactly one set of parentheses and one or more I edit descriptors. For example, FMT = '(I10)' specifies this I format for the entire matrix. FMT = '(2I10, 3I5)' specifies an I10 format for columns 1 and 2 and an I5 format for columns 3, 4 and 5. If the end of FMT is encountered and if some columns of the matrix remain, format control continues with the first format in FMT. FMT may only contain the I edit descriptor, e.g., the X edit descriptor is not allowed.
Default: FMT = ' '.

FORTRAN 90 Interface

Generic: CALL WRIRL (TITLE, MAT, RLABEL, CLABEL [,…])

Specific: The specific interface name is S_WRIRL.

FORTRAN 77 Interface

Single:
```
CALL WRIRL (TITLE, NRMAT, NCMAT, MAT, LDMAT, ITRING, FMT,
RLABEL, CLABEL)
```

Example

The following example prints all of a 3 × 4 matrix A = MAT where $a_{ij} = 10i + j$.

```
      USE WRIRL_INT
      INTEGER    ITRING, LDMAT, NCMAT, NRMAT

      PARAMETER  (ITRING=0, LDMAT=10, NCMAT=4, NRMAT=3)
!
      INTEGER    I, J, MAT(LDMAT,NCMAT)
      CHARACTER  CLABEL(5)*5, FMT*8, RLABEL(3)*5
```

```
!
      DATA FMT/'(I2)'/
      DATA CLABEL/'     ', 'Col 1', 'Col 2', 'Col 3', 'Col 4'/
      DATA RLABEL/'Row 1', 'Row 2', 'Row 3'/
!
      DO 20  I=1, NRMAT
         DO 10  J=1, NCMAT
            MAT(I,J) = I*10 + J
   10    CONTINUE
   20 CONTINUE
!                                  Write MAT matrix.
      CALL WRIRL ('MAT', MAT, RLABEL, CLABEL, NRMAT=NRMAT)
      END
```

Output

```
                  MAT
        Col 1  Col 2  Col 3  Col 4
Row 1     11     12     13     14
Row 2     21     22     23     24
Row 3     31     32     33     34
```

Comments

1. The output appears in the following form:

```
                                TITLE
          CLABEL(1)     CLABEL(2)  CALBEL(3)  CLABEL 4)
          RLABEL(1)     Xxxxx      xxxxx      xxxxx
          RLABEL(2)     Xxxxx      xxxxx      xxxxx
```

2. Use "% /" within titles or labels to create a new line. Long titles or labels are automatically wrapped.

3. A page width of 78 characters is used. Page width and page length can be reset by invoking PGOPT.

4. Horizontal centering, a method for printing large matrices, paging, printing a title on each page, and many other options can be selected by invoking WROPT.

5. Output is written to the unit specified by UMACH (see the Reference Material).

Description

Routine WRIRL prints an integer rectangular matrix (stored in MAT) with row and column labels (specified by RLABEL and CLABEL, respectively), according to a given format (stored in FMT). WRIRL can restrict printing to the elements of upper or lower triangles of matrices via the ITRING option. Generally, ITRING $\neq 0$ is used with symmetric matrices. In addition, one-dimensional arrays can be printed as column or row vectors. For a column vector, set NRMAT to the length of the array and set NCMAT = 1. For a row vector, set NRMAT = 1 and set NCMAT to the length of the array. In both cases, set LDMAT = NRMAT, and set ITRING = 0.

WROPT

Sets or retrieves an option for printing a matrix.

Required Arguments

IOPT — Indicator of option type. (Input)

IOPT	Description of Option Type
−1, 1	Horizontal centering or left justification of matrix to be printed
−2, 2	Method for printing large matrices
−3, 3	Paging
−4, 4	Method for printing NaN (not a number), and negative and positive machine infinity.
−5, 5	Title option
−6, 6	Default format for real and complex numbers
−7, 7	Spacing between columns
−8, 8	Maximum horizontal space reserved for row labels
−9, 9	Indentation of continuation lines for row labels
−10, 10	Hot zone option for determining line breaks for row labels
−11, 11	Maximum horizontal space reserved for column labels
−12, 12	Hot zone option for determining line breaks for column labels
−13, 13	Hot zone option for determining line breaks for titles
−14, 14	Option for the label that appears in the upper left hand corner that can be used as a heading for the row numbers or a label for the column headings for `WR**N` routines
−15, 15	Option for skipping a line between invocations of `WR**N` routines, provided a new page is not to be issued
−16, 16	Option for vertical alignment of the matrix values relative to the associated row labels that occupy more than one line

IOPT	Description of Option Type
0	Reset all the current settings saved in internal variables back to their last setting made with an invocation of WROPT with ISCOPE = 1. (This option is used internally by routines printing a matrix and is not useful otherwise.)

If IOPT is negative, ISET and ISCOPE are input and are saved in internal variables. If IOPT is positive, ISET is output and receives the currently active setting for the option (if ISCOPE = 0) or the last global setting for the option (if ISCOPE = 1). If IOPT = 0, ISET and ISCOPE are not referenced.

ISET — Setting for option selected by IOPT. (Input, if IOPT is negative; output, if IOPT is positive; not referenced if IOPT = 0)

IOPT	ISET	Meaning
–1, 1	0	Matrix is left justified
	1	Matrix is centered horizontally on page
–2, 2	0	A complete row is printed before the next row is printed. Wrapping is used if necessary.
	m	Here, *m* is a positive integer. Let n_1 be the maximum number of columns beginning with column 1 that fit across the page (as determined by the widths of the printing formats). First, columns 1 through n_1 are printed for rows 1 through *m*. Let n_2 be the maximum number of columns beginning with column $n_1 + 1$ that fit across the page. Second, columns $n_1 + 1$ through $n_1 + n_2$ are printed for rows 1 through *m*. This continues until the last columns are printed for rows 1 through *m*. Printing continues in this fashion for the next *m* rows, etc.
–3, 3	–2	Printing begins on the next line, and no paging occurs.
	–1	Paging is on. Every invocation of a WR*** routine begins on a new page, and paging occurs within each invocation as is needed
	0	Paging is on. The first invocation of a WR*** routine begins on a new page, and subsequent paging occurs as is needed. With this option, every invocation of a WR*** routine ends with a call to WROPT to reset this option to *k*, a positive integer giving the number of lines printed on the current page.

IOPT	ISET	Meaning
	k	Here, k is a positive integer. Paging is on, and *k* lines have been printed on the current page. If *k* is less than the page length IPAGE (see PGOPT, page 1428), then IPAGE – *k* lines are printed before a new page instruction is issued. If k is greater than or equal to IPAGE, then the first invocation of a WR*** routine begins on a new page. In any case, subsequent paging occurs as is needed. With this option, every invocation of a WR*** routine ends with a call to WROPT to reset the value of *k*.
–4, 4	0	NaN is printed as a series of decimal points, negative machine infinity is printed as a series of minus signs, and positive machine infinity is printed as a series of plus signs.
	1	NaN is printed as a series of blank characters, negative machine infinity is printed as a series of minus signs, and positive machine infinity is printed as a series of plus signs.
	2	NaN is printed as "NaN," negative machine infinity is printed as "-Inf" and positive machine infinity is printed as "Inf."
	3	NaN is printed as a series of blank characters, negative machine infinity is printed as "-Inf," and positive machine infinity is printed as "Inf."
–5, 5	0	Title appears only on first page.
	1	Title appears on the first page and all continuation pages.
–6, 6	0	Format is (W10.4). See Comment 2.
	1	Format is (W12.6). See Comment 2.
	2	Format is (1PE12.5).
	3	Format is *V*n.4 where the field width *n* is determined. See Comment 2.
	4	Format is *Vn*.6 where the field width *n* is determined. Comment 2.
	5	Format is 1PE*n.d* where $n = d + 7$, and $d + 1$ is the maximum number of significant digits.
–7, 7	k_1	Number of characters left blank between columns. k_1 must be between 0 and 5, inclusively.
–8, 8	k_2	Maximum width (in characters) reserved for row labels. $k_2 = 0$ means use the default.

IOPT	ISET	Meaning
–9, 9	k_3	Number of characters used to indent continuation lines for row labels. k_3 must be between 0 and 10, inclusively.
–10, 10	k_4	Width (in characters) of the hot zone where line breaks in row labels can occur. $k_4 = 0$ means use the default. k_4 must not exceed 50.
–11, 11	k_5	Maximum width (in characters) reserved for column labels. $k_5 = 0$ means use the default.
–12, 12	k_6	Width (in characters) of the hot zone where line breaks in column labels can occur. $k_6 = 0$ means use the default. k_6 must not exceed 50.
–13, 13	k_7	Width (in characters) of the hot zone where line breaks in titles can occur. k_7 must be between 1 and 50, inclusively.
–14	0	There is no label in the upper left hand corner.
	1	The label in the upper left hand corner is "Component" if a row vector or column vector is printed; the label is "Row/Column" if both the number of rows and columns are greater than one; otherwise, there is no label.
–15	0	A blank line is printed on each invocation of a WR**N routine before the matrix title provided a new page is not to be issued.
	1	A blank line is not printed on each invocation of a WR**N routine before the matrix title.
–16, 16	0	The matrix values are aligned vertically with the last line of the associated row label for the case IOPT = 2 and ISET is positive.
	1	The matrix values are aligned vertically with the first line of the associated row label.

ISCOPE — Indicator of the scope of the option. (Input if IOPT is nonzero; not referenced if IOPT = 0)

ISCOPE	Action
0	Setting is temporarily active for the next invocation of a WR*** matrix printing routine.
1	Setting is active until it is changed by another invocation of WROPT.

FORTRAN 90 Interface

Generic: `CALL WROPT (IOPT, ISET, ISCOPE)`

Specific: The specific interface name is `S_WROPT`.

FORTRAN 77 Interface

Single: `CALL WROPT (IOPT, ISET, ISCOPE)`

Example

The following example illustrates the effect of `WROPT` when printing a 3×4 real matrix A with `WRRRN` where $a_{ij} = i + j/10$. The first call to `WROPT` sets horizontal printing so that the matrix is first printed horizontally centered on the page. In the next invocation of `WRRRN`, the left-justification option has been set via routine `WROPT` so the matrix is left justified when printed. Finally, because the scope of left justification was only for the next call to a printing routine, the last call to `WRRRN` results in horizontally centered printing.

```
      USE WROPT_INT
      USE WRRRN_INT
      INTEGER    ITRING, LDA, NCA, NRA
      PARAMETER  (ITRING=0, LDA=10, NCA=4, NRA=3)
!
      INTEGER    I, IOPT, ISCOPE, ISETNG, J
      REAL       A(LDA,NCA)
!
      DO 20  I=1, NRA
         DO 10  J=1, NCA
            A(I,J) = I + J*0.1
   10    CONTINUE
   20 CONTINUE
!                                  Activate centering option.
!                                  Scope is global.
      IOPT   = -1
      ISETNG = 1
      ISCOPE = 1
!
      CALL WROPT (IOPT, ISETNG, ISCOPE)
!                                  Write A matrix.
      CALL WRRRN ('A', A, NRA=NRA)
!                                  Activate left justification.
!                                  Scope is local.
      IOPT   = -1
      ISETNG   = 0
      ISCOPE = 0
      CALL WROPT (IOPT, ISETNG, ISCOPE)
      CALL WRRRN ('A', A, NRA=NRA)
      CALL WRRRN ('A', A, NRA=NRA)
      END
```

Output

```
                                A
                        1       2       3       4
                1   1.100   1.200   1.300   1.400
                2   2.100   2.200   2.300   2.400
                3   3.100   3.200   3.300   3.400

                A
        1       2       3       4
1   1.100   1.200   1.300   1.400
2   2.100   2.200   2.300   2.400
3   3.100   3.200   3.300   3.400

                                A
                        1       2       3       4
                1   1.100   1.200   1.300   1.400
                2   2.100   2.200   2.300   2.400
                3   3.100   3.200   3.300   3.400
```

Comments

1. This program can be invoked repeatedly before using a `WR***` routine to print a matrix. The matrix printing routines retrieve these settings to determine the printing options. It is not necessary to call `WROPT` if a default value of a printing option is desired. The defaults are as follows.

IOPT	Default Value for **ISET**	Meaning
1	0	Left justified
2	1000000	Number lines before wrapping
3	–2	No paging
4	2	NaN is printed as "NaN," negative machine infinity is printed as "`-Inf`" and positive machine infinity is printed as "`Inf`."
5	0	Title only on first page.
6	3	Default format is *V*n.4.
7	2	2 spaces between columns.
8	0	Maximum row label width `MAXRLW` = 2 * `IPAGEW`/3 if matrix has one column; `MAXRLW` = `IPAGEW`/4 otherwise.
9	3	3 character indentation of row labels continued beyond one line.
10	0	Width of row label hot zone is `MAXRLW`/3 characters.

IOPT	Default Value for ISET	Meaning
11	0	Maximum column label width MAXCLW = min{max (NW + NW/2, 15), 40} for integer and real matrices, where NW is the field width for the format corresponding to the particular column. MAXCLW = min{max(NW + NW/2, 15), 83} for complex matrices, where NW is the sum of the two field widths for the formats corresponding to the particular column plus 3.
12	0	Width of column label hot zone is MAXCLW/3 characters.
13	10	Width of hot zone for titles is 10 characters.
14	0	There is no label in the upper left hand corner.
15	0	Blank line is printed.
16	0	The matrix values are aligned vertically with the last line of the associated row label.

For IOPT = 8, the default depends on the current value for the page width, IPAGEW (see PGOPT, on page1428).

2. The V and W formats are special formats that can be used to select a D, E, F, or I format so that the decimal points will be aligned. The V and W formats are specified as *Vn.d* and *Wn.d*. Here, *n* is the field width and *d* is the number of significant digits generally printed. Valid values for *n* are 3, 4, …, 40. Valid values for *d* are 1, 2, …, *n* – 2. While the V format prints trailing zeroes and a trailing decimal point, the W format does not.

Description

Routine WROPT allows the user to set or retrieve an option for printing a matrix. The options controlled by WROPT include the following: horizontal centering, a method for printing large matrices, paging, method for printing NaN (not a number) and positive and negative machine infinities, printing titles, default formats for numbers, spacing between columns, maximum widths reserved for row and column labels, indentation of row labels that continue beyond one line, widths of hot zones for breaking of labels and titles, the default heading for row labels, whether to print a blank line between invocations of routines, and vertical alignment of matrix entries with respect to row labels continued beyond one
line. (NaN and positive and negative machine infinities can be retrieved by AMACH and DMACH that are documented in the section “Machine-Dependent Constants” in the Reference Material.) Options can be set globally

(ISCOPE = 1) or temporarily for the next call to a printing routine
(ISCOPE = 0).

PGOPT

Sets or retrieves page width and length for printing.

Required Arguments

IOPT — Page attribute option. (Input)

IOPT	Description of Attribute
–1, 1	Page width.
–2, 2	Page length.

Negative values of IOPT indicate the setting IPAGE is input. Positive values of IOPT indicate the setting IPAGE is output.

IPAGE — Value of page attribute. (Input, if IOPT is negative; output, if IOPT is positive.)

IOPT	Description of Attribute	Settings for IPAGE
–1, 1	Page width (in characters)	10, 11, …
–2, 2	Page length (in lines)	10, 11, …

FORTRAN 90 Interface

Generic: `CALL PGOPT (IOPT, IPAGE)`

Specific: The specific interface name is `S_PGOPT`.

FORTRAN 77 Interface

Single: `CALL PGOPT (IOPT, IPAGE)`

Example

The following example illustrates the use of PGOPT to set the page width at 20 characters. Routine WRRRN (page 1410) is then used to print a 3×4 matrix A where $a_{ij} = i + j/10$.

```
      USE PGOPT_INT
      USE WRRRN_INT
      INTEGER    ITRING, LDA, NCA, NRA
      PARAMETER  (ITRING=0, LDA=3, NCA=4, NRA=3)
!
      INTEGER    I, IOPT, IPAGE, J
      REAL       A(LDA,NCA)
```

```
!
      DO 20  I=1, NRA
         DO 10  J=1, NCA
            A(I,J) = I + J*0.1
   10    CONTINUE
   20 CONTINUE
!                                  Set page width.
      IOPT  = -1
      IPAGE = 20
      CALL PGOPT (IOPT, IPAGE)
!                                  Print the matrix A.
      CALL WRRRN ('A', A)
      END
```

Output

```
          A
         1       2
1    1.100   1.200
2    2.100   2.200
3    3.100   3.200

         3       4
1    1.300   1.400
2    2.300   2.400
3    3.300   3.400
```

Description

Routine PGOPT is used to set or retrieve the page width or the page length for routines that perform printing.

PERMU

Rearranges the elements of an array as specified by a permutation.

Required Arguments

X — Real vector of length N containing the array to be permuted. (Input)

IPERMU — Integer vector of length N containing a permutation IPERMU(1), …, IPERMU(N) of the integers 1, …, N. (Input)

XPERMU — Real vector of length N containing the array X permuted. (Output)
If X is not needed, X and XPERMU can share the same storage locations.

Optional Arguments

N — Length of the arrays X and XPERMU. (Input)
Default: N = size (X,1).

IPATH — Integer flag. (Input)

Default: IPATH = 1.

IPATH = 1 means IPERMU represents a forward permutation, i.e., X(IPERMU(I)) is moved to XPERMU(I). IPATH = 2 means IPERMU represents a backward permutation, i.e., X(I) is moved to XPERMU(IPERMU(I)).

FORTRAN 90 Interface

Generic: CALL PERMU (X, IPERMU, XPERMU [,…])

Specific: The specific interface names are S_PERMU and D_PERMU.

FORTRAN 77 Interface

Single: CALL PERMU (N, X, IPERMU, IPATH, XPERMU)

Double: The double precision name is DPERMU.

Example

This example rearranges the array *X* using IPERMU; forward permutation is performed.

```
      USE PERMU_INT
      USE UMACH_INT
!                                  Declare variables
      INTEGER    IPATH, N
      PARAMETER  (IPATH=1, N=4)
!
      INTEGER    IPERMU(N), J, NOUT
      REAL       X(N), XPERMU(N)
!                                  Set values for  X, IPERMU
!
!                           X = ( 5.0  6.0  1.0  4.0 )
!                           IPERMU = ( 3 1 4 2 )
!
      DATA X/5.0, 6.0, 1.0, 4.0/, IPERMU/3, 1, 4, 2/
!                                  Permute X into XPERMU
      CALL PERMU (X, IPERMU, XPERMU)
!                                  Get output unit number
      CALL UMACH (2, NOUT)
!                                  Print results
      WRITE (NOUT,99999) (XPERMU(J),J=1,N)
!
99999 FORMAT (’  The output vector is:’, /, 10(1X,F10.2))
      END
```

Output

```
The Output vector is:
1.00       5.00       4.00       6.00
```

Description

Routine PERMU rearranges the elements of an array according to a permutation vector. It has the option to do both forward and backward permutations.

PERMA

Permutes the rows or columns of a matrix.

Required Arguments

A — NRA by NCA matrix to be permuted. (Input)

IPERMU — Vector of length K containing a permutation IPERMU(1), …, IPERMU(K) of the integers 1, …, K where K = NRA if the rows of A are to be permuted and K = NCA if the columns of A are to be permuted. (Input)

APER — NRA by NCA matrix containing the permuted matrix. (Output)
If A is not needed, A and APER can share the same storage locations.

Optional Arguments

NRA — Number of rows. (Input)
Default: NRA = size (A,1).

NCA — Number of columns. (Input)
Default: NCA = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement of the calling program. (Input)
Default: LDA = size (A, 1).

IPATH — Option parameter. (Input)
IPATH = 1 means the rows of A will be permuted. IPATH = 2 means the columns of A will be permuted.
Default: IPATH = 1.

LDAPER — Leading dimension of APER exactly as specified in the dimension statement of the calling program. (Input)
Default: LDAPER = size (APER,1).

FORTRAN 90 Interface

Generic: CALL PERMA (A, IPERMU, APER [,…])

Specific: The specific interface names are S_PERMA and D_PERMA.

FORTRAN 77 Interface

Single: `CALL PERMA (NRA, NCA, A, LDA, IPERMU, IPATH, APER, LDAPER)`

Double: The double precision name is `DPERMA`.

Example

This example permutes the columns of a matrix *A*.

```
      USE PERMA_INT
      USE UMACH_INT
!                                Declare variables
      INTEGER    IPATH, LDA, LDAPER, NCA, NRA
      PARAMETER  (IPATH=2, LDA=3, LDAPER=3, NCA=5, NRA=3)
!
      INTEGER    I, IPERMU(5), J, NOUT
      REAL       A(LDA,NCA), APER(LDAPER,NCA)
!                                Set values for  A, IPERMU
!                                A = ( 3.0  5.0  1.0  2.0  4.0 )
!                                    ( 3.0  5.0  1.0  2.0  4.0 )
!                                    ( 3.0  5.0  1.0  2.0  4.0 )
!
!                                IPERMU = ( 3 4 1 5 2 )
!
      DATA A/3*3.0, 3*5.0, 3*1.0, 3*2.0, 3*4.0/, IPERMU/3, 4, 1, 5, 2/
!                                Perform column permutation on A,
!                                giving APER
      CALL PERMA (A, IPERMU, APER, IPATH=IPATH)
!                                Get output unit number
      CALL UMACH (2, NOUT)
!                                Print results
      WRITE (NOUT,99999) ((APER(I,J),J=1,NCA),I=1,NRA)
!
99999 FORMAT (’  The output matrix is:’, /, 3(5F8.1,/))
      END
```

Output

```
The Output matrix is:
1.0     2.0     3.0     4.0     5.0
1.0     2.0     3.0     4.0     5.0
1.0     2.0     3.0     4.0     5.0
```

Comments

1. Workspace may be explicitly provided, if desired, by use of `P2RMA`/`DP2RMA`. The reference is:

   ```
   CALL P2RMA (NRA, NCA, A, LDA, IPERMU, IPATH, APER, LDAPER, WORK)
   ```

   The additional argument is:

   WORK — Real work vector of length `NCA`.

Description

Routine PERMA interchanges the rows or columns of a matrix using a permutation vector such as the one obtained from routines SVRBP (see the Utilities chapter in Math Library manual) or SVRGP (page 1440).

The routine PERMA permutes a column (row) at a time by calling PERMU (page 1429). This process is continued until all the columns (rows) are permuted. On completion, let B = APER and p_i = IPERMU(I), then

$$B_{ij} = A_{p_i j}$$

for all i, j.

RORDM

Reorders rows and columns of a symmetric matrix.

Required Arguments

AA — NAA by NAA symmetric matrix to be reordered. (Input)
Only elements in the upper triangle of AA are referenced.

INDAA — Index vector of length NA containing the indices of the rows/columns of AA that are being selected for inclusion into A. (Input)
INDAA(I) = J means the J-th row and column of AA will be the I-th row and column of A.

A — NAA by NAA matrix containing the reordered AA. (Output)
The first NA rows and columns of A are those specified by INDAA. The remaining elements of A contain the rows and columns not specified in INDAA

Optional Arguments

NAA — Order of the matrix AA. (Input)
Default: NAA = size (AA,2).

LDAA — Leading dimension of AA exactly as specified in the dimension statement in the calling program. (Input)
Default: LDAA = size (AA,1).

NA — Order of the reordered matrix A. (Input)
NA must be less than or equal to NAA.
Default: NA = size (INDAA,1).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

FORTRAN 90 Interface

Generic: `CALL RORDM (AA, INDAA, A [,…])`

Specific: The specific interface names are `S_RORDM` and `D_RORDM`.

FORTRAN 77 Interface

Single: `CALL RORDM (NAA, AA, LDAA, NA, INDAA, A, LDA)`

Double: The double precision name is `DRORDM`.

Example

A 4 × 4 symmetric matrix `AA` is reordered so that row/column 4, 3, and 1 of `AA` correspond to row/ column 1, 2, and 3 of `A`, respectively.

```
      USE RORDM_INT
      USE WRRRN_INT
      INTEGER    LDA, LDAA, NA, NAA
      PARAMETER  (NA=3, NAA=4, LDA=NAA, LDAA=NAA)
!
      INTEGER    INDAA(NA)
      REAL       A(LDA,NAA), AA(LDAA,NAA)
!
      DATA (AA(1,J),J=1,NAA)/10., 20., 40., 70./
      DATA (AA(2,J),J=1,NAA)/20., 30., 50., 80./
      DATA (AA(3,J),J=1,NAA)/40., 50., 60., 90./
      DATA (AA(4,J),J=1,NAA)/70., 80., 90., 100./
      DATA INDAA/4, 3, 1/
!
      CALL RORDM (AA, INDAA, A)
      CALL WRRRN ('A', A)
      END
```

Output

```
                A
        1       2       3       4
1   100.0    90.0    70.0    80.0
2    90.0    60.0    40.0    50.0
3    70.0    40.0    10.0    20.0
4    80.0    50.0    20.0    30.0
```

Comments

Workspace may be explicitly provided, if desired, by use of `R2RDM`/`DR2RDM`. The reference is:

```
CALL R2RDM (NAA, AA, LDAA, NA, INDAA, A, LDA, IWK)
```

The additional argument is

IWK — Work vector of length NAA indicating how the entire AA matrix has been reordered and returned in A. IWK(I) = J means the J-th row and column of AA are returned as the I-th row and column of A.

Description

Routine RORDM reorders the rows and columns of a symmetric matrix. Frequently in practice a sum of squares and crossproducts matrix is first computed for all variables in a data set. Then, a sum of squares and crossproducts matrix is needed for some subset of the data set variables. Alternatively, a specific order for the selected variables may be required for input into an analysis routine. For example, in regression, IMSL routine RCOV requires the sum of squares and crossproducts matrix for the independent variables and the dependent variables. Sums of squares and crossproducts for the independent variables must appear first, followed by entries for the dependent variables. Variables not in the regression analysis, but in the data set, can appear last. RORDM can be used to reorder the sum of squares and crossproducts matrix for input to RCOV.

MVNAN

Moves any rows of a matrix with the IMSL missing value code NaN (not a number) in the specified columns to the last rows of the matrix.

Required Arguments

IIND — Index vector option. (Input)

IIND	Meaning
< 0	The first −IIND columns of X are checked for NaN.
> 0	The IIND columns of X given by IND are checked for NaN.

IND — Index vector of length IIND containing the column numbers of X that are to be checked for NaN. (Input if IIND is positive)
If IIND is negative, IND is not referenced and can be a vector of length one.

X — NROW by NCOL matrix whose rows are checked for NaN (not a number). (Input/Output)
On output, the rows of X containing NaN are the last NRMISS rows of X.

ISWP — Vector of length NROW specifying the rows that were exchanged (swapped). (Output)
The number of nonzero elements in ISWP is the number of swaps that took place. ISWP(I) = J (J greater than zero) means that rows I and J of X were swapped, i.e., row I of the input X is row J of the output X and row J of the input X is row I of the output X.

Optional Arguments

NROW — Number of rows. (Input)
Default: NROW = size (X,1).

NCOL — Number of columns. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement of the calling program. (Input)
Default: LDX = size (X,1).

NRMISS — Number of rows that contained NaN in the specified columns of X. (Output)

FORTRAN 90 Interface

Generic: `CALL MVNAN (IIND, IND, X, ISWP [,…])`

Specific: The specific interface names are S_MVNAN and D_MVNAN.

FORTRAN 77 Interface

Single: `CALL MVNAN (NROW, NCOL, IIND, IND, X, LDX, ISWP, NRMISS)`

Double: The double precision name is DMVNAN.

Example 1

In this example, MVNAN is used to move rows containing NaN in columns 1 and 2 of a 5 by 3 matrix X to the last rows.

```
      USE IMSL_LIBRARIES
      INTEGER    LDX, NCOL, NROW
      PARAMETER  (NCOL=3, NROW=5, LDX=NROW)
!
      INTEGER    IIND, IND(1), ISWP(NROW), NOUT, NRMISS
      REAL       X(LDX,NCOL)
!
      DATA (X(1,J),J=1,NCOL)/1.0, 10.0, 100.0/
      DATA (X(2,J),J=1,NCOL)/2.0, 20.0, 200.0/
      DATA (X(3,J),J=1,NCOL)/3.0, 30.0, 300.0/
      DATA (X(4,J),J=1,NCOL)/4.0, 40.0, 400.0/
      DATA (X(5,J),J=1,NCOL)/5.0, 50.0, 500.0/
!
      X(2,2) = AMACH(6)
      X(4,1) = AMACH(6)
      IIND   = -2
      CALL WRRRN ('Input X', X)
      CALL MVNAN (IIND, IND, X, ISWP, NRMISS=NRMISS)
      CALL WRRRN ('Output X', X)
      CALL WRIRN ('ISWP', ISWP)
```

```
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' '
      WRITE (NOUT,*) 'NRMISS = ', NRMISS
      END
```

Output

```
          Input X
         1       2       3
1     1.0    10.0   100.0
2     2.0     NaN   200.0
3     3.0    30.0   300.0
4     NaN    40.0   400.0
5     5.0    50.0   500.0

         Output X
         1       2       3
1     1.0    10.0   100.0
2     5.0    50.0   500.0
3     3.0    30.0   300.0
4     NaN    40.0   400.0
5     2.0     NaN   200.0

ISWAP
1   0
2   5
3   0
4   0
5   0
NRMISS =   2
```

Description

Routine MVNAN moves any rows containing the IMSL missing value code NaN (not a number) in the specified columns to the last rows of the matrix. NaN can be assigned to the element of a matrix by using routine AMACH. (See the following examples and the section "Machine-Dependent Constants" in the Reference Material.)

The columns that are checked for NaN are specified through arguments IIND and IND. Let c_1, c_2, …, c_k be the column numbers specified. The main steps performed by MVNAN are as follows:

1. Initialize the NROW elements of ISWAP to zero.

2. Exchange (swap) the first row of X containing NaN in at least one of the specified columns c_1, c_2, …, c_k with the last row of X that does not contain NaN in any of the specified columns c_1, c_2, …, c_k. Let i and j ($i < j$) correspond to the rows that are exchanged. Set the i-th element of ISWP to j.

3. Starting with the $(i + 1)$-st row of X, exchange the first row containing NaN with the last row not containing NaN, beginning from row $j - 1$ and continuing backward. Reset i and j ($i < j$) to correspond to the rows exchanged.

4. Continue in this manner until all rows have been examined for NaN.

Additional Example

Example 2

In this example, MVNAN is used to move rows containing NaN in column 1 and 3 of a 5 by 3 matrix X to the last rows.

```
      USE IMSL_LIBRARIES
      INTEGER    LDX, NCOL, NROW
      PARAMETER  (NCOL=3, NROW=5, LDX=NROW)
!
      INTEGER    IIND, IND(2), ISWP(NROW), NOUT, NRMISS
      REAL       X(LDX,NCOL)
!
      DATA (X(1,J),J=1,NCOL)/1.0, 10.0, 100.0/
      DATA (X(2,J),J=1,NCOL)/2.0, 20.0, 200.0/
      DATA (X(3,J),J=1,NCOL)/3.0, 30.0, 300.0/
      DATA (X(4,J),J=1,NCOL)/4.0, 40.0, 400.0/
      DATA (X(5,J),J=1,NCOL)/5.0, 50.0, 500.0/
      DATA IND/1, 3/
!
      X(2,2) = AMACH(6)
      X(4,1) = AMACH(6)
      IIND   = 2
      CALL WRRRN ('Input X', X)
      CALL MVNAN (IIND, IND, X, ISWP, NRMISS=NRMISS)
      CALL WRRRN ('Output X', X)
      CALL WRIRN ('ISWP', ISWP)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' '
      WRITE (NOUT,*) 'NRMISS = ', NRMISS
      END
```

Output

```
          Input X
            1       2       3
1       1.0    10.0   100.0
2       2.0     NaN   200.0
3       3.0    30.0   300.0
4       NaN    40.0   400.0
5       5.0    50.0   500.0

          Output X
            1       2       3
1       1.0    10.0   100.0
2       2.0     NaN   200.0
3       3.0    30.0   300.0
4       5.0    50.0   500.0
5       NaN    40.0   400.0
```

```
ISWP
1   0
2   0
3   0
4   5
5   0
NRMISS =   1
```

SVRGN

Sorts a real array by algebraically increasing value.

Required Arguments

RA — Vector of length N containing the array to be sorted. (Input)

RB — Vector of length N containing the sorted array. (Output)
If RA is not needed, RA and RB can share the same storage locations.

Optional Arguments

N — Number of elements in the array to be sorted. (Input)
Default: N = size (RA,1).

FORTRAN 90 Interface

Generic: `CALL SVRGN (RA, RB [,…])`

Specific: The specific interface names are `S_SVRGN` and `D_SVRGN`.

FORTRAN 77 Interface

Single: `CALL SVRGN (N, RA, RB)`

Double: The double precision name is `DSVRGN`.

Example

This example sorts the 10-element array RA algebraically.

```
      USE SVRGN_INT
      USE UMACH_INT
!                                  Declare variables
      PARAMETER  (N=10)
      REAL       RA(N), RB(N)
!                                  Set values for  RA
!     RA = ( -1.0  2.0  -3.0  4.0  -5.0  6.0  -7.0  8.0  -9.0  10.0 )
!
      DATA RA/-1.0, 2.0, -3.0, 4.0, -5.0, 6.0, -7.0, 8.0, -9.0, 10.0/
!                                  Sort RA by algebraic value into RB
```

```
      CALL SVRGN (RA, RB)
!                                  Print results
      CALL UMACH (2,NOUT)
      WRITE (NOUT, 99999) (RB(J),J=1,N)
!
99999 FORMAT (’  The output vector is:’, /, 10(1X,F5.1))
      END
```

Output

```
The Output vector is:
-9.0  -7.0  -5.0  -3.0  -1.0   2.0   4.0   6.0   8.0  10.0
```

Description

Routine SVRGN sorts the elements of an array, A, into ascending order by algebraic value. The array A is divided into two parts by picking a central element T of the array. The first and last elements of A are compared with T and exchanged until the three values appear in the array in ascending order. The elements of the array are rearranged until all elements greater than or equal to the central element appear in the second part of the array and all those less than or equal to the central element appear in the first part. The upper and lower subscripts of one of the segments are saved, and the process continues iteratively on the other segment. When one segment is finally sorted, the process begins again by retrieving the subscripts of another unsorted portion of the array. On completion, $A_j \le A_i$ for $j < i$. For more details, see Singleton (1969), Griffin and Redish (1970), and Petro (1970).

SVRGP

Sorts a real array by algebraically increasing value and return the permutation that rearranges the array.

Required Arguments

RA — Vector of length N containing the array to be sorted. (Input)

RB — Vector of length N containing the sorted array. (Output)
If RA is not needed, RA and RB can share the same storage locations.

IPERM — Vector of length N. (Input/Output)
On input, IPERM should be initialized to the values 1, 2, …, N. On output, IPERM contains a record of permutations made on the vector RA.

Optional Arguments

N — Number of elements in the array to be sorted. (Input)
Default: N = size (IPERM,1).

FORTRAN 90 Interface

Generic: CALL SVRGP (RA, RB, IPERM [,…])

Specific: The specific interface names are S_SVRGP and D_SVRGP.

FORTRAN 77 Interface

Single: CALL SVRGP (N, RA, RB, IPERM)

Double: The double precision name is DSVRGP.

Example

This example sorts the 10-element array RA algebraically.

```
      USE SVRGP_INT
      USE UMACH_INT
!                                  Declare variables
      PARAMETER  (N=10)
      REAL       RA(N), RB(N)
      INTEGER    IPERM(N)
!                                  Set values for  RA and IPERM
!     RA    = ( 10.0  -9.0  8.0  -7.0  6.0  5.0  4.0  -3.0  -2.0  -1.0 )
!
!     IPERM = ( 1  2  3  4  5  6  7  8  9  10)
!
      DATA RA/10.0, -9.0, 8.0, -7.0, 6.0, 5.0, 4.0, -3.0, -2.0, -1.0/
      DATA IPERM/1, 2, 3, 4, 5, 6, 7, 8, 9, 10/
!                                  Sort RA by algebraic value into RB
      CALL SVRGP (RA, RB, IPERM)
!                                  Print results
      CALL UMACH (2,NOUT)
      WRITE (NOUT, 99998) (RB(J),J=1,N)
      WRITE (NOUT, 99999) (IPERM(J),J=1,N)
!
99998 FORMAT (’  The output vector is:’, /, 10(1X,F5.1))
99999 FORMAT (’  The permutation vector is:’, /, 10(1X,I5))
      END
```

Output

```
The output vector is:
-9.0  -7.0  -3.0  -2.0  -1.0   4.0   5.0   6.0   8.0  10.0

The permutation vector is:
2     4     8     9    10     7     6     5     3     1
```

Comments

For wider applicability, integers (1, 2, …, N) that are to be associated with RA(I) for I = 1, 2, …, N may be entered into IPERM(I) in any order. Note that these integers must be unique.

Description

Routine SVRGP sorts the elements of an array, *A*, into ascending order by algebraic value, keeping a record in *P* of the permutations to the array *A*. That is, the elements of *P* are moved in

the same manner as are the elements in A as A is being sorted. The routine SVRGP uses the algorithm discussed in SVRGN (page 1439). On completion, $A_j \leq A_i$ for $j < i$.

SVIGN

Sorts an integer array by algebraically increasing value.

Required Arguments

IA — Integer vector of length N containing the array to be sorted. (Input)

IB — Integer vector of length N containing the sorted array. (Output)
If IA is not needed, IA and IB can share the same storage locations.

Optional Arguments

N — Number of elements in the array to be sorted. (Input)
Default: N = size (IA,1).

FORTRAN 90 Interface

Generic: `CALL SVIGN (IA, IB [,…])`

Specific: The specific interface name is `S_SVIGN`.

FORTRAN 77 Interface

Single: `CALL SVIGN (N, IA, IB)`

Example

This example sorts the 10-element array IA algebraically.

```
      USE SVIGN_INT
      USE UMACH_INT
!                                  Declare variables
      PARAMETER  (N=10)
      INTEGER    IA(N), IB(N)
!                                  Set values for  IA
!     IA = ( -1  2  -3  4  -5  6  -7  8  -9  10 )
!
      DATA IA/-1, 2, -3, 4, -5, 6, -7, 8, -9, 10/
!                                  Sort IA by algebraic value into IB
      CALL SVIGN (IA, IB)
!                                  Print results
      CALL UMACH (2,NOUT)
      WRITE (NOUT, 99999) (IB(J),J=1,N)
!
99999 FORMAT (’  The output vector is:’, /, 10(1X,I5))
      END
```

Output

```
The Output vector is:
-9    -7    -5    -3    -1     2     4     6     8    10
```

Description

Routine SVIGN sorts the elements of an integer array, A, into ascending order by algebraic value. The routine SVIGN uses the algorithm discussed in SVRGN (page 1439). On completion, $A_j \le A_i$ for $j < i$.

SVIGP

Sorts an integer array by algebraically increasing value and return the permutation that rearranges the array.

Required Arguments

IA — Integer vector of length N containing the array to be sorted. (Input)

IB — Integer vector of length N containing the sorted array. (Output)
If IA is not needed, IA and IB can share the same storage locations.

IPERM — Vector of length N. (Input/Output)
On input, IPERM should be initialized to the values 1, 2, …, N. On output, IPERM contains a record of permutations made on the vector IA.

Optional Arguments

N — Number of elements in the array to be sorted. (Input)
Default: N = size (IPERM,1).

FORTRAN 90 Interface

Generic: CALL SVIGP (IA, IB, IPERM [,…])

Specific: The specific interface name is S_SVIGP.

FORTRAN 77 Interface

Single: CALL SVIGP (N, IA, IB, IPERM)

Example

This example sorts the 10-element array IA algebraically.

```
      USE SVIGP_INT
      USE UMACH_INT
!                                  Declare variables
```

```
      PARAMETER  (N=10)
      INTEGER    IA(N), IB(N), IPERM(N)
!                                Set values for  IA and IPERM
!     IA    = ( 10  -9  8  -7  6  5  4  -3  -2  -1 )
!
!     IPERM = ( 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 )
!
      DATA IA/10, -9, 8, -7, 6, 5, 4, -3, -2, -1/
      DATA IPERM/1, 2, 3, 4, 5, 6, 7, 8, 9, 10/
!                                Sort IA by algebraic value into IB
      CALL SVIGP (IA, IB, IPERM)
!                                Print results
      CALL UMACH (2,NOUT)
      WRITE (NOUT, 99998) (IB(J),J=1,N)
      WRITE (NOUT, 99999) (IPERM(J),J=1,N)
!
99998 FORMAT (' The output vector is:', /, 10(1X,I5))
99999 FORMAT (' The permutation vector is:', /, 10(1X,I5))
      END
```

Output

```
The Output vector is:
-9    -7    -3    -2    -1     4     5     6     8    10

The permutation vector is:
2     4     8     9    10     7     6     5     3     1
```

Comments

For wider applicability, integers (1, 2, …, N) that are to be associated with IA(I) for I = 1, 2, …, N may be entered into IPERM(I) in any order. Note that these integers must be unique.

Description

Routine SVIGP sorts the elements of an integer array, A, into ascending order by algebraic value, keeping a record in P of the permutations to the array A. That is, the elements of P are moved in the same manner as are the elements in A as A is being sorted. The routine SVIGP uses the algorithm discussed in SVRGN (page 1439). On completion, $A_j \leq A_i$ for $j < i$.

SCOLR

Sorts columns of a real rectangular matrix using keys in rows.

Required Arguments

X — NRX by NCX matrix. (Input, if IRET = 1; input/output if IRET = 0)
On input, X contains the matrix to be sorted. If IRET = 0, the output X contains the sorted matrix.

INDKEY — Vector of length NKEY giving the row numbers of X which are to be used in the sort. (Input)

IPERM — Permutation vector of length NCX specifying the rearrangement of the columns. (Output)
IPERM (I) = J means column I of the sorted X is column J of the unsorted X.

NGROUP — Number of groups. (Output)
The columns of the sorted X are partitioned into groups. A group contains columns that are equal with respect to the method of comparison. NGROUP is the number of groups of different columns.

NI — Vector of length NGROUP containing the number of columns in each group. (Output)
The first NI(1) columns of the sorted X are group number 1; the next NI(2) columns of the sorted X are group number 2;… the last NI(NGROUP) columns of the sorted X are group number NGROUP. If NGROUP is not known prior to the invocation of this routine, NCX(an upper bound for NGROUP) can be used as the dimension of NI.

Optional Arguments

NRX — Number of rows of X. (Input)
Default: NRX = size (X,1).

NCX — Number of columns of X. (Input)
Default: NCX = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

ICOMP — Option giving the method of comparison of the column vectors. (Input)
Default: ICOMP = 0.

ICOMP	Action
0	Elementwise, by algebraic values
1	Elementwise, by absolute values

IORDR — Option giving the sorting order. (Input)
Default: IORDR = 0.

IORDR	Action
0	Ascending
1	Descending

IRET — Option for determining whether the columns of X are to be permuted. (Input)
Default: IRET = 0.

IRET **Action**

0 The columns of X are sorted.

1 X is unchanged (detached key sort).

NKEY — Number of rows of X on which to sort. (Input)
Default: NKEY = size (INDKEY,1).

FORTRAN 90 Interface

Generic: `CALL SCOLR (X, INDKEY, IPERM, NGROUP, NI [,…])`

Specific: The specific interface names are `S_SCOLR` and `D_SCOLR`.

FORTRAN 77 Interface

Single:
```
CALL SCOLR (NRX, NCX, X, LDX, ICOMP, IORDR, IRET, NKEY,
INDKEY, IPERM, NGROUP, NI)
```

Double: The double precision name is `DSCOLR`.

Example

The columns of a 5 × 10 matrix *X* are sorted in descending order by absolute value using rows 1, 2, 3, and 5 as the keys. The permutations to put the columns of *X* in order are returned. The input matrix *X* is not changed.

```
      USE SCOLR_INT
      USE WRRRL_INT
      USE WRIRL_INT
      USE UMACH_INT
      INTEGER    LDX, NCX, NKEY, NRX
      PARAMETER  (NCX=10, NKEY=4, NRX=5, LDX=NRX)
!
      INTEGER    ICOMP, INDKEY(NKEY), IORDR, IPERM(NCX), IRET, NI(NCX), &
                 NGROUP, NOUT
      REAL       X(LDX,NCX)
      CHARACTER  CLABEL(1)*10, FMT*10, RLABEL(1)*23
!
      DATA CLABEL(1)/'NONE'/, RLABEL(1)/'NONE'/
      DATA X/-1.0, -10.0, -11.0, 10.0, -1.0, 2.0, 20.0, 22.0, -20.0, &
           -2.0, -3.0, -30.0, 33.0, 30.0, -3.0, 4.0, 40.0, 44.0, &
           -40.0, -4.0, -5.0, -50.0, 55.0, 50.0, -5.0, -1.0, 60.0, &
           -66.0, -60.0, 6.0, 2.0, -70.0, -77.0, 70.0, 7.0, -3.0, &
           -30.0, -88.0, 80.0, 8.0, 4.0, 40.0, -99.0, -90.0, 9.0, &
           -5.0, -50.0, -100.0, 100.0, 10.0/
      DATA INDKEY/1, 2, 3, 5/
!
      ICOMP = 1
      IORDR = 1
```

```
      IRET  = 1
      CALL SCOLR (X, INDKEY, IPERM, NGROUP, NI, ICOMP=ICOMP, &
                  IORDR=IORDR, IRET=IRET)
!
      FMT       = '(F6.1)'
      RLABEL(1) = 'NONE'
      CALL WRRRL ('X', X, RLABEL, CLABEL)
!
      FMT       = '(I4)'
      RLABEL(1) = 'IPERM = '
      CALL WRIRL ('%/', IPERM, RLABEL, CLABEL, 1, NCX, 1, FMT='(I4)')
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'NGROUP = ', NGROUP
!
      RLABEL(1) = 'NI = '
      CALL WRIRL ('%/', NI, RLABEL, CLABEL, 1, NGROUP, 1, FMT='(I4)')
!
      END
```

Output

```
                                        X
 -1.0    2.0   -3.0     4.0    -5.0    -1.0     2.0    -3.0     4.0    -5.0
-10.0   20.0  -30.0    40.0   -50.0    60.0   -70.0   -30.0    40.0   -50.0
-11.0   22.0   33.0    44.0    55.0   -66.0   -77.0   -88.0   -99.0  -100.0
 10.0  -20.0   30.0   -40.0    50.0   -60.0    70.0    80.0   -90.0   100.0
 -1.0   -2.0   -3.0    -4.0    -5.0     6.0     7.0     8.0     9.0    10.0
IPERM =    10      5      9      4      8      3      7      2      6      1
NGROUP =   10
NI =      1      1      1      1      1      1      1      1      1      1
```

Comments

1. Workspace may be explicitly provided, if desired, by use of S2OLR/DS2OLR. The reference is:

   ```
   CALL S2OLR (NRX, NCX, X, LDX, ICOMP, IORDR, IRET, NKEY, INDKEY,
   IPERM, NGROUP, NI, WK, IWK)
   ```

   The additional arguments are as follows:

   WK — Work vector of length 2 * *m*.

   IWK — Work vector of length *m* + INT(2.8854 ln(*m*)) + 2.

2. When X is sorted by algebraic value (ICOMP = 0) in ascending order, the resulting array X is such that:
   For *i* = 1, 2, …, NCX – 1, X(INDKEY(1), *i*) ≤ X(INDKEY(1), *i* + 1)
   For *k* = 2, …, NKEY, if X(INDKEY(*j*), *i*) = X(INDKEY(*j*), *i* + 1) for *j* = 1, 2, …, *k* – 1, then X(INDKEY (*k*), *i*) ≤ X(INDKEY(*k*), *i* + 1).
   When ICOMP = 1, the absolute values are compared instead.

Description

Routine SCOLR sorts the columns of a real matrix *X* using particular rows in *X* as the keys. One of two methods for comparing the columns can be used for sorting.

1. Algebraic with the first key as the most significant, the second key next most significant and so forth.

2. Absolute values with the first key as the most significant, the second key next most significant and so forth.

The columns of *X* can be put in ascending or descending order.

The routine is useful for data containing classification variables. Routine CSTAT (see Chapter 1, Basic Statistics) can be used to form the cells and frequency counts for a multi-way table from data. The columns of the output matrix contain the values of each combination of values of the classification variables along with the tallies. SCOLR can then be used to sort the columns of this output matrix using the classification variables as keys.

SCOLR is based on a quicksort method given by Singleton (1969). Modifications by Griffin and Redish (1970) and Petro (1970) are incorporated.

SROWR

Sorts rows of a real rectangular matrix using keys in columns.

Required Arguments

X — NROW by NCOL matrix. (Input, if IRET = 1; input/output if IRET = 0)
On input, X contains the matrix to be sorted. If IRET = 0, the output X contains the sorted matrix.

INDKEY — Vector of length NKEY giving the column numbers of X which are to be used in the sort. (Input)

IPERM — Permutation vector of length NROW specifying the rearrangement of the rows. (Output)
IPERM(I) = J means row I of the sorted X is row J of the unsorted X.

NGROUP — Number of groups. (Output, if IRET ≤ 1)
The rows of the sorted X are partitioned into groups. A group contains rows that are equal with respect to the method of comparison. NGROUP is the number of groups of different rows.

NI — Vector of length NGROUP containing the number of rows in each group. (Output, if IRET ≤ 1)

The first NI(1) rows of the sorted X are group number 1. The next NI(2) rows of the sorted X are group number 2. … The last NI(NGROUP) rows of the sorted X are group

number NGROUP. If NGROUP is not known prior to the invocation of this routine, NROW(an upper bound for NGROUP) can be used as the dimension of NI. If IRET ≥ 2, NI is not referenced and can be a vector of length one.

Optional Arguments

NROW — Number of rows of X. (Input)
Default: NROW = size (X,1).

NCOL — Number of columns of X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement of the calling program. (Input)
Default: LDX = size (X,1).

ICOMP — Option giving the method of comparison of the row vectors. (Input)
Default: ICOMP = 0.

ICOMP	Action
0	Elementwise, by algebraic values
1	Elementwise, by absolute values

IORDR — Option giving the sorting order. (Input)
Default: IORDR = 0.

IORDR	Action
0	Ascending
1	Descending

IRET — Option to indicate information returned. (Input)
Default: IRET = 0.

IRET	Action
0	The sorted X is returned along with NGROUP and NI.
1	X is unchanged (detached key sort) and NGROUP and NI are returned.
2	The sorted X is returned, but NGROUP and NI are not returned.
3	X is unchanged (detached key sort) and NGROUP and NI are not returned.

NKEY — Number of columns of X on which to sort. (Input)
Default: NKEY = size (INDKEY,1).

NRMISS — Number of rows that contained NaN in the columns of X used in the sort. (Output)
These rows are considered as a separate group from the other NGROUP groups and are put as the last NRMISS rows of the sorted X.

FORTRAN 90 Interface

Generic: `CALL SROWR (X, INDKEY, IPERM, NGROUP, NI [,…])`

Specific: The specific interface names are S_SROWR and D_SROWR.

FORTRAN 77 Interface

Single:
```
CALL SROWR (NROW, NCOL, X, LDX, ICOMP, IORDR, IRET, NKEY,
INDKEY, IPERM, NGROUP, NI, NRMISS)
```

Double: The double precision name is DSROWR.

Example

The rows of a 10 × 3 matrix X are sorted in ascending order by algebraic value using columns 2 and 3 as the keys. The permutations to put the rows of the input X into sorted order are returned along with the sorted X.

```
      USE IMSL_LIBRARIES
      INTEGER    LDX, NCOL, NKEY, NROW
      PARAMETER  (NCOL=3, NKEY=2, NROW=10, LDX=NROW)
!
      INTEGER    ICOMP, INDKEY(NKEY), IORDR, IPERM(NROW), IRET, &
                 NGROUP, NI(NROW), NOUT, NRMISS
      REAL       X(LDX,NCOL)
!
      DATA (X(1,J),J=1,3)/1.0, 1., 1./
      DATA (X(2,J),J=1,3)/2.0, 2., 1./
      DATA (X(3,J),J=1,3)/3.0, 1., 1./
      DATA (X(4,J),J=1,3)/4.0, 1., 1./
      DATA (X(5,J),J=1,3)/5.0, 2., 2./
      DATA (X(6,J),J=1,3)/6.0, 1., 2./
      DATA (X(7,J),J=1,3)/7.0, 1., 2./
      DATA (X(8,J),J=1,3)/8.0, 1., 1./
      DATA (X(9,J),J=1,3)/9.0, 2., 2./
      DATA (X(10,J),J=1,3)/9.0, 1., 1./
      DATA INDKEY/2, 3/
!
      X(5,3) = AMACH(6)
      X(7,2) = AMACH(6)
      CALL SROWR (X, INDKEY, IPERM, NGROUP, NI, NRMISS=NRMISS)
      CALL WRRRN ('X', X)
```

```
      CALL WRIRN ('IPERM', IPERM)
      CALL WRIRN ('NI', NI, NGROUP, 1, NGROUP)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' '
      WRITE (NOUT,*) 'NRMISS = ', NRMISS
      END
```

Output

```
              X
         1       2       3
 1   1.000   1.000   1.000
 2   9.000   1.000   1.000
 3   3.000   1.000   1.000
 4   4.000   1.000   1.000
 5   8.000   1.000   1.000
 6   6.000   1.000   2.000
 7   2.000   2.000   1.000
 8   9.000   2.000   2.000
 9   7.000     NaN   2.000
10   5.000   2.000     NaN

 IPERM
 1    1
 2   10
 3    3
 4    4
 5    8
 6    6
 7    2
 8    9
 9    7
10    5

  NI
 1   5
 2   1
 3   1
 4   1
NRMISS =   2
```

Comments

1. Workspace may be explicitly provided, if desired, by use of S2OWR/DS2OWR. The reference is:

   ```
   CALL S2OWR (NROW, NCOL, X, LDX, ICOMP, IORDR, IRET, NKEY,
   INDKEY, IPERM, NGROUP, NI, NRMISS, WK, IWK)
   ```

   The additional arguments are as follows:

   WK — Work vector of length 2 * *m*.

   IWK — Work vector of length *m* + INT(2.8854 * ln(*m*)) + 2.

2. When X is sorted by algebraic values (ICOMP = 0), in ascending order, the resulting array X is such that:
   For $i = 1, 2, \ldots,$ NROW $- 1$, X(i, INDKEY(1)) $\leq$ X($i + 1$, INDKEY(1)).
   For k = 2, …, NKEY, if X(i, INDKEY(j)) = X($i + 1$, INDKEY(j)) for $j = 1, 2, \ldots, k - 1$; then X(i, INDKEY(k)) $\leq$ X($i + 1$, INDKEY(k)).
   When ICOMP = 1, the absolute values are compared instead.

Description

Routine SROWR sorts the rows of a real matrix X using particular rows in X as the keys. One of two methods for comparing the rows can be used for sorting.

1. Algebraic with the first key as the most significant, the second key next most significant and so forth.

2. Absolute values with the first key as the most significant, the second key next most significant and so forth.

The rows of X can be put in ascending or descending order.

The routine is useful for grouping data based on values of specified variables. The rows of X containing the IMSL missing value code NaN (not a number) in at least one of the specified columns are considered as an additional group of NRMISS rows. These rows are moved to the end of the sorted X. SROWR is based on a quicksort method given by Singleton (1969). Modifications by Griffin and Redish (1970) and Petro (1970) are incorporated.

SRCH

Searches a sorted vector for a given scalar and return its index.

Required Arguments

VALUE — Scalar to be searched for in Y. (Input)

X — Vector of length N * INCX. (Input)
Y is obtained from X for I = 1, 2, …, N by Y(I) = X(1 + (I − 1) * INCX). Y(1), Y(2), …, Y(N) must be in ascending order.

INDEX — Index of Y pointing to VALUE. (Output)
If INDEX is positive, VALUE is found in Y. If INDEX is negative, VALUE is not found in Y.

INDEX	**Location of VALUE**
1 thru N	VALUE = Y(INDEX)
–1	VALUE < Y(1) or N = 0

INDEX	Location of VALUE
–N thru –2	Y(–INDEX – 1) < VALUE < Y(INDEX)
–(N + 1)	VALUE > Y(N)

Optional Arguments

N — Length of vector Y. (Input)
Default: N = (size (X,1)) / INCX.

INCX — Displacement between elements of X. (Input)
INCX must be greater than zero.
Default: INCX = 1.

FORTRAN 90 Interface

Generic: CALL SRCH (VALUE, X, INDEX [,…])

Specific: The specific interface names are S_SRCH and D_SRCH.

FORTRAN 77 Interface

Single: CALL SRCH (N, VALUE, X, INCX, INDEX)

Double: The double precision name is DSRCH.

Example

This example searches a real vector sorted in ascending order for the value 653.0. The problem is discussed by Knuth (1973, pages 407–409).

```
      USE SRCH_INT
      USE UMACH_INT
      INTEGER    N
      PARAMETER  (N=16)
!
      INTEGER    INDEX, NOUT
      REAL       VALUE, X(N)
!
      DATA X/61.0, 87.0, 154.0, 170.0, 275.0, 426.0, 503.0, 509.0, &
          512.0, 612.0, 653.0, 677.0, 703.0, 765.0, 897.0, 908.0/
!
      VALUE = 653.0
      CALL SRCH (VALUE, X, INDEX)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'INDEX = ', INDEX
      END
```

Output

```
INDEX =   11
```

Description

Routine SRCH searches a real vector x (stored in X), whose n elements are sorted in ascending order for a real number c (stored in VALUE). If c is found in x, its index i (stored in INDEX) is returned so that $x_i = c$. Otherwise, a negative number i is returned for the index. Specifically,

if $1 \le i \le n$	then $x_i = c$
if $i = -1$	then $c < x_1$ or $n = 0$
if $-n \le I \le -2$	then $x_{-i-1} < c < x_{-I}$
if $i = -(n + 1)$	then $c > x_n$

The argument INCX is useful if a row of a matrix, for example, row number I of a matrix X, must be searched. The elements of row I are assumed to be in ascending order. In this case, set INCX equal to the leading dimension of X exactly as specified in the dimension statement in the calling program. With X declared

```
REAL X(LDX,N)
```

the invocation

```
CALL SRCH (VALUE, X(I, 1), INDEX, N=N, INCX=LDX)
```

returns an index that will reference a column number of X.

Routine SRCH performs a binary search. The routine is an implementation of algorithm *B* discussed by Knuth (1973, pages 407–411).

ISRCH

Searches a sorted integer vector for a given integer and return its index.

Required Arguments

IVALUE — Scalar to be searched for in IY. (Input)

IX — Vector of length N * INCX. (Input)
IY is obtained from IX for I = 1, 2, …, N by IY(I) = IX(1 + (I – 1) * INCX). IY(1), IY(2), …, IY(N) must be in ascending order.

INDEX — Index of IY pointing to IVALUE. (Output)
If INDEX is positive, IVALUE is found in IY. If INDEX is negative, IVALUE is not found in IY.

INDEX	Location of VALUE
1 thru N	IVALUE = IY(INDEX)
–1	IVALUE < IY(1) or N = 0
–N thru –2	IY(–INDEX – 1) < IVALUE < IY(–INDEX)
–(N + 1)	IVALUE > Y(N)

Optional Arguments

N — Length of vector IY. (Input)
Default: N = size (IX,1) / INCX.

INCX — Displacement between elements of IX. (Input)
INCX must be greater than zero.
Default: INCX = 1.

FORTRAN 90 Interface

Generic: `CALL ISRCH (IVALUE, IX, INDEX [,…])`

Specific: The specific interface name is `S_ISRCH`.

FORTRAN 77 Interface

Single: `CALL ISRCH (N, IVALUE, IX, INCX, INDEX)`

Example

This example searches an integer vector sorted in ascending order for the value 653. The problem is discussed by Knuth (1973, pages 407–409).

```
      USE ISRCH_INT
      USE UMACH_INT
      INTEGER    N
      PARAMETER  (N=16)
!
      INTEGER    INDEX, NOUT
      INTEGER    IVALUE, IX(N)
!
      DATA IX/61, 87, 154, 170, 275, 426, 503, 509, 512, 612, 653, 677, &
             703, 765, 897, 908/
!
      IVALUE = 653
      CALL ISRCH (IVALUE, IX, INDEX)
```

```
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'INDEX = ', INDEX
      END
```

Output

```
INDEX =   11
```

Description

Routine ISRCH searches an integer vector x (stored in IX), whose n elements are sorted in ascending order for an integer c (stored in IVALUE). If c is found in x, its index i (stored in INDEX) is returned so that $x_i = c$. Otherwise, a negative number i is returned for the index. Specifically,

if $1 \le i \le n$	Then $x_i = c$
if $i = -1$	Then $c < x_1$ or $n = 0$
if $-n \le i \le -2$	Then $x_{-i-1} < c < x_{-i}$
if $i = -(n + 1)$	Then $c > x_n$

The argument INCX is useful if a row of a matrix, for example, row number I of a matrix IX, must be searched. The elements of row I are assumed to be in ascending order. Here, set INCX equal to the leading dimension of IX exactly as specified in the dimension statement in the calling program. With IX declared

```
INTEGER IX(LDIX,N)
```

the invocation

```
CALL ISRCH (VALUE, X(I, 1), INDEX, N=N, INCX=LDIX)
```

returns an index that will reference a column number of IX.

The routine ISRCH performs a binary search. The routine is an implementation of algorithm *B* discussed by Knuth (1973, pages 407–411).

SSRCH

Searches a character vector, sorted in ascending ASCII order, for a given string and return its index.

Required Arguments

N — Length of vector CHY. (Input)
Default: N = size (CHX,1) / INCX.

STRING — Character string to be searched for in CHY. (Input)

CHX — Vector of length N * INCX containing character strings. (Input)
CHY is obtained from CHX for I = 1, 2, …, N by CHY(I) = CHX(1 + (I – 1) * INCX). CHY(1), CHY(2), …, CHY(N) must be in ascending ASCII order.

INCX — Displacement between elements of CHX. (Input)
INCX must be greater than zero.
Default: INCX = 1.

INDEX — Index of CHY pointing to STRING. (Output)
If INDEX is positive, STRING is found in CHY. If INDEX is negative, STRING is not found in CHY.

INDEX	**Location of STRING**
1 thru N	STRING = CHY(INDEX)
–1	STRING < CHY(1) or N = 0
–N thru –2	CHY(–INDEX – 1) < STRING < CHY(–INDEX)
–(N + 1)	STRING > CHY(N)

FORTRAN 90 Interface

Generic: `CALL SSRCH (N, STRING, CHX, INCX, INDEX)`

Specific: The specific interface name is SSRCH.

FORTRAN 77 Interface

Single: `CALL SSRCH (N, STRING, CHX, INCX, INDEX)`

Example

This example searches a CHARACTER * 2 vector containing 9 character strings, sorted in ascending ASCII order, for the value 'CC'.

```
      USE SSRCH_INT
      USE UMACH_INT
      INTEGER    N
      PARAMETER  (N=9)
!
      INTEGER    INDEX, NOUT
      CHARACTER  CHX(N)*2, STRING*2
!
      DATA CHX/'AA', 'BB', 'CC', 'DD', 'EE', 'FF', 'GG', 'HH', &
          'II'/
!
```

```
      INCX   = 1
      STRING = 'CC'
      CALL SSRCH (N, STRING, CHX, INCX, INDEX)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'INDEX = ', INDEX
      END
```

Output

```
INDEX =   3
```

Description

Routine SSRCH searches a vector of character strings x (stored in CHX), whose n elements are sorted in ascending ASCII order, for a character string c (stored in STRING). If c is found in x, its index i (stored in INDEX) is returned so that $x_i = c$. Otherwise, a negative number i is returned for the index. Specifically,

if $1 \le i \le n$	Then $x_i = c$
if $i = -1$	Then $c < x_1$ or $n = 0$
if $-n \le I \le -2$	Then $x_{-i-1} < c < x_{-i}$
if $i = -(n + 1)$	Then $c > x_n$

Here, "<" and ">" are in reference to the ASCII collating sequence. For comparisons made between character strings c and x_i with different lengths, the shorter string is considered as if it were extended on the right with blanks to the length of the longer string. (SSRCH uses FORTRAN intrinsic functions LLT and LGT.)

The argument INCX is useful if a row of a matrix, for example, row number I of a matrix CHX, must be searched. The elements of row I are assumed to be in ascending ASCII order. In this case, set INCX equal to the leading dimension of CHX exactly as specified in the dimension statement in the calling program. With CHX declared

```
CHARACTER * 7 CHX(LDCHX,N)
```

the invocation

```
CALL SSRCH (STRING, CHX(I:,1), INDEX, N=N, INCX=LDCHX)
```

returns an index that will reference a column number of CHX.

Routine SSRCH performs a binary search. The routine is an implementation of algorithm *B* discussed by Knuth (1973, pages 407–411).

ACHAR

This function returns a character given its ASCII value.

Function Return Value

ACHAR — CHARACTER * 1 string containing the character in the I-th position of the ASCII collating sequence. (Output)

Required Arguments

I — Integer ASCII value of the character desired. (Input)
I must be greater than or equal to zero and less than or equal to 127.

FORTRAN 90 Interface

Generic: ACHAR (I)

Specific: The specific interface name is ACHAR.

FORTRAN 77 Interface

Single: ACHAR (I)

Example

This example returns the character of the ASCII value 65.

```
      USE ACHAR_INT
      USE UMACH_INT
      INTEGER    I, NOUT
!
      CALL UMACH (2, NOUT)
!                                Get character for ASCII value
!                                of 65 ('A')
      I = 65
      WRITE (NOUT,99999) I, ACHAR(I)
!
99999 FORMAT (' For the ASCII value of ', I2, ', the character is : ', &
             A1)
      END
```

Output

```
For the ASCII value of 65, the character is : A
```

Description

Routine ACHAR returns the character of the input ASCII value. The input value should be between 0 and 127. If the input value is out of range, the value returned in ACHAR is machine dependent.

IACHAR

This function returns the integer ASCII value of a character argument.

Function Return Value

IACHAR — Integer ASCII value for CH. (Output)
The character CH is in the IACHAR-th position of the ASCII collating sequence.

Required Arguments

CH — Character argument for which the integer ASCII value is desired. (Input)

FORTRAN 90 Interface

Generic: IACHAR(CH)

Specific: The specific interface name is IACHAR.

FORTRAN 77 Interface

Single: IACHAR(CH)

Example

This example gives the ASCII value of character A.

```
      USE IACHAR_INT
      INTEGER    NOUT
      CHARACTER  CH
!
      CALL UMACH (2, NOUT)
!                                Get ASCII value for the character
!                                'A'.
      CH = 'A'
      WRITE (NOUT,99999) CH, IACHAR(CH)
!
99999 FORMAT (' For the character  ', A1, '  the ASCII value is : ', &
             I3)
      END
```

Output

```
For the character  A  the ASCII value is :  65
```

Description

Routine IACHAR returns the ASCII value of the input character.

ICASE

This function returns the ASCII value of a character converted to uppercase.

Function Return Value

ICASE — Integer ASCII value for CH without regard to the case of CH. (Output)
Routine ICASE returns the same value as IACHAR (page 1459) for all but lowercase letters. For these, it returns the IACHAR value for the corresponding uppercase letter.

Required Arguments

CH — Character to be converted. (Input)

FORTRAN 90 Interface

Generic: ICASE(CH)

Specific: The specific interface name is ICASE.

FORTRAN 77 Interface

Single: ICASE(CH)

Example

This example shows the case insensitive conversion.

```
      USE ICASE_INT
      USE UMACH_INT
      INTEGER    NOUT
      CHARACTER  CHR
!                                  Get output unit number
      CALL UMACH (2, NOUT)
!                                  Get ASCII value for the character
!                                  'a'.
      CHR = 'a'
      WRITE (NOUT,99999) CHR, ICASE(CHR)
!
99999 FORMAT (' For the character  ', A1, '  the ICASE value is : ', &
             I3)
      END
```

Output

```
For the character  a  the ICASE value is :  65
```

Description

Routine ICASE converts a character to its integer ASCII value. The conversion is case insensitive; that is, it returns the ASCII value of the corresponding uppercase letter for a lowercase letter.

IICSR

This function compares two character strings using the ASCII collating sequence but without regard to case.

Function Return Value

IICSR — Comparison indicator. (Output)

Let USTR1 and USTR2 be the uppercase versions of STR1 and STR2, respectively. The following table indicates the relationship between USTR1 and USTR2 as determined by the ASCII collating sequence.

IICSR	Meaning
–1	USTR1 precedes USTR2
0	USTR1 equals USTR2
1	USTR1 follows USTR2

Required Arguments

STR1 — First character string. (Input)

STR2 — Second character string. (Input)

FORTRAN 90 Interface

Generic: `IICSR(STR1, STR2)`

Specific: The specific interface name is `IICSR`.

FORTRAN 77 Interface

Single: `IICSR(STR1, STR2)`

Example

This example shows different cases on comparing two strings.

```
      USE IICSR_INT
      USE UMACH_INT
      INTEGER    NOUT
      CHARACTER  STR1*6, STR2*6
!                                Get output unit number
      CALL UMACH (2, NOUT)
!                                Compare String1 and String2
!                                String1 is 'bigger' than String2
      STR1 = 'ABc 1'
```

```
      STR2 = ' '
      WRITE (NOUT,99999) STR1, STR2, IICSR(STR1,STR2)
!
!                                  String1 is 'equal' to String2
      STR1 = 'AbC'
      STR2 = 'ABc'
      WRITE (NOUT,99999) STR1, STR2, IICSR(STR1,STR2)
!
!                                  String1 is 'smaller' than String2
      STR1 = 'ABc'
      STR2 = 'aBC 1'
      WRITE (NOUT,99999) STR1, STR2, IICSR(STR1,STR2)
!
99999 FORMAT (' For String1 = ', A6, 'and String2 = ', A6, &
             ' IICSR = ', I2, /)
      END
```

Output

```
For String1 = ABc 1 and String2 =        IICSR =  1

For String1 = AbC   and String2 = ABc    IICSR =  0

For String1 = ABc   and String2 = aBC 1  IICSR = -1
```

Comments

If the two strings, STR1 and STR2, are of unequal length, the shorter string is considered as if it were extended with blanks to the length of the longer string.

Description

Routine IICSR compares two character strings. It returns –1 if the first string is less than the second string, 0 if they are equal, and 1 if the first string is greater than the second string. The comparison is case insensitive.

IIDEX

This funcion determines the position in a string at which a given character sequence begins without regard to case.

Function Return Value

IIDEX — Position in CHRSTR where KEY begins. (Output)
If KEY occurs more than once in CHRSTR, the starting position of the first occurrence is returned. If KEY does not occur in CHRSTR, then IIDEX returns a zero.

Required Arguments

CHRSTR — Character string to be searched. (Input)

KEY — Character string that contains the key sequence. (Input)

FORTRAN 90 Interface

Generic: `IIDEX(CHRSTR, KEY)`

Specific: The specific interface name is `S_IIDEX`.

FORTRAN 77 Interface

Single: `IIDEX(CHRSTR, KEY)`

Example

This example locates a key string.

```
      USE IIDEX_INT
      USE UMACH_INT
      INTEGER    NOUT
      CHARACTER  KEY*5, STRING*10
!                                  Get output unit number
      CALL UMACH (2, NOUT)
!                                  Locate KEY in STRING
      STRING = 'a1b2c3d4e5'
      KEY    = 'C3d4E'
      WRITE (NOUT,99999) STRING, KEY, IIDEX(STRING,KEY)
!
      KEY = 'F'
      WRITE (NOUT,99999) STRING, KEY, IIDEX(STRING,KEY)
!
99999 FORMAT (' For STRING = ', A10, ' and KEY = ', A5, ' IIDEX = ', I2, &
            /)
      END
```

Output

```
For STRING = a1b2c3d4e5 and KEY = C3d4E IIDEX =  5

For STRING = a1b2c3d4e5 and KEY = F     IIDEX =  0
```

Comments

If the length of `KEY` is greater than the length `CHRSTR`, `IIDEX` returns a zero.

Description

Routine `IIDEX` searches for a key string in a given string and returns the index of the starting element at which the key character string begins. It returns 0 if there is no match. The comparison is case insensitive. For a case-sensitive version, use the FORTRAN 77 intrinsic function `INDEX`.

CVTSI

Converts a character string containing an integer number into the corresponding integer form.

Required Arguments

STRING — Character string containing an integer number. (Input)

NUMBER — The integer equivalent of `STRING`. (Output)

FORTRAN 90 Interface

Generic: `CALL CVTSI (STRING, NUMBER)`

Specific: The specific interface name is `CVTSI`.

FORTRAN 77 Interface

Single: `CALL CVTSI (STRING, NUMBER)`

Example

The string "12345" is converted to an `INTEGER` variable.

```
      USE CVTSI_INT
      USE UMACH_INT
      INTEGER    NOUT, NUMBER
      CHARACTER  STRING*10
!
      DATA STRING/'12345'/
!
      CALL CVTSI (STRING, NUMBER)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'NUMBER = ', NUMBER
      END
```

Output

```
NUMBER =   12345
```

Description

Routine `CVTSI` converts a character string containing an integer to an `INTEGER` variable. Leading and trailing blanks in the string are ignored. If the string contains something other than an integer, a terminal error is issued. If the string contains an integer larger than can be represented by an `INTEGER` variable as determined from routine `IMACH` (see the Reference Material), a terminal error is issued.

CPSEC

This fuction returns CPU time used in seconds.

Function Return Value

CPSEC — CPU time used (in seconds) since first call to `CPSEC`. (Output)

Required Arguments

None

FORTRAN 90 Interface

Generic: `CPSEC ()`

Specific: The specific interface name is `CPSEC`.

FORTRAN 77 Interface

Single: `CPSEC ()`

Comments

1. The first call to `CPSEC` returns 0.0.

2. The accuracy of this routine depends on the hardware and the operating system. On some systems, identical runs can produce timings differing by more than 10 percent.

TIMDY

Gets time of day.

Required Arguments

IHOUR — Hour of the day. (Output)
`IHOUR` is between 0 and 23 inclusive.

MINUTE — Minute within the hour. (Output)
`MINUTE` is between 0 and 59 inclusive.

ISEC — Second within the minute. (Output)
`ISEC` is between 0 and 59 inclusive.

FORTRAN 90 Interface

Generic: `CALL TIMDY (IHOUR, MINUTE, ISEC)`

Specific: The specific interface name is `TIMDY`.

FORTRAN 77 Interface

Single: `CALL TIMDY (IHOUR, MINUTE, ISEC)`

Example

The following example uses `TIMDY` to return the current time. Obviously, the output is dependent upon the time at which the program is run.

```
      USE TIMDY_INT
      USE UMACH_INT
      INTEGER    IHOUR, IMIN, ISEC, NOUT
!
      CALL TIMDY (IHOUR, IMIN, ISEC)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'Hour:Minute:Second = ', IHOUR, ':', IMIN, &
                     ':', ISEC
      IF (IHOUR .EQ. 0) THEN
         WRITE (NOUT,*) 'The time is ', IMIN, ' minute(s), ', ISEC, &
                        ' second(s) past midnight.'
      ELSE IF (IHOUR .LT. 12) THEN
         WRITE (NOUT,*) 'The time is ', IMIN, ' minute(s), ', ISEC, &
                        ' second(s) past ', IHOUR, ' am.'
      ELSE IF (IHOUR .EQ. 12) THEN
         WRITE (NOUT,*) 'The time is ', IMIN, ' minute(s), ', ISEC, &
                        ' second(s) past noon.'
      ELSE
         WRITE (NOUT,*) 'The time is ', IMIN, ' minute(s), ', ISEC, &
                        ' second(s) past ', IHOUR-12, ' pm.'
      END IF
      END
```

Output

```
Hour:Minute:Second =   16:  52:  29
The time is   52 minute(s),   29 second(s) past   4 pm.
```

Description

Routine `TIMDY` is used to retrieve the time of day.

TDATE

Gets today's date.

Required Arguments

IDAY — Day of the month. (Output)
`IDAY` is between 1 and 31 inclusive.

MONTH — Month of the year. (Output)
`MONTH` is between 1 and 12 inclusive.

IYEAR — Year. (Output)
For example, IYEAR = 1985.

FORTRAN 90 Interface

Generic: `CALL TDATE (IDAY, MONTH, IYEAR)`

Specific: The specific interface name is `TDATE`.

FORTRAN 77 Interface

Single: `CALL TDATE (IDAY, MONTH, IYEAR)`

Example

The following example uses `TDATE` to return today's date.

```
      USE TDATE_INT
      USE UMACH_INT
      INTEGER    IDAY, IYEAR, MONTH, NOUT
!
      CALL TDATE (IDAY, MONTH, IYEAR)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'Day-Month-Year = ', IDAY, '-', MONTH, &
                     '-', IYEAR
      END
```

Output

```
Day-Month-Year =   2-  4-  1991
```

Description

Routine `TDATE` is used to retrieve today's date. Obviously, the output is dependent upon the date the program is run.

NDAYS

This function computes the number of days from January 1, 1900, to the given date.

Function Return Value

NDAYS — Function value. (Output)
If `NDAYS` is negative, it indicates the number of days prior to January 1, 1900.

Required Arguments

IDAY — Day of the input date. (Input)

MONTH — Month of the input date. (Input)

IYEAR — Year of the input date. (Input)
1950 would correspond to the year 1950 A.D. and 50 would correspond to year 50 A.D.

FORTRAN 90 Interface

Generic: `NDAYS(IDAY, MONTH, IYEAR)`

Specific: The specific interface name is `NDAYS`.

FORTRAN 77 Interface

Single: `NDAYS(IDAY, MONTH, IYEAR)`

Example

The following example uses `NDAYS` to compute the number of days from January 15, 1986, to February 28, 1986:

```
      USE NDAYS_INT
      USE UMACH_INT
      INTEGER    IDAY, IYEAR, MONTH, NDAY0, NDAY1, NOUT
!
      IDAY  = 15
      MONTH = 1
      IYEAR = 1986
      NDAY0 = NDAYS(IDAY,MONTH,IYEAR)
      IDAY  = 28
      MONTH = 2
      IYEAR = 1986
      NDAY1 = NDAYS(IDAY,MONTH,IYEAR)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'Number of days = ', NDAY1 - NDAY0
      END
```

Output

```
Number of days =   44
```

Comments

1. Informational error

Type	Code	
1	1	The Julian calendar, the first modern calendar, went into use in 45 B.C. No calendar prior to 45 B.C. was as universally used nor as accurate as the Julian. Therefore, it is assumed that the Julian calendar was in use prior to 45 B.C.

2. The number of days from one date to a second date can be computed by two references to `NDAYS` and then calculating the difference.

3. The beginning of the Gregorian calendar was the first day after October 4, 1582, which became October 15, 1582. Prior to that, the Julian calendar was in use. NDAYS makes the proper adjustment for the change in calendars.

Description

Function NDAYS returns the number of days from January 1, 1900, to the given date. The function NDAYSNDAYS returns negative values for days prior to January 1, 1900. A negative IYEAR can be used to specify B.C. Input dates in year 0 and for October 5, 1582, through October 14, 1582, inclusive, do not exist; consequently, in these cases, NDAYS issues a terminal error.

NDYIN

Gives the date corresponding to the number of days since January 1, 1900.

Required Arguments

NDAYS — Number of days since January 1, 1900. (Input)

IDAY — Day of the input date. (Output)

MONTH — Month of the input date. (Output)

IYEAR — Year of the input date. (Output)
1950 would correspond to the year 195 A.D. and −50 would correspond to year 50 B.C.

FORTRAN 90 Interface

Generic: `CALL NDYIN (NDAYS, IDAY, MONTH, IYEAR)`

Specific: The specific interface name is NDYIN.

FORTRAN 77 Interface

Single: `CALL NDYIN (NDAYS, IDAY, MONTH, IYEAR)`

Example

The following example uses NDYIN to compute the date for the 100th day of 1986. This is accomplished by first using NDAYS (page 1468) to get the "day number" for December 31, 1985.

```
      USE NDYIN_INT
      USE NDAYS_INT
      USE UMACH_INT
      INTEGER    IDAY, IYEAR, MONTH, NDAYO, NOUT
!
```

```
      NDAY0 = NDAYS(31,12,1985)
      CALL NDYIN (NDAY0+100, IDAY, MONTH, IYEAR)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'Day 100 of 1986 is (day-month-year) ', IDAY, &
                     '-', MONTH, '-', IYEAR
      END
```

Output

```
Day 100 of 1986 is (day-month-year)   10-  4-  1986
```

Comments

The beginning of the Gregorian calendar was the first day after October 4, 1582, which became October 15, 1582. Prior to that, the Julian calendar was in use. Routine NDYIN makes the proper adjustment for the change in calendars.

Description

Routine NDYIN computes the date corresponding to the number of days since January 1, 1900. For an input value of NDAYS that is negative, the date
computed is prior to January 1, 1900. The routine NDYIN is the inverse of NDAYS (page 1468).

IDYWK

This function computes the day of the week for a given date.

Function Return Value

IDYWK — Function value. (Output)
The value of IDYWK ranges from 1 to 7, where 1 corresponds to Sunday and 7 corresponds to Saturday.

Required Arguments

IDAY — Day of the input date. (Input)

MONTH — Month of the input date. (Input)

IYEAR — Year of the input date. (Input)
1950 would correspond to the year 1950 A.D. and 50 would correspond to year 50 A.D.

FORTRAN 90 Interface

Generic: `IDYWK(IDAY, MONTH, IYEAR)`

Specific: The specific interface name is IDYWK.

FORTRAN 77 Interface

Single: `IDYWK(IDAY, MONTH, IYEAR)`

Example

The following example uses `IDYWK` to return the day of the week for February 24, 1963.

```
      USE IDYWK_INT
      USE UMACH_INT
      INTEGER    IDAY, IYEAR, MONTH, NOUT
!
      IDAY  = 24
      MONTH = 2
      IYEAR = 1963
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'IDYWK (index for day of week) = ', &
                     IDYWK(IDAY,MONTH,IYEAR)
      END
```

Output

```
IDYWK (index for day of week) =   1
```

Comments

1. Informational error

Type	Code	
1	1	The Julian calendar, the first modern calendar, went into use in 45 B.C. No calendar prior to 45 B.C. was as universally used nor as accurate as the Julian. Therefore, it is assumed that the Julian calendar was in use prior to 45 B.C.

2. The beginning of the Gregorian calendar was the first day after October 4, 1582, which became October 15, 1582. Prior to that, the Julian calendar was in use. Function `IDYWK` makes the proper adjustment for the change in calendars.

Description

Function `IDYWK` returns an integer code that specifies the day of week for a given date. Sunday corresponds to 1, Monday corresponds to 2, and so forth.

A negative `IYEAR` can be used to specify B.C. Input dates in year 0 and for October 5, 1582, through October 14, 1582, inclusive, do not exist; consequently, in these cases, `IDYWK` issues a terminal error.

VERSL

This function obtains STAT/LIBRARY-related version, system and serial numbers.

Function Return Value

VERSL — CHARACTER string containing information. (Output)

Required Arguments

ISELCT — Option for the information to retrieve. (Input)

ISELCT	VERSL
1	IMSL STAT/LIBRARY version number
2	Operating system (and version number) for which the library was produced.
3	Fortran compiler (and version number) for which the library was produced.
4	IMSL STAT/LIBRARY serial number

FORTRAN 90 Interface

Generic: VERSL(ISELCT)

Specific: The specific interface name is S_VERSL.

FORTRAN 77 Interface

Single: VERSL(ISELCT)

Example

In this example, we print all of the information returned by VERSL on a particular machine. The output is omitted because the results are system dependent.

```
      USE UMACH_INT
      USE VERSL_INT
      INTEGER    ISELCT, NOUT
      CHARACTER  STRING(4)*50, TEMP*32
!
      STRING(1) = '('' IMSL STAT/LIBRARY Version Number:  '', A)'
      STRING(2) = '('' Operating System ID Number:  '', A)'
      STRING(3) = '('' Fortran Compiler Version Number:  '', A)'
      STRING(4) = '('' IMSL STAT/LIBRARY Serial Number:  '', A)'
!                                 Print the versions and numbers.
      CALL UMACH (2, NOUT)
      DO 10  ISELCT=1, 4
         TEMP = VERSL(ISELCT)
         WRITE (NOUT,STRING(ISELCT)) TEMP
   10 CONTINUE
      END
```

GDATA

Retrieves a commonly analyzed data set.

Required Arguments

IDATA — Data set indicator. (Input)

IDATA	NOBS	NVAR	Description of Data Set
1	16	7	Longley
2	176	2	Wolfer sunspot
3	150	5	Fisher iris
4	144	1	Box and Jenkins Series G
5	13	5	Draper and Smith Appendix B
6	197	1	Box and Jenkins Series A
7	296	2	Box and Jenkins Series J
8	100	4	Robinson Multichannel Time Series
9	113	34	Afifi and Azen Data Set A

Set IDATA = 0 to print a description of all the data sets above. In this case, the remaining arguments are not referenced.

X — NOBS by NVAR matrix containing the data set. (Output)

NOBS — Number of observations or rows in the output matrix. (Output)

NVAR — Number of variables or columns in the output matrix. (Output)

Optional Arguments

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Rows 1 through 10 of X are printed.
2	All rows of X are printed.

When printing is performed, a header listing the data set name and a reference is printed.

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

NDX — Second dimension of the matrix X exactly as specified in the dimension statement of the calling program. (Input)
Default: NDX = size (X,2).

FORTRAN 90 Interface

Generic: `CALL GDATA (IDATA, X, NOBS, NVAR,[,…])`

Specific: The specific interface names are `S_GDATA` and `D_GDATA`.

FORTRAN 77 Interface

Single: `CALL GDATA (IDATA, IPRINT, NOBS, NVAR, X, LDX, NDX)`

Double: The double precision name is `DGDATA`.

Example

`GDATA` is used to copy the Longley data set into the matrix `X`.

```
      USE GDATA_INT
      INTEGER    LDX, NDX
      PARAMETER  (LDX=200, NDX=10)
!
      INTEGER    IDATA, IPRINT, NOBS, NVAR
      REAL       X(LDX,NDX)
!
      IDATA  = 1
      IPRINT = 2
      CALL GDATA (IDATA, X, NOBS, NVAR, IPRINT=IPRINT)
!
      END
```

Output

```
The Longley data.

Longley, James W. (1967), An appraisal of least squares programs for the
electronic computer from the point of view of the user, Journal of the
American Statistical Association, 62, 819-841.

This data set consists of 16 observations on 7 variables.

                              X
          1          2          3          4          5          6
 1       83.0   234289.0     2356.0     1590.0   107608.0     1947.0
 2       88.5   259426.0     2325.0     1456.0   108632.0     1948.0
 3       88.2   258054.0     3682.0     1616.0   109773.0     1949.0
 4       89.5   284599.0     3351.0     1650.0   110929.0     1950.0
```

```
 5       96.2   328975.0     2099.0     3099.0   112075.0     1951.0
 6       98.1   346999.0     1932.0     3594.0   113270.0     1952.0
 7       99.0   365385.0     1870.0     3547.0   115094.0     1953.0
 8      100.0   363112.0     3578.0     3350.0   116219.0     1954.0
 9      101.2   397469.0     2904.0     3048.0   117388.0     1955.0
10      104.6   419180.0     2822.0     2857.0   118734.0     1956.0
11      108.4   442769.0     2936.0     2798.0   120445.0     1957.0
12      110.8   444546.0     4681.0     2637.0   121950.0     1958.0
13      112.6   482704.0     3813.0     2552.0   123366.0     1959.0
14      114.2   502601.0     3931.0     2514.0   125368.0     1960.0
15      115.7   518173.0     4806.0     2572.0   127852.0     1961.0
16      116.9   554894.0     4007.0     2827.0   130081.0     1962.0

            7
 1    60323.0
 2    61122.0
 3    60171.0
 4    61187.0
 5    63221.0
 6    63639.0
 7    64989.0
 8    63761.0
 9    66019.0
10    67857.0
11    68169.0
12    66513.0
13    68655.0
14    69564.0
15    69331.0
16    70551.0
```

Description

Routine GDATA retrieves a standard data set frequently cited in statistics textbooks or in this manual. The following table gives the references for each data set:

IDATA	Reference
1	Longley (1967)
2	Anderson (1971, page 660)
3	Fisher (1936); Mardia, Kent, and Bibby (1979, Table 1.2.2)
4	Box and Jenkins (1976, page 531)
5	Draper and Smith (1981, pages 629–630)
6	Box and Jenkins (1976, page 525)
7	Box and Jenkins (1976, page 532–533)
8	Robinson (1967, page 204)|
9	Afifi and Azen (1979, pages 16–22)

Chapter 20: Mathematical Support

Routines

GIRTS

Solves a triangular (possibly singular) set of linear systems and/or compute a generalized inverse of an upper triangular matrix.

Required Arguments

R — N by N upper triangular matrix. (Input)
If R contains a zero along the diagonal, the remaining elements of the row must also be zero. Only the upper triangle of R is referenced.

B — N by NB matrix containing the right hand sides of the linear system. (Input, if NB > 0)
If NB = 0, B is not referenced and can be a vector length one.

IRANK — Rank of R. (Output)

X — N by NB matrix containing the solution matrix corresponding to the right hand side B. (Output, if NB > 0)
If B is not needed, then X and B can share the same storage locations. If NB = 0, x is not referenced and can be a vector of length one.

Optional Arguments

N — Order of the upper triangular matrix R. (Input)
Default: N = size (R,2).

LDR — Leading dimension of R exactly as specified in the dimension statement of the calling program. (Input)
Default: LDR = size (R,1).

NB — Number of columns in B. (Input)
NB must be nonnegative. If NB is zero, no linear systems are solved.
Default: NB = size (B,2).

LDB — Leading dimension of B exactly as specified in the dimension statement of the calling program. (Input)
Default: LDB = size (B,1).

IPATH — Path option. (Input)
Default: IPATH = 1.

IPATH	Action
1	Solve R * X = B.
2	Solve R^T * X = B.
3	Solve R * X = B and compute RINV.
4	Solve R^T * X = B and compute RINV.

LDX — Leading dimension of X exactly as specified in the dimension statement of the calling program. (Input)
Default: LDX = size (X,1).

RINV — N by N upper triangular matrix that is the inverse of R when R is nonsingular. (Output, if IPATH equals 3 or 4)
(When R is singular, RINV is g_3 inverse. See the Algorithm section for an explanation of g_3 inverses.) If IPATH = 1 or 2, RINV is not referenced and can be a 1x1 array. If IPATH = 3 or 4 and R is not needed, then R and RINV can share the same storage locations.

LDRINV — Leading dimension of RINV exactly as specified in the dimension statement of the calling program. (Input)

FORTRAN 90 Interface

Generic: `CALL GIRTS (R, B, IRANK, X [,…])`

Specific: The specific interface names are `S_GIRTS` and `D_GIRTS`.

FORTRAN 77 Interface

Single: `CALL GIRTS (N, R, LDR, NB, B, LDB, IPATH, IRANK, X, LDX, RINV, LDRINV)`

Double: The double precision name is `DGIRTS`.

Example

The following example is taken from Maindonald (1984, pp. 102-105). A linear system $Rx = B$ is solved, and a g_3 inverse of R is computed.

```
      USE GIRTS_INT
      USE WRRRN_INT
      INTEGER    LDB, LDR, LDRINV, LDX, N, NB
      PARAMETER  (N=4, NB=1, LDB=N, LDR=N, LDRINV=N, LDX=N)
!
      INTEGER    IPATH, IRANK
      REAL       B(LDB,NB), R(LDR,N), RINV(LDRINV,N), X(LDX,NB)
!
      DATA (R(1,J),J=1,N)/6.0, 2.0, 5.0, 1.0/, B(1,1)/3.0/
      DATA (R(2,J),J=1,N)/0.0, 4.0,-2.0, 2.0/, B(2,1)/4.0/
      DATA (R(3,J),J=1,N)/0.0, 0.0, 0.0, 0.0/, B(3,1)/0.0/
      DATA (R(4,J),J=1,N)/0.0, 0.0, 0.0, 3.0/, B(4,1)/3.0/
!
      IPATH = 3
      CALL GIRTS (R, B, IRANK, X, IPATH=IPATH, RINV=RINV)
!
      CALL WRRRN ('RINV', RINV)
      CALL WRRRN ('X', X)
      END
```

Output

```
           RINV
        1       2       3       4
1   0.1667  -0.0833  0.0000  0.0000
2   0.0000   0.2500  0.0000 -0.1667
3   0.0000   0.0000  0.0000  0.0000
4   0.0000   0.0000  0.0000  0.3333

    X
1   0.167
2   0.500
3   0.000
4   1.000
```

Comments

1. Informational error
   Type Code

   3 1 The linear system of equations is inconsistent.

2. Routine GIRTS assumes that a singular R is represented by zero rows in R. No other forms of singularity in R are allowed.

Description

Routine GIRTS solves systems of linear algebraic equations with a triangular coefficient matrix. Inversion of the coefficient matrix is an option. The coefficient matrix can contain a zero diagonal element, but if so, the remaining elements in the row must be zero also. (A terminal error message is issued if a nonzero element appears in the row of the coefficient matrix where a zero diagonal element appears.)

If solution of a linear system is requested (i.e., NB > 0) and row i of the coefficient matrix contains elements all equal to zero, the following action is taken:

- The i-th row of the solution X is set to zero.
- If IPATH is 1 or 3, a warning error is issued when the i-th row of the right-hand side B is not zero.
- If IPATH is 2 or 4, a warning error is issued when the i-th row of the reduced right-hand side (obtained after the first $i - 1$ variables are eliminated from row i) is not zero within a computed tolerance.

If an inverse of the coefficient matrix is requested and row i contains elements all equal to zero, row i and column i elements of RINV are set to zero. The resulting inverse is a g_3 inverse of R. For a matrix G to be g_3 inverse of a matrix A, G must satisfy Conditions 1, 2, and 3 for the Moore-Penrose inverse but generally fail Condition 4. The four conditions for G to be a Moore-Penrose inverse of A are as follows:

1. $AGA = A$
2. $GAG = G$
3. AG is symmetric
4. GA is symmetric

For a detailed description of the algorithm, see Section 2 in Sallas and Lionti (1988).

CHFAC

Computes an upper triangular factorization of a real symmetric nonnegative definite matrix.

Required Arguments

A — N by N symmetric nonnegative definite matrix for which an upper triangular factorization is desired. (Input)
Only elements in the upper triangle of A are referenced.

IRANK — Rank of A. (Output)
N − IRANK is the number of effective zero pivots.

R — N by N upper triangular matrix containing the *R* matrix from a Cholesky decomposition $R^T R$ of A. (Output)
The elements of the appropriate rows of *R* are set to 0.0 if linear dependence of the columns of A is declared. (There are N − IRANK rows of *R* whose elements are set to 0.0.) If A is not needed, then R and A can share the same storage locations.

Optional Arguments

N — Order of the matrix. (Input)
Default: N = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

TOL — Tolerance used in determining linear dependence. (Input)
TOL = 100 * AMACH(4) is a common choice. See documentation for routine AMACH.
Default: TOL = 1.19 for single precision and 2.2d –14 for double precision.

LDR — Leading dimension of R exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL CHFAC (A, IRANK, R [,…])`

Specific: The specific interface names are S_CHFAC and D_CHFAC.

FORTRAN 77 Interface

Single: `CALL CHFAC (N, A, LDA, TOL, IRANK, R, LDR)`

Double: The double precision name is DCHFAC.

Example

A Cholesky factorization of a 5 × 5 symmetric nonnegative definite matrix is computed. Maindonald (1984, pages 85–86) discusses in detail the computations for this problem.

```
!                                  SPECIFICATIONS FOR PARAMETERS
      USE IMSL_LIBRARIES
      INTEGER    LDA, LDR, N
      PARAMETER  (N=5, LDA=N, LDR=N)
!
      INTEGER    IRANK, NOUT
      REAL       A(LDA,N), R(LDR,N), TOL
!
      DATA (A(1,J),J=1,N)/36.0, 12.0, 30.0,  6.0, 18.0/
      DATA (A(2,J),J=1,N)/12.0, 20.0,  2.0, 10.0, 22.0/
      DATA (A(3,J),J=1,N)/30.0,  2.0, 29.0,  1.0,  7.0/
      DATA (A(4,J),J=1,N)/ 6.0, 10.0,  1.0, 14.0, 20.0/
      DATA (A(5,J),J=1,N)/ 8.0, 22.0,  7.0, 20.0, 40.0/
!
      CALL CHFAC (A, IRANK, R)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'IRANK = ', IRANK
      CALL WRRRN ('R', R)
      END
```

Output

```
IRANK =   4

                    R
        1       2       3       4       5
1   6.000   2.000   5.000   1.000   3.000
2   0.000   4.000  -2.000   2.000   4.000
3   0.000   0.000   0.000   0.000   0.000
4   0.000   0.000   0.000   3.000   3.000
5   0.000   0.000   0.000   0.000   2.449
```

Comments

1. Informational error

Type	Code	
3	1	The input matrix is not nonnegative definite within the tolerance defined by TOL.

2. Elements of row i of R are set to 0.0 if a linear dependence is declared. Linear dependence is declared if

$$\left| a_{ii} - \sum_{j=1}^{i-1} r_{ji}^2 \right| \leq \text{TOL} * \left| a_{ii} \right|$$

Description

Routine CHFAC computes a Cholesky factorization $R^T R = A$ of an $n \times n$ symmetric nonnegative definite matrix A. The matrix R is taken to be an upper triangular matrix. The diagonal elements of R are taken to be nonnegative. If A is singular and has rank r, $n - r$ rows of R have all their elements taken to be zero.

The algorithm is based on the work of Healy (1968). The algorithm proceeds sequentially by columns. The i-th column is declared to be linearly dependent on the first $i - 1$ columns if

$$\left| a_{ii} - \sum_{j=1}^{i-1} r_{ji}^2 \right| \leq \varepsilon \left| a_{ii} \right|$$

where ε (stored in TOL) is the input tolerance. When a linear dependence is declared, all elements in the i-th row of R are set to zero.

Modifications due to Farebrother and Berry (1974) and Barrett and Healy (1978) for checking for matrices that are not nonnegative definite are also incorporated. Routine CHFAC declares A to not be nonnegative definite and issues an error message with an error code of 1 if either of the following conditions is satisfied:

1. $a_{ii} - \sum_{j=1}^{i-1} r_{ji}^2 < -\varepsilon \left| a_{ii} \right|$
2. $r_{ii} = 0$ and $\left| a_{ik} - \sum_{j=1}^{i-1} r_{ji} r_{jk} \right| > \varepsilon \sqrt{a_{ii} a_{kk}}$, $k > i$

Healy's (1968) algorithm and CHFAC permit the matrices A and R to occupy the same storage locations. Barrett and Healy (1978) in their remark neglect this fact. Routine CHFAC uses

$$\sum_{j=1}^{i-1} r_{ji}^2$$

for a_{ii} in the Condition 2 above to remedy this problem.

MCHOL

Computes an upper triangular factorization of a real symmetric matrix A plus a diagonal matrix D, where D is determined sequentially during the Cholesky factorization in order to make $A + D$ nonnegative definite.

Required Arguments

A — N by N symmetric matrix for which a Cholesky factorization is attempted. (Input)
Only elements in the upper triangle and diagonal of A are referenced.

IRANK — Rank of $A + D$. (Output)

R — N by N upper triangular matrix containing the R matrix from a Cholesky decomposition $R^T R$ of $A + D$. (Output)
The lower triangle of R is not referenced. If A is not needed, then R and A can share the same storage locations.

DMAX — Largest diagonal element of D. (Output)
If DMAX equals 0.0, then A is nonnegative definite, and R is a Cholesky factorization of A. If DMAX is positive, then A is indefinite, i.e., A has at least one negative eigenvalue. In this case, DMAX is an upper bound on the absolute value of the minimum eigenvalue.

IND — Index for subsequent computation of a descent direction in the case of a saddle point. (Output)
If IND = 0, then A is nonnegative definite. For positive IND, let e be a unit vector with IND-th element 1 and remaining elements 0. The solution s of $Rs = e$ is a direction of negative curvature, i.e., $s^T As$ is negative.

Optional Arguments

N — Order of the matrix. (Input)
Default: N = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

TOL — Tolerance used in determining linear dependence. (Input)
TOL = 100 * AMACH(4) is a common choice. See documentation for routine AMACH.
Default: TOL = 1.e-5 for single precision and 2.d –14 for double precision.

LDR — Leading dimension of R exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

FORTRAN 90 Interface

Generic: `CALL MCHOL (A, IRANK, R, DMAX, IND [,…])`

Specific: The specific interface names are S_MCHOL and D_MCHOL.

FORTRAN 77 Interface

Single: `CALL MCHOL (N, A, LDA, TOL, IRANK, R, LDR, DMAX, IND)`

Double: The double precision name is DMCHOL.

Example 1

A Cholesky factorization of a 5 × 5 symmetric nonnegative definite matrix is computed. Maindonald (1984, pages 85–86) discusses the example.

```
!                                  SPECIFICATIONS FOR PARAMETERS
```

```
      USE MCHOL_INT
      USE UMACH_INT
      USE WRRRN_INT

      INTEGER    LDA, LDR, N
      PARAMETER  (N=5, LDA=N, LDR=N)
!
      INTEGER    IND, IRANK, NOUT
      REAL       A(LDA,N), DMAX, R(LDR,N), TOL
!
      DATA (A(1,J),J=1,N)/36.0, 12.0, 30.0, 6.0, 18.0/
      DATA (A(2,J),J=1,N)/12.0, 20.0, 2.0, 10.0, 22.0/
      DATA (A(3,J),J=1,N)/30.0, 2.0, 29.0, 1.0, 7.0/
      DATA (A(4,J),J=1,N)/6.0, 10.0, 1.0, 14.0, 20.0/
      DATA (A(5,J),J=1,N)/8.0, 22.0, 7.0, 20.0, 40.0/
!
      TOL = 0.00001
      CALL MCHOL (A, IRANK, R, DMAX, IND, TOL=TOL)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99998) ' IRANK = ', IRANK
      WRITE (NOUT,99999) ' DMAX =  ', DMAX
      WRITE (NOUT,99998) ' IND =   ', IND
99998 FORMAT (A, I3)
99999 FORMAT (A, 1PE10.3)
      CALL WRRRN ('R', R)
      END
```

Output

```
IRANK =   4
DMAX =   0.000E+00
IND =     0

                    R
        1       2       3       4       5
1   6.000   2.000   5.000   1.000   3.000
2   0.000   4.000  -2.000   2.000   4.000
3   0.000   0.000   0.000   0.000   0.000
4   0.000   0.000   0.000   3.000   3.000
5   0.000   0.000   0.000   0.000   2.449
```

Description

Routine MCHOL computes a Cholesky factorization, $R^T R$, of $A + D$ where A is symmetric and D is a diagonal matrix with sufficiently large diagonal elements such that $A + D$ is nonnegative definite. The routine is similar to one described by Gill, Murray, and Wright (1981, pages 108–111). Here, though, we allow $A + D$ to be singular.

The algorithm proceeds sequentially by rows. If $A + D$ is singular, the Cholesky factor R is taken to have some rows that are entirely zero. The i-th row of $A + D$ is declared to be linearly dependent on the first $i - 1$ rows if the following two conditions are satisfied:

1. $\left|a_{ii} - \sum_{j=1}^{i-1} r_{ji}^2\right| \le \varepsilon |a_{ii}|$

2. $\left|a_{ik} - \sum_{j=1}^{i-1} r_{ji} r_{jk}\right| \le \varepsilon \sqrt{|a_{ii} a_{kk}|}, \; k > i$

where ε is the input argument TOL.

The routine MCHOL is often used to find a descent direction in a minimization problem. Let A and g be the current Hessian and gradient, respectively, associated with the minimization problem. The solution s of $As = -g$ may not give a descent direction if A is not nonnegative definite. Instead, in order to guarantee a descent direction, a solution s of $(A + D)s = -g$ can be found where $A + D$ is nonnegative definite. Routine MCHOL is useful for computing the upper triangular Cholesky factor R of $A + D$ so that routine GIRTS can be invoked to compute the descent direction s by solving successively the two triangular linear systems $R^T x = -g$ and $Rs = x$ for x and then s. Also if $g = 0$ and A is not nonnegative definite, i.e., the current solution is a saddle point, GIRTS can be used to compute a descent direction s from the linear system $Rs = e$ where e is a unit vector with

$$\varepsilon_i = \begin{cases} 1 & \text{if } i = \text{IND} \\ 0 & \text{otherwise} \end{cases}$$

Additional Example

Example 2

A modified Cholesky factorization of a 3 × 3 symmetric indefinite matrix A is computed. A solution of $Rs = e_3$ is also obtained using routine GIRTS. Note that $s^T As$ is negative as verified by using routine BLINF (IMSL MATH/LIBRARY). Gill, Murray, and Wright (1981, page 111) discuss the example.

```
!                                  SPECIFICATIONS FOR PARAMETERS
      USE IMSL_LIBRARIES

      INTEGER    LDA, LDR, N
      PARAMETER  (N=3, LDA=N, LDR=N)
!
      INTEGER    IND, IRANK, NOUT
      REAL       A(LDA,N), DMAX, E(N,1), R(LDR,N), S(N, 1), SPAS, TOL
!
      DATA (A(1,J),J=1,N)/1, 1, 2/
      DATA (A(2,J),J=1,N)/1, 1, 3/
      DATA (A(3,J),J=1,N)/2, 3, 1/
!
      TOL = 0.00001
      CALL MCHOL (A, IRANK, R, DMAX, IND, TOL=TOL)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99998) ' IRANK = ', IRANK
      WRITE (NOUT,99999) ' DMAX  = ', DMAX
      WRITE (NOUT,99998) ' IND   = ', IND
      CALL WRRRN ('R', R)
      IF (IND .GT. 0) THEN
         E= 0.0
```

```
         E(IND, 1) = 1.0
         CALL GIRTS (R, E, IRANK, S)
         SPAS = BLINF(A, S(:,1), S(:,1))
         WRITE (NOUT,*) ' '
         WRITE (NOUT,99999) ' trans(s)*A*s = ', SPAS
      END IF
99998 FORMAT (A, I3)
99999 FORMAT (A, F10.3)
      END
```

Output

```
IRANK =   3
DMAX  =      5.016
IND   =   3

            R
         1       2       3
1    1.942   0.515   1.030
2    0.000   2.398   1.030
3    0.000   0.000   1.059

trans(s)*A*s =     -2.254
```

ENOS

This function evaluates the expected value of a normal order statistic.

Function Return Value

ENOS — Function value, the expected value of the `I`-th order statistic in a sample of size `N` from the standard normal distribution. (Output)
See Comment 1.

Required Arguments

I — Rank of the order statistic. (Input)

N — Sample size. (Input)

FORTRAN 90 Interface

Generic: `ENOS(I, N)`

Specific: The specific interface names are `S_ENOS` and `D_ENOS`.

FORTRAN 77 Interface

Single: `ENOS(I, N)`

Double: The double precision name is DENOS.

Example

In this example, we compute the expected value of the first order statistic in a sample of size 5 from a standard normal distribution.

```
      USE UMACH_INT
      USE ENOS_INT

      INTEGER    I, N, NOUT
      REAL       EX
!
      CALL UMACH (2, NOUT)
      I  = 1
      N  = 5
      EX = ENOS(I,N)
      WRITE (NOUT,99999) EX
99999 FORMAT (' The expected value of the smallest order statistic', &
            /, ' in a normal sample of size 5 is ', F9.5)
      END
```

Output

```
The expected value of the smallest order statistic
in a normal sample of size 5 is  -1.16296
```

Comments

1. If the generic version of this function is used, the immediate result must be stored in a variable before use in an expression. For example:

   ```
   X = ENOS(I, N)
   Y = SQRT(X)
   ```

   must be used rather than

   ```
   Y = SQRT(ENOS(I, N))
   ```

   If this is too much of a restriction on the programmer, then the specific name can be used without this restriction.Informational errors

2. Informational errors

   Type Code

   3 1 The rank of the order statistic is less than 1. A rank of 1 is assumed.

   3 2 The rank of the order statistic is greater than sample size (N). A rank of N is assumed.

Description

Let $X_1 \le X_2 \le \ldots \le X_n$ be the order statistics of a random sample of size n from a standard normal distribution. The expected value of X_i is given by

$$\frac{n!}{(n-i)!(i-1)!}\int_{-\infty}^{\infty} x\left[\Phi(x)\right]^{i-1}\left[1-\Phi(x)\right]^{n-i}\phi(x)\,dx$$

where $\phi(x)$ and $\Phi(x)$ are the standard normal density and cumulative distribution functions respectively (David 1981).

Function `ENOS` evaluates the integral using a trapezoidal rule after first making a logarithmic transformation. This is the method used by Harter (1961). Although the method permits computations for any value of n, extremely large values of n cannot be guaranteed to be as accurate as smaller values of n. For $n > 2500$, the method is inappropriate.

AMILLR

This functon evaluates Mill's ratio (the ratio of the ordinate to the upper tail area of the standardized normal distribution).

Function Return Value

AMILLR — Function value, Mill's ratio. (Output)

Required Arguments

X — Value at which Mill's ratio is evaluated. (Input)
In order to avoid overflow, `X` must be less than a bound that is machine dependent. On most machines, the bound is greater than –13. The function underflows (and is set to 0.0) for small values of `X`. On most machines, the underflow does not occur unless `X` is less than –13.

FORTRAN 90 Interface

Generic: `AMILLR(X)`

Specific: The specific interface names are `S_AMILLR` and `D_AMILLR`.

FORTRAN 77 Interface

Single: `AMILLR(X)`

Double: The double precision name is `DMILLR`.

Example

In this example, we compute Mill's ratio at $x = -1.0$.

```
      USE UMACH_INT
      USE AMILLR_INT
      INTEGER    NOUT
      REAL       R, X
!
      CALL UMACH (2, NOUT)
      X = -1.0
      R = AMILLR(X)
      WRITE (NOUT,99999) R
99999 FORMAT (' Mill''s ratio at -1.0 is ', F8.5)
      END
```

Output

```
Mill's ratio at -1.0 is  0.28760
```

Comments

Informational error

Type	Code	
2	1	The function underflows because X is too small.

Description

Function AMILLR evaluates Mill's ratio, the *hazard rate* for the standard normal distribution. It is computed as the ratio of the ordinate to the upper tail area of the standard normal distribution, that is, $\phi(x)/(1 - \Phi(x))$, where $\phi(x)$ and $\Phi(x)$ are the standard normal density and cumulative distribution functions, respectively. The reciprocal of Mill's ratio is called the *failure rate* in reliability and life testing applications. As x becomes small, the ratio goes to zero. For large x (how large is machine dependent), the ratio cannot be computed. Function AMILLR computes $1 - \Phi(x)$ using the complementary error function (IMSL 1991) rather than as one minus the normal distribution function, which would underflow sooner as x gets small.

QUADT

Forms a *k-d* tree.

Required Arguments

X — NROW by NCOL matrix containing the data to be used on this call. (Input/Output)
On output the rows of X have been rearranged to form the *k-d* tree. X must not contain missing values (NaN).

IND — Vector of length NVAR containing the column numbers in X to be used in the forming the *k-d* tree. (Input)

NBUCK — Bucket size. (Input)
NBUCK gives the maximum number of observations in a leaf of the *k-d* tree. NBUCK = 3 is a common choice. NBUCK should be small when compared to NROW.

IDISCR — Vector of length NROW containing the element number in IND that points to the column of X to be used as the discriminator in the *k-d* tree. (Output)
IDISCR(I) = 0 if the observation is a terminal node. IND(IDISCR(I)) is the column number in X to be used as the discriminator.

PART — Vector of length NROW containing the value to be used in the partition for this observation. (Output)

Optional Arguments

NROW — Number of rows of X to be used in forming the *k-d* tree. (Input)
Default: NROW = size (X,1).

NVAR — Number of variables to be used in forming the tree. (Input)
Default: NVAR = size (IND,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

FORTRAN 90 Interface

Generic: `CALL QUADT (X, IND, NBUCK, IDISCR, PART [,…])`

Specific: The specific interface names are S_QUADT and D_QUADT.

FORTRAN 77 Interface

Single:
```
CALL QUADT (NROW, NVAR, NCOL, X, LDX, IND, NBUCK, IDISCR,
PART)
```

Double: The double precision name is DQUADT.

Example

The following example creates a *k-d* tree from financial data collected for firms approximately 2 years prior to bankruptcy and for financially sound firms at about the same point in time. The data on five variables, X_1 = (population), X_2 = (cash flow)/(total debt), X_3 = (net income)/(total assets), X_4 = (current assets)/(current liabilities), and X_5 = (current assets)/(net sales) are taken from Johnson and Wichern (1988, page 536).

```
      USE QUADT_INT
      USE WRRRN_INT
      USE WRIRN_INT
      INTEGER    LDX, NBUCK, NCOL, NROW, NVAR
      PARAMETER  (LDX=47, NBUCK=3, NCOL=5, NROW=47, NVAR=4)
!
      INTEGER    IDISCR(NROW), IND(NVAR)
      REAL       PART(NROW), X(LDX,NCOL)
!
      DATA IND/2, 3, 4, 5/
      DATA (X(I,1),I=1,47)/1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., &
          1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2., 2., 2., 2., &
          2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., &
          2., 2., 2., 2., 2., 2./
      DATA (X(I,2),I=1,47)/-0.4485, -0.5633, 0.0643, -0.0721, -0.1002, &
          -0.1421, 0.0351, -0.0653, 0.0724, -0.1353, -0.2298, 0.0713, &
          0.0109, -0.2777, 0.1454, 0.3703, -0.0757, 0.0451, 0.0115, &
          0.1227, -0.2843, 0.5135, 0.0769, 0.3776, 0.1933, 0.3248, &
          0.3132, 0.1184, -0.0173, 0.2169, 0.1703, 0.1460, -0.0985, &
          0.1398, 0.1379, 0.1486, 0.1633, 0.2907, 0.5383, -0.3330, &
          0.4785, 0.5603, 0.2029, 0.2029, 0.4746, 0.1661, 0.5808/
      DATA (X(I,3),I=1,47)/-0.4106, -0.3114, -0.3114, -0.0930, &
          -0.0917, -0.0651, 0.0147, -0.0566, -0.0076, -0.1433, &
          -0.2961, 0.0205, 0.0011, -0.2316, 0.0500, 0.1098, -0.0821, &
          0.0263, -0.0032, 0.1055, -0.2703, 0.1001, 0.0195, 0.1075, &
          0.0473, 0.0718, 0.0511, 0.0499, 0.0233, 0.0779, 0.0695, &
          0.0518, -0.0123, -0.0312, 0.0728, 0.0564, 0.0486, 0.0597, &
          0.1064, -0.0854, 0.0910, 0.1112, 0.0792, 0.0792, 0.1380, &
          0.0351, 0.0371/
      DATA (X(I,4),I=1,47)/1.0865, 1.5134, 1.0077, 1.4544, 1.5644, &
          0.7066, 1.5046, 1.3737, 1.3723, 1.4196, 0.3310, 1.3124, &
          2.1495, 1.1918, 1.8762, 1.9941, 1.5077, 1.6756, 1.2602, &
          1.1434, 1.2722, 2.4871, 2.0069, 3.2651, 2.2506, 4.2401, &
          4.4500, 2.5210, 2.0538, 2.3489, 1.7973, 2.1692, 2.5029, &
          0.4611, 2.6123, 2.2347, 2.3080, 1.8381, 2.3293, 3.0124, &
          1.2444, 4.2918, 1.9936, 1.9936, 2.9166, 2.4527, 5.0594/
      DATA (X(I,5),I=1,47)/0.4526, 0.1642, 0.3978, 0.2589, 0.6683, &
          0.2794, 0.7080, 0.4032, 0.3361, 0.4347, 0.1824, 0.2497, &
          0.6969, 0.6601, 0.2723, 0.3828, 0.4215, 0.9494, 0.6038, &
          0.1655, 0.5128, 0.5368, 0.5304, 0.3548, 0.3309, 0.6279, &
          0.6852, 0.6925, 0.3483, 0.3970, 0.5174, 0.5500, 0.5778, &
          0.2643, 0.5151, 0.5563, 0.1978, 0.3786, 0.4835, 0.4730, &
          0.1847, 0.4443, 0.3018, 0.3018, 0.4487, 0.1370, 0.1268/
!
      CALL QUADT (X, IND, NBUCK, IDISCR, PART)
      CALL WRRRN ('first 10 rows of X after QUADT', X, 10, NCOL, LDX)
      CALL WRRRN ('PART', PART, 1, NROW, 1)
      CALL WRIRN ('IDISCR', IDISCR, 1, NROW, 1)
!
      END
```

Output

```
 first 10 rows of X after QUADT
    1       2       3       4       5
```

```
 1   1.000  -0.230  -0.296   0.331   0.182
 2   2.000   0.140  -0.031   0.461   0.264
 3   1.000  -0.142  -0.065   0.707   0.279
 4   1.000  -0.449  -0.411   1.087   0.453
 5   1.000   0.064  -0.311   1.008   0.398
 6   1.000   0.123   0.105   1.143   0.166
 7   1.000  -0.284  -0.270   1.272   0.513
 8   1.000  -0.278  -0.232   1.192   0.660
 9   1.000   0.012  -0.003   1.260   0.604
10   1.000   0.071   0.021   1.312   0.250

                                    PART
    1      2      3      4      5      6      7      8      9      10
0.000  0.461  0.857  0.000  0.064  1.168  0.000 -0.278  0.041  0.000

   11     12     13     14     15     16     17     18     19     20
0.072  1.373  0.000 -0.072  0.412  0.000  0.435 -0.015  0.000  1.876

   21     22     23     24     25     26     27     28     29     30
0.448  0.000   .708  1.994  0.000  0.203  2.152  0.000  2.308  0.390

   31     32     33     34     35     36     37     38     39     40
0.000   .550  0.147  0.000  0.217  2.453  0.000  2.521  0.128  0.000

   41      42      43      44      45      46      47
2.612   3.012   0.000   4.240   4.292   4.755   0.000

                                  IDISCR
 1  2  3  4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
 0  3  3  0   1   3   0   1   1   0   1   3   0   1   4   0   4   1   0   3

21 22 23 24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
 4  0  4  3   0   1   3   0   3   4   0   4   1   0   1   3   0   3   1   0

41  42  43  44  45  46  47
 3   3   0   3   3   3   0
```

Comments

Workspace may be explicitly provided, if desired, by use of Q2ADT/DQ2ADT. The reference is:

```
CALL Q2ADT (NROW, NVAR, NCOL, X, LDX, IND, NBUCK, IDISCR, PART, ILOW,
IHIGH, WK, IWK)
```

The additional arguments are as follows:

ILOW — Work vector of length $\log_2$(NROW) + 3.

IHIGH — Work vector of length $\log_2$(NROW) + 3.

WK — Work vector of length NROW.

IWK — Work vector of length NROW.

Description

Routine QUADT creates the data structure required for a *k-d* tree. A *k-d* tree is a multivariate form of B-tree that is especially useful for finding nearest neighbors but may be of use in other situations. Once the *k-d* tree has been formed, routine NGHBR (page 1494) may be used to find the nearest neighbors for any point in logarithmic time.

The basic algorithm is given by Friedman, Bentley, and Finkel (1977) and can be summarized as follows:

1. Let $l = 1$ and $h =$ NROW.
2. Let $k = (l + h)/2$.
3. Each column in X to be used in forming the *k-d* tree is examined over the range $[l, h]$ in order to find the column with the maximum spread. Let j equal this column number.
4. The k-th element of PART is set to the median value in the range $[l, h]$ of the j-th column in X while IDISCR(k) is set to the element in IND that points to this column.
5. The rows of X are interchanged so that all rows of X with values in column j less than or equal to the median value computed in Step 4 occur before (or at) the k-th element.
6. Go to Step 2 repeatedly with zero, one, or two submatrices in X. Go to Step 2 with the submatrix formed from rows l to k of X if $k - l$ is greater than NBUCK. Go to Step 2 with the submatrix formed from rows $k + 1$ to h of X if $h - k - 1$ is greater than NBUCK.

The bucket size, NBUCK, is the maximum number of observations allowed in the lowest level of the *k-d* tree, i.e., in the leaves of the tree. The choice of NBUCK can affect the speed with which nearest neighbors are found. A value of 3 or 5 is a common choice, but if the number of nearest neighbors to be obtained is large, a larger value for NBUCK should probably be used.

NGHBR

Search's a *k-d* tree for the *k* nearest neighbors of a key.

Required Arguments

XKEY — Vector of length NVAR containing the key for which nearest neighbors are desired. (Input)
Note that the elements in XKEY are not arranged in the same manner as the columns in X.

K — Number of nearest neighbors to find. (Input)

X — NROW by NCOL matrix containing the data used to form the *k-d* tree as output from routine QUADT (page 1490). (Input)
X must not contain missing values (NaN).

IND — Vector of length NVAR containing the column numbers in X used in forming the *k-d* tree. (Input)

NBUCK — Bucket size. (Input)
NBUCK is the maximum number of observations in a leaf of the *k-d* tree. The value of NBUCK should be the same as the value used in forming the *k-d* tree (i.e. as input to the routine QUADT).

IDISCR — Vector of length NROW containing the element number in IND that points to the column of X to be used as the discriminator in the *k-d* tree, as output from routine QUADT. (Input)
IDISCR(I) = 0 if the observation is a terminal node. IND(IDISCR(I)) is the column number in X to be used as the discriminator.

PART — Vector of length NROW containing the median value to be used for the partition, as output from routine QUADT. (Input)

IPQR — Vector of length K containing the indices of the nearest neighbors. (Output)

PQD — Vector of length K containing the nearest neighbor distances. (Output)

Optional Arguments

NVAR — Number of variables used to form the *k-d* tree. (Input)
Default: NVAR = size (XKEY,1).

NROW — Number of rows of X used to form the *k-d* tree. (Input)
Default: NROW = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

METRIC — Metric to use in computing the *k* nearest neighbors. (Input)
Default: METRIC = 0.

METRIC	Metric used
0	Euclidean distance
1	L_1 norm
2	L_∞ norm

FORTRAN 90 Interface

Generic: `CALL NGHBR (XKEY, K, X, IND, NBUCK, IDISCR, PART, IPQR, PQD [,…])`

Specific: The specific interface names are `S_NGHBR` and `D_NGHBR`.

FORTRAN 77 Interface

Single: `CALL NGHBR (NVAR, XKEY, K, NROW, NCOL, X, LDX, IND, NBUCK, IDISCR, PART, METRIC, IPQR)`

Double: The double precision name is `DNGHBR`.

Example

The following example creates a *k-d* tree from financial data collected for firms approximately 2 years prior to bankruptcy and for financially sound firms at about the same point in time. The data on five variables, X_1 = (population), X_2 = (cash flow)/(total dept), X_3 = (net income)/(total assets), X_4 = (current assets)/(current liabilities), and X_5 = (current assets)/(net sales) are taken from Johnson and Wichern, page 536. Routine `NGHBR` is then used to determine the 5 nearest neighbors of the first row in `X`. As expected, one of the nearest neighbors found is the key (the first row in `X`).

```
      USE QUADT_INT
      USE NGHBR_INT
      USE WRIRN_INT
      USE WRRRN_INT
      INTEGER    K, LDX, METRIC, NBUCK, NCOL, NROW, NVAR
      PARAMETER  (K=5, LDX=47, METRIC=1, NBUCK=3, NCOL=5, NROW=47, &
                 NVAR=4)
!
      INTEGER    I, IDISCR(NROW), IND(NVAR), IPQR(K)
      REAL       PART(NROW), PQD(K), X(LDX,NCOL), XKEY(NVAR)
!
      DATA IND/2, 3, 4, 5/
      DATA (X(I,1),I=1,47)/1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., &
          1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2., 2., 2., 2., &
          2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., &
          2., 2., 2., 2., 2., 2./
      DATA (X(I,2),I=1,47)/-0.4485, -0.5633, 0.0643, -0.0721, -0.1002, &
          -0.1421, 0.0351, -0.0653, 0.0724, -0.1353, -0.2298, 0.0713, &
          0.0109, -0.2777, 0.1454, 0.3703, -0.0757, 0.0451, 0.0115, &
          0.1227, -0.2843, 0.5135, 0.0769, 0.3776, 0.1933, 0.3248, &
          0.3132, 0.1184, -0.0173, 0.2169, 0.1703, 0.1460, -0.0985, &
          0.1398, 0.1379, 0.1486, 0.1633, 0.2907, 0.5383, -0.3330, &
          0.4785, 0.5603, 0.2029, 0.2029, 0.4746, 0.1661, 0.5808/
      DATA (X(I,3),I=1,47)/-0.4106, -0.3114, -0.3114, -0.0930, &
          -0.0917, -0.0651, 0.0147, -0.0566, -0.0076, -0.1433, &
          -0.2961, 0.0205, 0.0011, -0.2316, 0.0500, 0.1098, -0.0821, &
          0.0263, -0.0032, 0.1055, -0.2703, 0.1001, 0.0195, 0.1075, &
          0.0473, 0.0718, 0.0511, 0.0499, 0.0233, 0.0779, 0.0695, &
```

```
          0.0518, -0.0123, -0.0312, 0.0728, 0.0564, 0.0486, 0.0597, &
          0.1064, -0.0854, 0.0910, 0.1112, 0.0792, 0.0792, 0.1380, &
          0.0351, 0.0371/
      DATA (X(I,4),I=1,47)/1.0865, 1.5134, 1.0077, 1.4544, 1.5644, &
          0.7066, 1.5046, 1.3737, 1.3723, 1.4196, 0.3310, 1.3124, &
          2.1495, 1.1918, 1.8762, 1.9941, 1.5077, 1.6756, 1.2602, &
          1.1434, 1.2722, 2.4871, 2.0069, 3.2651, 2.2506, 4.2401, &
          4.4500, 2.5210, 2.0538, 2.3489, 1.7973, 2.1692, 2.5029, &
          0.4611, 2.6123, 2.2347, 2.3080, 1.8381, 2.3293, 3.0124, &
          1.2444, 4.2918, 1.9936, 1.9936, 2.9166, 2.4527, 5.0594/
      DATA (X(I,5),I=1,47)/0.4526, 0.1642, 0.3978, 0.2589, 0.6683,&
          0.2794, 0.7080, 0.4032, 0.3361, 0.4347, 0.1824, 0.2497, &
          0.6969, 0.6601, 0.2723, 0.3828, 0.4215, 0.9494, 0.6038, &
          0.1655, 0.5128, 0.5368, 0.5304, 0.3548, 0.3309, 0.6279, &
          0.6852, 0.6925, 0.3483, 0.3970, 0.5174, 0.5500, 0.5778, &
          0.2643, 0.5151, 0.5563, 0.1978, 0.3786, 0.4835, 0.4730, &
          0.1847, 0.4443, 0.3018, 0.3018, 0.4487, 0.1370, 0.1268/
!
!                                 Create the k-d tree
!
      CALL QUADT (X, IND, NBUCK, IDISCR, PART)
!
      DO 10  I=1, NVAR
         XKEY(I) = X(1,IND(I))
   10 CONTINUE
!
      CALL NGHBR (XKEY, K, X, IND, NBUCK, IDISCR, PART, IPQR, PQD, &
                  METRIC=METRIC)
!
      CALL WRIRN ('Indices of the nearest neighbors, IPQR.', IPQR, 1, K, 1)
      CALL WRRRN ('Nearest neighbor distances, PQD.', PQD, 1, K, 1)
!
      END
```

Output

```
Indices of the nearest neighbors, IPQR.
          1   2   3   4   5
          1   3   2   5   7

  Nearest neighbor distances, PQD.
    1       2       3       4       5
0.000   0.791   0.847   1.201   1.352
```

Comments

1. Workspace may be explicitly provided, if desired, by use of N2HBR/DN2HBR. The reference is:

   ```
   CALL N2HBR (NVAR, XKEY, K, NROW, NCOL, X, LDX, IND, IDISCR,
   PART, METRIC, IPQR, PQD, ILOW,   IHIGH, ISIDE, BNDL, BNDH)
   ```

   The additional arguments are as follows:

ILOW — Work vector of length $\log_2(\texttt{NROW}) + 3$.

IHIGH — Work vector of length $\log_2(\texttt{NROW}) + 3$.

ISIDE — Work vector of length $\log_2(\texttt{NROW}) + 3$.

BNDL — Work vector of length NVAR * ($\log_2(\texttt{NROW}) + 3$).

BNDH — Work vector of length NVAR * ($\log_2(\texttt{NROW}) + 3$).

2. Informational error

Type	Code	
4	1	The data structure input is not a *k-d* tree. Use routine QUADT to create the *k-d* tree.

Description

Routine NGHBR finds the *k* nearest neighbors in an input *k-d* tree for an arbitrary key, XKEY in logarithmic time. A *k-d* tree is a form of B-tree that is especially useful for finding nearest neighbors. The *k-d* tree input into routine NGHBR should be produced by routine QUADT (page 1490). Three metrics, Euclidean, L_1, and L_∞, are available for defining the nearest neighbors. The user should note that if the input key is a row of the *k-d* tree, then the row will be returned as one of the nearest neighbors. In this case, only $k - 1$ nearest neighbors will be found.

The algorithm is given by Friedman, Bentley, and Finkel (1977) and is summarized in the following. The basic idea is to traverse the *k-d* tree in order to determine which leaves of the tree need to be examined for the nearest neighbor. The algorithm is efficient because most leaves are not examined.

1. Let $l = 1$ and $h =$ NROW.

2. Let $k = (l + h)/2$, and j and p be the k-th elements of IDISCR and PART, respectively.

3. If $(h - l)$ is less than NBUCK, then go to Step 4. Otherwise, let m be the j-th element of IND. If the (k, m)-th element of X is greater than p, then let $l = k + 1$ and go to Step 2. Otherwise, set $h = k$ and go to Step 2.

4. Examine each row in X from row l to row h to determine if it is a nearest neighbor. Check to see if rows in X (leaves of the tree) adjacent to these rows need to be examined (see Friedman, Bentley, and Finkel (1977)). If necessary, examine the adjacent rows for nearest neighbors.

The value used for the bucket size, NBUCK, must be the same value as was used in routine QUADT when the *k-d* tree was created. A common choice for NBUCK is three.

Reference Material

Contents

User Errors

IMSL routines attempt to detect user errors and handle them in a way that provides as much information to the user as possible. To do this, we recognize various levels of severity of errors, and we also consider the extent of the error in the context of the purpose of the routine; a trivial error in one situation may be serious in another. IMSL routines attempt to report as many errors as they can reasonably detect. Multiple errors present a difficult problem in error detection because input is interpreted in an uncertain context after the first error is detected.

What Determines Error Severity

In some cases, the user's input may be mathematically correct, but because of limitations of the computer arithmetic and of the algorithm used, it is not possible to compute an answer accurately. In this case, the assessed degree of accuracy determines the severity of the error. In cases where the routine computes several output quantities, if some are not computable but most are, an error condition exists. The severity depends on an assessment of the overall impact of the error.

Terminal errors

If the user's input is regarded as meaningless, such as $N = -1$ when "N" is the number of equations, the routine prints a message giving the value of the erroneous input argument(s) and the reason for the erroneous input. The routine will then cause the user's program to stop. An error in which the user's input is meaningless is the most severe error and is called a *terminal error*. Multiple terminal error messages may be printed from a single routine.

Informational errors

In many cases, the best way to respond to an error condition is simply to correct the input and rerun the program. In other cases, the user may want to take actions in the program itself based on errors that occur. An error that may be used as the basis for corrective action within the program is called an *informational error*. If an informational error occurs, a user-retrievable code is set. A routine can return at most one informational error for a single reference to the routine. The codes for the informational error codes are printed in the error messages.

Other errors

In addition to informational errors, IMSL routines issue error messages for which no user-retrievable code is set. Multiple error messages for this kind of error may be printed. These errors, which generally are not described in the documentation, include terminal errors as well as less serious errors. Corrective action within the calling program is not possible for these errors.

Kinds of Errors and Default Actions

Five levels of severity of errors are defined in the STAT/LIBRARY. Each level has an associated PRINT attribute and a STOP attribute. These attributes have default settings (YES or NO), but they may also be set by the user. The purpose of having multiple error severity levels is to provide independent control of actions to be taken for errors of different severity. Upon return from an IMSL routine, exactly one error state exists. (A code 0 "error" is no informational error.) Even if more than one informational error occurs, only one message is printed (if the PRINT attribute is YES). Multiple errors for which no corrective action within the calling program is reasonable or necessary result in the printing of multiple messages (if the PRINT attribute for their severity level is YES). Errors of any of the severity levels except level 5 may be informational errors.

Level 1: **Note**. A *note* is issued to indicate the possibility of a trivial error or simply to provide information about the computations. Default attributes: PRINT = NO, STOP = NO

Level 2: **Alert**. An *alert* indicates that the user should be advised about events occurring in the software. Default attributes: PRINT = NO, STOP = NO

Level 3: **Warning**. A *warning* indicates the existence of a condition that may require corrective action by the user or calling routine. A warning error may be issued because the results are accurate to only a few decimal places, because some of the output may be erroneous but most of the output is correct, or because some assumptions underlying the analysis technique are violated. Often no corrective action is necessary and the condition can be ignored. Default attributes: PRINT = YES, STOP = NO

Level 4: **Fatal**. A *fatal* error indicates the existence of a condition that may be serious. In most cases, the user or calling routine must take corrective action to recover. Default attributes: PRINT = YES, STOP = YES

Level 5: **Terminal**. A *terminal* error is serious. It usually is the result of an incorrect specification, such as specifying a negative number as the number of equations. These errors may also be caused by various programming errors impossible to diagnose correctly in FORTRAN. The resulting error message may be perplexing to the user. In

such cases, the user is advised to compare carefully the actual arguments passed to the routine with the dummy argument descriptions given in the documentation. Special attention should be given to checking argument order and data types.

A terminal error is not an informational error because corrective action within the program is generally not reasonable. In normal usage, execution is terminated immediately when a terminal error occurs. Messages relating to more than one terminal error are printed if they occur. Default attributes: PRINT = YES, STOP = YES

The user can set PRINT and STOP attributes by calling ERSET as described in "Routines for Error Handling."

Errors in Lower-Level Routines

It is possible that a user's program may call an IMSL routine that in turn calls a nested sequence of lower-level IMSL routines. If an error occurs at a lower level in such a nest of routines and if the lower-level routine cannot pass the information up to the original user- called routine, then a traceback of the routines is produced. The only common situation in which this can occur is when an IMSL routine calls a user-supplied routine that in turn calls another IMSL routine.

Routines for Error Handling

There are three ways in which the user may interact with the IMSL error handling system: (1) to change the default actions, (2) to retrieve the integer code of an informational error so as to take corrective action, and (3) to determine the severity level of an error. The routines to use are ERSET, IERCD, and N1RTY, respectively.

ERSET

Change the default printing or stopping actions when errors of a particular error severity level occur.

Required Arguments

IERSVR — Error severity level indicator. (Input)
If IERSVR = 0, actions are set for levels 1 to 5. If IERSVR is 1 to 5, actions are set for errors of the specified severity level.

IPACT — Printing action. (Input)

IPACT	Action
–1	Do not change current setting(s).
0	Do not print.
1	Print.
2	Restore the default setting(s).

ISACT — Stopping action. (Input)

ISACT	Action
–1	Do not change current setting(s).
0	Do not stop.
1	Stop.
2	Restore the default setting(s).

FORTRAN 90 Interface

Generic: `CALL ERSET (IERSVR, IPACT, ISACT)`

Specific: The specific interface name is `ERSET`.

FORTRAN 77 Interface

Single: `CALL ERSET(IERSVR, IPACT, ISACT)`

IERCD and N1RTY

The last two routines for interacting with the error handling system, `IERCD` and `N1RTY`, are `INTEGER` functions and are described in the following material.

`IERCD` retrieves the integer code for an informational error. Since it has no arguments, it may be used in the following way:

```
ICODE = IERCD( )
```

The function retrieves the code set by the most recently called IMSL routine.

`N1RTY` retrieves the error type set by the most recently called IMSL routine. It is used in the following way:

```
ITYPE = N1RTY(1)
```

`ITYPE` = 1, 2, 4, and 5 correspond to error severity levels 1, 2, 4, and 5, respectively. `ITYPE` = 3 and `ITYPE` = 6 are both warning errors, error severity level 3. While `ITYPE` = 3 errors are informational errors (`IERCD`( ) ≠ 0), `ITYPE` = 6 errors are not informational errors (`IERCD`( ) = 0).

For software developers requiring additional interaction with the IMSL error handling system, see Aird and Howell (1991).

Examples

Changes to default actions

Some possible changes to the default actions are illustrated below. The default actions remain in effect for the kinds of errors not included in the call to `ERSET`.

To turn off printing of warning error messages:

```
CALL ERSET (3, 0, –1)
```

To stop if warning errors occur:

```
CALL ERSET (3, –1, 1)
```

To print all error messages:

```
CALL ERSET (0, 1, –1)
```

To restore all default settings:

```
CALL ERSET (0, 2, 2)
```

Use of informational error to determine program action

In the program segment below, the Cholesky factorization of a matrix is to be performed. If it is determined that the matrix is not nonnegative definite (and often this is not immediately obvious), the program is to take a different branch.

```
                                    .
                                    .
                                    .
      CALL CHFAC (A, IRANK, R)
      IF (IERCD() .EQ. 1) THEN
!            Handle matrix that is not nonnegative definite
                                    .
                                    .
                                    .
      END IF
```

Examples of All Types of Errors

The program below illustrates each of the different types of errors detected by the STAT/LIBRARY routines. If the call to ERSET was not made, messages for errors of levels 1 and 2 would not be printed.

The error messages refer to the argument names that are used in the documentation for the routine, rather than the user's name of the variable used for the argument. In the messages generated by IMSL routine CHFAC in this example, references are made to LDA and LDR, whereas in the program literals were used for these arguments. Note that error codes are printed as part of the messages for informational errors.

```
      USE IMSL_LIBRARIES
!                                   Specifications for local variables
      INTEGER   IDO, IOPT, IRANK, N, NMISS, NOBS, NPOP, NROW, NUM &
      REAL      A(2,2), CHSQ, CONPER, DF, PR, R(2,2), RCOEF, STAT(20),
                SUMRY(11), TOL, X(10), XMEAN, Y(10)
!
      DATA X/-5.0, -4.0, -3.0, -2.0, -1.0, 1.0, 2.0, 3.0, 4.0, 5.0/
      DATA Y/3.0, 5.0, 4.0, 5.0, 6.0, 7.0, 6.0, 8.0, 7.0, 9.0/
      DATA A/2.0, 0.0, 0.0, -3.0/
!                                    Turn on printing and turn off
!                                    stopping for all error types.
      CALL ERSET (0, 1, 0)
!                                    Generate level 1 informational error.
      DF = 1000.0
      CHSQ = -1.0
```

```
      PR = CHIDF(CHSQ,DF)
!                                     Generate level 2 informational error.
      DF = 1000.0
      CHSQ = 10.0
      PR = CHIDF(CHSQ,DF)
!                                     Generate level 3 informational error.
      NUM = 11
      CALL LETTR (X, SUMRY, NMISS, NUM=NUM)
!                                     Generate level 4 informational error.
      N = 2
      TOL = 0.0001
      CALL CHFAC (A, IRANK, R, TOL=TOL)
!                                     Generate several level 5 errors.
      CALL CHFAC (A, IRANK, R, TOL=TOL, LDR = -2)
!                                     Generate several warning errors that
!                                     do not allow corrective action
!                                     (because no codes are listed for
!                                     these errors in the document for the
!                                     routine).
      NROW = 10
      NPOP = 100
      IOPT = 1
      CONPER = 0.95
      CALL SMPRR (NROW, X, Y, NPOP, XMEAN, STAT, IOPT=IOPT, CONPER=CONPER)
      END
```

Output

```
*** NOTE      ERROR 1 from CHIDF. Since CHSQ = -1.000000E+00 is less than
***           zero, the distribution function is zero at CHSQ.
*** ALERT     ERROR 3 from CHIDF. The normal distribution is used for large
***           degrees of freedom. However, it has produced underflow.
***           Therefore, the probability is set to 0.
*** WARNING   ERROR 3 from LETTR. NUM = 11 and the number of observations =
***           10. Since NUM is greater than the number of observations, it
***           is likely that the results are not useful.
*** WARNING   ERROR 1 from CHFAC. The leading 2 by 2 submatrix of the input
***           matrix is not nonnegative definite within the tolerance
***           definedby TOL = 1.000000E-04.
*** TERMINAL  ERROR 3 from CHFAC. N = 2 and LDA = 1. N must be less than or
***           equal to LDA.
*** TERMINAL  ERROR 5 from CHFAC. LDR = -2. LDR must be greater than or
***           equal to 1.
*** WARNING   ERROR 1 from SMPRR. CONPER = 9.500000E-01. The confidence
***           percentage is less than 50.0. Commonly used confidence
***           percentages are: 90.0, 95.0 or 99.0.
*** WARNING   ERROR 3 from SMPRR. The sample size, STAT(19) = 10. This is
***           less than 30. The confidence limits, which are computed using
***           a normal approximation, may not be very accurate.
*** WARNING   ERROR 7 from SMPRR. The coefficient of variation of one or
***           both of the variables exceeds 10%. The confidence limits,
***           which are computed using a normal approximation, may not be
***           very accurate.
```

Example of Traceback

The next program illustrates a situation in which a traceback is produced. Although the traceback shows an error code associated with a terminal error, this code has no meaning to the user; the printed message contains all relevant information and it is not assumed that the user would take corrective action based on knowledge of the code.

```
      USE IMSL_LIBRARIES
!                                  Specifications for local variables
      REAL       A, B, ERRABS, ERRREL, RESULT, ERREST
!                                  Specifications for common variables
      REAL       PIN, QIN, SAMP
      COMMON     PIN, QIN, SAMP
!                                  Specifications for functions
      EXTERNAL F
      REAL F
!                                  Compute the expected value of the
!                                  maximum order statistic in a sample
!                                  of size SAMP from a beta distribution.
      A = 0.0
      B = 1.0
      ERRABS = 0.0
      ERRREL = 0.001
!                                  Initialize parameters for the beta
!                                  order statistic of interest.
      SAMP = 10.0
      PIN = 2.0
      QIN = -3.0
!                                  The parameters for the beta must be
!                                  nonnegative -- hence, the preceeding
!                                  assignment causes an error.
      CALL QDAGS (F, A, B, RESULT, ERRABS=ERRABS, ERREST=ERREST)
!
      WRITE (*, *) RESULT, ERREST
      END
!
      REAL FUNCTION F (X)
      USE BETDF_INT
      REAL       X, PIN, QIN, SAMP
      COMMON     PIN, QIN, SAMP
!
      F = X*BETDF(X,PIN,QIN)**(SAMP-1.0)
      RETURN
      END
```

Output

```
*** TERMINAL ERROR 4 from BETDF.   QIN = -3.000000E+00 must be greater than
***          0.0.
    Here is a traceback of subprogram calls in reverse order:
    Routine name                     Error type Error code
    ------------                     ---------- ----------
    BETDF                                 5          4
    Q2AGS                                 0          0 (Called internally)
    QDAGS                                 0          0
    USER                                  0          0
```

Machine-Dependent Constants

The function subprograms in this section return machine-dependent information and can be used to enhance portability of programs between different computers. The routines IMACH, and AMACH describe the computer's arithmetic. The routine UMACH describes the input, ouput, and error output unit numbers.

IMACH

This function retrieves machine integer constants that define the arithmetic used by the computer.

Function Return Value

IMACH(1) = Number of bits per integer storage unit.

IMACH(2) = Number of characters per integer storage unit:

Integers are represented in M-digit, base A form as

$$\sigma \sum_{k=0}^{M} x_k A^k$$

where σ is the sign and $0 \le x_k < A$, $k = 0, \ldots, M$.

Then,

IMACH(3) = A, the base.

IMACH(4) = M, the number of base-A digits.

IMACH(5) = $A^M - 1$, the largest integer.

The machine model assumes that floating-point numbers are represented in normalized N-digit base B form as

$$\sigma B^E \sum_{k=1}^{N} x_k B^{-k}$$

where σ is the sign, $0 < x_1 < B$, $0 \le x_k < B$, k = 2, …, N and $E_{\min} \le E \le E_{\max}$. Then,

IMACH(6) = B, the base.

IMACH(7) = N_s, the number of base-B digits in single precision.

IMACH(8) = $E_{\min_s}$, the smallest single precision exponent.

IMACH(9) = $E_{\max_s}$, the largest single precision exponent.

IMACH(10) = N_d, the number of base-B digits in double precision.

IMACH(11) = $E_{\min_d}$, the smallest double precision exponent.

IMACH(12) = $E_{\max_d}$, the number of base-B digits in double precision

Required Arguments

I — Index of the desired constant. (Input)

FORTRAN 90 Interface

Generic: `IMACH (I)`

Specific: The specific interface name is `IMACH`.

FORTRAN 77 Interface

Single: `IMACH (I)`

AMACH

The function subprogram `AMACH` retrieves machine constants that define the computer's single-precision or double precision arithmetic. . Such floating-point numbers are represented in normalized N_s-digit, base B form as

$$\sigma B^E \sum_{k=1}^{N_s} x_k B^{-k}$$

where σ is the sign, $0 < x_1 < B$, $0 \le x_k < B$, $k = 2, \ldots, N_s$ and

$$E_{\min_s} \le E \le E_{\max_s}$$

Function Return Value

AMACH(l) = $B^{E_{\min}-1}$, the smallest normalized positive number.

AMACH(2)=$B^{E_{\max}}\left(1-B^{-N}\right)$, the largest number.

AMACH(3)=B^{-N}, the smallest relative spacing.

AMACH(4)=B^{1-N}, the largest relative spacing.

AMACH(5)=$\log_{10}(B)$.

AMACH(6) = NaN (*quiet* not a number).

AMACH(7) = positive machine infinity.

AMACH(8) = negative machine infinity.

See Comment 1 for a description of the use of the generic version of this function.

See Comment 2 for a description of `min`, `max`, and `N`.

Required Arguments

I — Index of the desired constant. (Input)

FORTRAN 90 Interface

Generic: `AMACH (I)`

Specific: The specific interface names are `S_AMACH` and `D_AMACH`.

FORTRAN 77 Interface

Single: `AMACH (I)`

Double: The double precision name is `DMACH`.

Comments

1. If the generic version of this function is used, the immediate result must be stored in a variable before use in an expression. For example:

```
X = AMACH(I)
Y = SQRT(X)
```

must be used rather than

```
Y = SQRT(AMACH(I)).
```

If this is too much of a restriction on the programmer, then the specific name can be used without this restriction..

2. Note that for single precision B = `IMACH`(6), N = `IMACH`(7).
E`min` = `IMACH`(8), and E`max` = `IMACH`(9).
For double precision B = `IMACH`(6), N = `IMACH`(10).
E`min` = `IMACH`(11), and E`max` = `IMACH`(12).

3. The IEEE standard for binary arithmetic (see IEEE 1985) specifies *quiet* NaN (not a number) as the result of various invalid or ambiguous operations, such as 0/0. The intent is that AMACH(6) return a quiet NaN. On computers that do not support a *quiet* NaN, a *quiet* NaN, a special value near AMACH(2) is returned for AMACH(6). On computers that do not have a special representation for infinity, AMACH(7) returns the same value as AMACH(2).

DMACH

See AMACH.

IFNAN(X)

This logical function checks if the argument X is NaN (not a number).

Function Return Value

IFNAN **-** Logical function value. True is returned if the input argument is a NAN**.** Otherwise, False is returned. (Output)

Required Arguments

X – Argument for which the test for NAN is desired. (Input)

FORTRAN 90 Interface

Generic: `IFNAN(X)`

Specific: The specific interface names are `S_IFNAN` and `D_IFNAN`.

FORTRAN 77 Interface

Single: `IFNAN (X)`

Double: The double precision name is `DIFNAN`.

Example

```
      USE IFNAN_INT
      USE AMACH_INT
      USE UMACH_INT
      INTEGER      NOUT
      REAL         X
!
      CALL UMACH (2, NOUT)
!
      X = AMACH(6)
```

```
      IF (IFNAN(X)) THEN
         WRITE (NOUT,*) ' X is NaN (not a number).'
      ELSE
         WRITE (NOUT,*) ' X = ', X
      END IF
!
      END
```

Output

```
X is NaN (not a number).
```

Description

The logical function IFNAN checks if the single or double precision argument X is NAN (not a number). The function IFNAN is provided to facilitate the transfer of programs across computer systems. This is because the check for NaN can be tricky and not portable across computer systems that do not adhere to the IEEE standard. For example, on computers that support the IEEE standard for binary arithmetic (see IEEE 1985), NaN is specified as a bit format not equal to itself. Thus, the check is performed as

```
IFNAN = X .NE. X
```

On other computers that do not use IEEE floating-point format, the check can be performed as:

```
IFNAN = X .EQ. AMACH(6)
```

The function IFNAN is equivalent to the specification of the function Isnan listed in the Appendix, (IEEE 1985). The above example illustrates the use of IFNAN. If X is NaN, a message is printed instead of X. (Routine UMACH, which is described in the following section, is used to retrieve the output unit number for printing the message.)

UMACH

Routine UMACH sets or retrieves the input, output, or error output device unit numbers.

Required Arguments

N — Integer value indicating the action desired. If the value of N is negative, the input, output, or error output unit number is reset to NUNIT. If the value of N is positive, the input, output, or error output unit number is returned in NUNIT. See the table in argument NUNIT for legal values of N. (Input)

NUNIT — The unit number that is either retrieved or set, depending on the value of input argument N. (Input/Output)

The arguments are summarized by the following table:

N	Effect
1	Retrieves input unit number in NUNIT.

N	Effect
2	Retrieves output unit number in NUNIT.
3	Retrieves error output unit number in NUNIT.
–1	Sets the input unit number to NUNIT.
–2	Sets the output unit number to NUNIT.
–3	Sets the error output unit number to NUNIT.

FORTRAN 90 Interface

Generic: `CALL UMACH (N, NUNIT)`

Specific: The specific interface name is `UMACH`.

FORTRAN 77 Interface

Single: `CALL UMACH (N, NUNIT)`

Example

In the following example, a terminal error is issued from the MATH/LIBRARY AMACH function since the argument is invalid. With a call to UMACH, the error message will be written to a local file named "CHECKERR".

```
      USE AMACH_INT
      USE UMACH_INT
      INTEGER     N, NUNIT
      REAL        X
!                                       Set Parameter
      N = 0
!
      NUNIT = 9
      CALL UMACH (-3, NUNIT)
      OPEN (UNIT=9,FILE='CHECKERR')
      X = AMACH(N)
      END
```

Output

The output from this example, written to "CHECKERR" is:

```
*** TERMINAL ERROR 5 from AMACH.  The argument must be between 1 and 8
***           inclusive. N = 0
```

Description

Routine UMACH sets or retrieves the input, output, or error output device unit numbers. UMACH is set automatically so that the default FORTRAN unit numbers for standard input, standard output,

and standard error are used. These unit numbers can be changed by inserting a call to UMACH at the beginning of the main program that calls MATH/LIBRARY routines. If these unit numbers are changed from the standard values, the user should insert an appropriate OPEN statement in the calling program.

Matrix Storage Modes

In this section, the word *matrix* will be used to refer to a mathematical object, and the word *array* will be used to refer to its representation as a FORTRAN data structure.

General Mode

A *general* matrix is an $N \times N$ matrix A. It is stored in a FORTRAN array that is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter LDA is called the *leading dimension* of A. It must be at least as large as N. IMSL general matrix subprograms only refer to values A_{ij} for $i = 1, \ldots, N$ and $j = 1, \ldots, N$. The data type of a general array can be one of REAL, DOUBLE PRECISION, or COMPLEX. If your FORTRAN compiler allows, the nonstandard data type DOUBLE COMPLEX can also be declared.

Rectangular Mode

A *rectangular* matrix is an $M \times N$ matrix A. It is stored in a FORTRAN array that is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter LDA is called the *leading dimension* of A. It must be at least as large as M. IMSL rectangular matrix subprograms only refer to values A_{ij} for $i = 1, \ldots, M$ and $j = 1, \ldots, N$. The data type of a rectangular array can be REAL, DOUBLE PRECISION, or COMPLEX. If your FORTRAN compiler allows, you can declare the nonstandard data type DOUBLE COMPLEX.

Symmetric Mode

A symmetric matrix is a square $N \times N$ matrix A, such that $A^T = A$. (A^T is the transpose of A.) It is stored in a FORTRAN array that is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter LDA is called the *leading dimension* of A. It must be at least as large as N. IMSL symmetric matrix subprograms only refer to the upper or to the lower half of A (i.e., to values A_{ij} for $i = 1, \ldots, N$ and $j = i, \ldots, N$, or A_{ij} for $j = 1, \ldots, N$ and $i = j, \ldots, N$). The data type of a symmetric array can be one of REAL or DOUBLE PRECISION. Use of the upper half of the array is denoted in the BLAS that compute with symmetric matrices, see Chapter 9, Programming Notes for BLAS, using the CHARACTER*1 flag UPLO = 'U'. Otherwise, UPLO = 'L' denotes that the lower half of the array is used.

Hermitian Mode

A *Hermitian* matrix is a square $N \times N$ matrix A, such that

$$\overline{A}^T = A$$

The matrix

$$\overline{A}$$

is the complex conjugate of A and

$$A^H \equiv \overline{A}^T$$

is the conjugate transpose of A. For Hermitian matrices, $A^H = A$. The matrix is stored in a FORTRAN array that is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter `LDA` is called the *leading dimension* of `A`. It must be at least as large as N. IMSL Hermitian matrix subprograms only refer to the upper or to the lower half of `A` (i.e., to values A_{ij} for $i = 1, \ldots, N$ and $j = i, \ldots, N$., or A_{ij} for $j = 1, \ldots, N$ and $i = j, \ldots, N$). Use of the upper half of the array is denoted in the BLAS that compute with Hermitian matrices, see Chapter 9, Programming Notes for BLAS, using the `CHARACTER*1` flag `UPLO = 'U'`. Otherwise, `UPLO = 'L'` denotes that the lower half of the array is used. The data type of a Hermitian array can be `COMPLEX` or, if your FORTRAN compiler allows, the nonstandard data type `DOUBLE COMPLEX`.

Triangular Mode

A *triangular* matrix is a square $N \times N$ matrix A such that values $A_{ij} = 0$ for $i < j$ or $A_{ij} = 0$ for $i > j$. The first condition defines a *lower* triangular matrix while the second condition defines an *upper* triangular matrix. A lower triangular matrix A is stored in the lower triangular part of a FORTRAN array `A`. An upper triangular matrix is stored in the upper triangular part of a FORTRAN array. Triangular matrices are called *unit* triangular whenever $A_{jj} = 1, j = 1, \ldots, N$. For unit triangular matrices, only the strictly lower or upper parts of the array are referenced. This is denoted in the BLAS that compute with triangular matrices, see Chapter 9, Programming Notes for BLAS, using the `CHARACTER*1` flag `DIAG = 'U'`. Otherwise, `DIAG = 'N'` denotes that the diagonal array terms should be used. For unit triangular matrices, the diagonal terms are each used with the mathematical value 1. The array diagonal term does not need to be 1.0 in this usage. Use of the upper half of the array is denoted in the BLAS that compute with triangular matrices, see Chapter 9, Programming Notes for BLAS, using the `CHARACTER*1` flag `UPLO = 'U'`. Otherwise, `UPLO = 'L'` denotes that the lower half of the array is used. The data type of an array that contains a triangular matrix can be one of `REAL`, `DOUBLE PRECISION`, or `COMPLEX`. If your FORTRAN compiler allows, the nonstandard data type `DOUBLE COMPLEX` can also be declared.

Band Storage Mode

A *band matrix* is an $M \times N$ matrix A with all of its nonzero elements "close" to the main diagonal. Specifically, values $A_{ij} = 0$ if $i - j >$ `NLCA` or $j - i >$ `NUCA`. The integers `NLCA` and `NUCA` are the *lower* and *upper* band widths. The integer $m =$ `NLCA` + `NUCA` + 1 is the total band width. The diagonals, other than the main diagonal, are called *codiagonals*. While any $M \times N$ matrix is a band

matrix, the band matrix mode is most useful only when the number of nonzero codiagonals is much less than m.

In the band storage mode, the NLCA lower codiagonals and NUCA upper codiagonals are stored in the rows of a FORTRAN array of dimension $m \times N$. The elements are stored in the same column of the array as they are in the matrix. The values A_{ij} inside the band width are stored in array positions $(i - j + \text{NUCA} + 1, j)$. This array is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter LDA is called the *leading dimension* of A. It must be at least as large as m. The data type of a band matrix array can be one of REAL, DOUBLE PRECISION, COMPLEX or, if your FORTRAN compiler allows, the nonstandard data type DOUBLE COMPLEX . Use of the CHARACTER*1 flag TRANS='N' in the BLAS, , see Chapter 9, Programming Notes for BLAS, specifies that the matrix A is used. The flag value

$$\text{TRANS} = \text{'T' uses } A^T$$

while

$$\text{TRANS} = \text{'C' uses } \overline{A}^T$$

For example, consider a real 5×5 band matrix with 1 lower and 2 upper codiagonals, stored in the FORTRAN array declared by the following statements:

```
PARAMETER (N=5, NLCA=1, NUCA=2)
REAL A(NLCA+NUCA+1, N)
```

The matrix A has the form

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 & 0 \\ A_{21} & A_{22} & A_{23} & A_{24} & 0 \\ 0 & A_{32} & A_{33} & A_{34} & A_{35} \\ 0 & 0 & A_{43} & A_{44} & A_{45} \\ 0 & 0 & 0 & A_{54} & A_{55} \end{bmatrix}$$

As a FORTRAN array, it is

$$A = \begin{bmatrix} \times & \times & A_{13} & A_{24} & A_{35} \\ \times & A_{12} & A_{23} & A_{34} & A_{45} \\ A_{11} & A_{22} & A_{33} & A_{44} & A_{55} \\ A_{21} & A_{32} & A_{43} & A_{54} & \times \end{bmatrix}$$

The entries marked with an x in the above array are not referenced by the IMSL band subprograms.

Band Symmetric Storage Mode

A *band symmetric* matrix is a band matrix that is also symmetric. The band symmetric storage mode is similar to the band mode except only the lower or upper codiagonals are stored.

In the band symmetric storage mode, the NCODA upper codiagonals are stored in the rows of a FORTRAN array of dimension (NCODA + 1) × N. The elements are stored in the same column of the array as they are in the matrix. Specifically, values $A_{ij}, j \le i$ inside the band are stored in array positions $i - j$ + NCODA + 1, j). This is the storage mode designated by using the CHARACTER*1 flag UPLO = 'U' in Level 2 BLAS that compute with band symmetric matrices, , see Chapter 9, Programming Notes for BLAS. Alternatively, $A_{ij}, j \le i$, inside the band, are stored in array positions ($i - j$ + 1, j). This is the storage mode designated by using the CHARACTER*1 flag UPLO = 'L' in these Level 2 BLAS, see Chapter 9, Programming Notes for BLAS. The array is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter LDA is called the *leading dimension* of A. It must be at least as large as NCODA + 1. The data type of a band symmetric array can be REAL or DOUBLE PRECISION.

For example, consider a real 5 × 5 band matrix with 2 codiagonals. Its FORTRAN declaration is

```
PARAMETER (N=5, NCODA=2)
REAL A(NCODA+1, N)
```

The matrix A has the form

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 & 0 \\ A_{12} & A_{22} & A_{23} & A_{24} & 0 \\ A_{13} & A_{23} & A_{33} & A_{34} & A_{35} \\ 0 & A_{24} & A_{34} & A_{44} & A_{45} \\ 0 & 0 & A_{35} & A_{45} & A_{55} \end{bmatrix}$$

Since A is symmetric, the values $A_{ij} = A_{ji}$. In the FORTRAN array, it is

$$A = \begin{bmatrix} \times & \times & A_{13} & A_{24} & A_{35} \\ \times & A_{12} & A_{23} & A_{34} & A_{45} \\ A_{11} & A_{22} & A_{33} & A_{44} & A_{55} \end{bmatrix}$$

The entries marked with an × in the above array are not referenced by the IMSL band symmetric subprograms.

An alternate storage mode for band symmetric matrices is designated using the CHARACTER*1 flag UPLO = 'L' in Level 2 BLAS that compute with band symmetric matrices, see Chapter 9, Programming Notes for BLAS. In that case, the example matrix is represented as

$$A = \begin{bmatrix} A_{11} & A_{22} & A_{33} & A_{44} & A_{55} \\ A_{12} & A_{23} & A_{34} & A_{45} & \times \\ A_{13} & A_{24} & A_{35} & \times & \times \end{bmatrix}$$

Band Hermitian Storage Mode

A *band Hermitian* matrix is a band matrix that is also Hermitian. The band Hermitian mode is a complex analogue of the band symmetric mode.

In the band Hermitian storage mode, the NCODA upper codiagonals are stored in the rows of a FORTRAN array of dimension (NCODA + 1) × *N*. The elements are stored in the same column of the array as they are in the matrix. In the Level 2 BLAS, see Chapter 9, Programming Notes for BLAS, this is denoted by using the CHARACTER*1 flag UPLO ='U'. The array is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter LDA is called the *leading dimension* of *A*. It must be at least as large as (NCODA + 1) . The data type of a band Hermitian array can be COMPLEX or, if your FORTRAN compiler allows, the nonstandard data type DOUBLE COMPLEX.

For example, consider a complex 5 × 5 band matrix with 2 codiagonals. Its FORTRAN declaration is

```
PARAMETER (N=5, NCODA = 2)
COMPLEX A(NCODA + 1, N)
```

The matrix *A* has the form

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 & 0 \\ \overline{A}_{12} & A_{22} & A_{23} & A_{24} & 0 \\ \overline{A}_{13} & \overline{A}_{23} & A_{33} & A_{34} & A_{35} \\ 0 & \overline{A}_{24} & \overline{A}_{34} & A_{44} & A_{45} \\ 0 & 0 & \overline{A}_{35} & \overline{A}_{45} & A_{55} \end{bmatrix}$$

where the value

$$\overline{A}_{ij}$$

is the complex conjugate of A_{ij}. This matrix represented as a FORTRAN array is

$$A = \begin{bmatrix} \times & \times & A_{13} & A_{24} & A_{35} \\ \times & A_{12} & A_{23} & A_{34} & A_{45} \\ A_{11} & A_{22} & A_{33} & A_{44} & A_{55} \end{bmatrix}$$

The entries marked with an × in the above array are not referenced by the IMSL band Hermitian subprograms.

An alternate storage mode for band Hermitian matrices is designated using the CHARACTER*1 flag UPLO = 'L' in Level 2 BLAS that compute with band Hermitian matrices, see Chapter 9, Programming Notes for BLAS. In that case, the example matrix is represented as

$$A = \begin{bmatrix} A_{11} & A_{22} & A_{33} & A_{44} & A_{55} \\ \overline{A}_{12} & \overline{A}_{23} & \overline{A}_{34} & \overline{A}_{45} & \times \\ \overline{A}_{13} & \overline{A}_{24} & \overline{A}_{35} & \times & \times \end{bmatrix}$$

Band Triangular Storage Mode

A *band triangular* matrix is a band matrix that is also triangular. In the band triangular storage mode, the NCODA codiagonals are stored in the rows of a FORTRAN array of dimension (NCODA +

1) × N. The elements are stored in the same column of the array as they are in the matrix. For usage in the Level 2 BLAS, see Chapter 9, Programming Notes for BLAS, the `CHARACTER*1` flag `DIAG` has the same meaning as used in section "Triangular Storage Mode". The flag `UPLO` has the meaning analogous with its usage in the section "Banded Symmetric Storage Mode". This array is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter `LDA` is called the *leading dimension* of A. It must be at least as large as (`NCODA` + 1).

For example, consider a 5 ×5 band upper triangular matrix with 2 codiagonals. Its FORTRAN declaration is

```
PARAMETER (N = 5, NCODA = 2)
COMPLEX A(NCODA + 1, N)
```

The matrix A has the form

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 & 0 \\ 0 & A_{22} & A_{23} & A_{24} & 0 \\ 0 & 0 & A_{33} & A_{34} & A_{35} \\ 0 & 0 & 0 & A_{44} & A_{45} \\ 0 & 0 & 0 & 0 & A_{55} \end{bmatrix}$$

This matrix represented as a FORTRAN array is

$$A = \begin{bmatrix} \times & \times & A_{13} & A_{24} & A_{35} \\ \times & A_{12} & A_{23} & A_{34} & A_{45} \\ A_{11} & A_{22} & A_{33} & A_{44} & A_{55} \end{bmatrix}$$

This corresponds to the `CHARACTER*1` flags `DIAG = 'N'` and `UPLO = 'U'`. The matrix A^T is represented as the FORTRAN array

$$A = \begin{bmatrix} A_{11} & A_{22} & A_{33} & A_{44} & A_{55} \\ A_{12} & A_{23} & A_{34} & A_{45} & \times \\ A_{13} & A_{24} & A_{35} & \times & \times \end{bmatrix}$$

This corresponds to the `CHARACTER*1` flags `DIAG = 'N'` and `UPLO = 'L'`. In both examples, the entries indicated with an × are not referenced by IMSL subprograms.

Codiagonal Band Symmetric Storage Mode

This is an alternate storage mode for band symmetric matrices. It is not used by any of the BLAS, see Chapter 9, Programming Notes for BLAS. Storing data in a form transposed from the **Band Symmetric Storage Mode** maintains unit spacing between consecutive referenced array elements. This data structure is used to get good performance in the Cholesky decomposition algorithm that solves positive definite symmetric systems of linear equations $Ax = b$. The data type can be `REAL` or `DOUBLE PRECISION`. In the codiagonal band symmetric storage mode, the `NCODA` upper

codiagonals and right-hand-side are stored in columns of this FORTRAN array. This array is declared by the following statement:

```
DIMENSION A(LDA, NCODA + 2)
```

The parameter LDA is the *leading positive dimension* of A. It must be at least as large as N + NCODA.

Consider a real symmetric 5 × 5 matrix with 2 codiagonals

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 & 0 \\ A_{12} & A_{22} & A_{23} & A_{24} & 0 \\ A_{13} & A_{23} & A_{33} & A_{34} & A_{35} \\ 0 & A_{24} & A_{34} & A_{44} & A_{45} \\ 0 & 0 & A_{35} & A_{45} & A_{55} \end{bmatrix}$$

and a right-hand-side vector

$$b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

A FORTRAN declaration for the array to hold this matrix and right-hand-side vector is

```
PARAMETER (N = 5, NCODA = 2, LDA = N + NCODA)
REAL A(LDA, NCODA + 2)
```

The matrix and right-hand-side entries are placed in the FORTRAN array A as follows:

$$A = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ A_{11} & \times & \times & b_1 \\ A_{22} & A_{12} & \times & b_2 \\ A_{33} & A_{23} & A_{13} & b_3 \\ A_{44} & A_{34} & A_{24} & b_4 \\ A_{55} & A_{45} & A_{35} & b_5 \end{bmatrix}$$

Entries marked with an × do not need to be defined. Certain of the IMSL band symmetric subprograms will initialize and use these values during the solution process. When a solution is computed, the b_i, $i = 1, \ldots, 5$, are replaced by x_i, $i = 1, \ldots, 5$.

The nonzero A_{ij}, $j \geq i$, are stored in array locations $A(j + \text{NCODA}, (j - i) + 1)$. The right-hand-side entries b_j are stored in locations $A(j + \text{NCODA}, \text{NCODA} + 2)$. The solution entries x_j are returned in $A(j + \text{NCODA}, \text{NCODA} + 2)$.

Codiagonal Band Hermitian Storage Mode

This is an alternate storage mode for band Hermitian matrices. It is not used by any of the BLAS, see Chapter 9, Programming Notes for BLAS. In the codiagonal band Hermitian storage mode, the real and imaginary parts of the 2 * NCODA + 1 upper codiagonals and right-hand-side are stored in columns of a FORTRAN array. Note that there is no explicit use of the COMPLEX or the nonstandard data type DOUBLE COMPLEX data type in this storage mode.

For *Hermitian* complex matrices,

$$A = U + \sqrt{-1}V$$

where U and V are real matrices. They satisfy the conditions $U = U^T$ and $V = -V^T$. The right-hand-side

$$b = c + \sqrt{-1}\, d$$

where c and d are real vectors. The solution vector is denoted as

$$x = u + \sqrt{-1}v$$

where u and v are real. The storage is declared with the following statement

```
DIMENSION A(LDA, 2*NCODA + 3)
```

The parameter LDA is the *leading positive dimension* of A. It must be at least as large as N + NCODA.

The diagonal terms U_{jj} are stored in array locations A (j + NCODA, 1). The diagonal V_{jj} are zero and are not stored. The nonzero $U_{ij}, j > i$, are stored in locations A(j + NCODA, 2 * ($j - i$)).

The nonzero V_{ij} are stored in locations A(j + NCODA, 2*($j - i$) + 1). The right side vector b is stored with c_j and d_j in locations A(j + NCODA, 2*NCODA + 2) and A(j + NCODA, 2*NCODA + 3) respectively. The real and imaginary parts of the solution, u_j and v_j, respectively overwrite c_j and d_j.

Consider a complex hermitian 5 × 5 matrix with 2 codiagonals

$$A = \begin{bmatrix} U_{11} & U_{12} & U_{13} & 0 & 0 \\ U_{12} & U_{22} & U_{23} & U_{24} & 0 \\ U_{13} & U_{23} & U_{33} & U_{34} & U_{35} \\ 0 & U_{24} & U_{34} & U_{44} & U_{45} \\ 0 & 0 & U_{35} & U_{45} & U_{55} \end{bmatrix} + \sqrt{-1} \begin{bmatrix} 0 & V_{12} & V_{13} & 0 & 0 \\ -V_{12} & 0 & V_{23} & V_{24} & 0 \\ -V_{13} & -V_{23} & 0 & V_{34} & V_{35} \\ 0 & -V_{24} & -V_{34} & 0 & V_{45} \\ 0 & 0 & -V_{35} & -V_{45} & 0 \end{bmatrix}$$

and a right-hand-side vector

$$b = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} + \sqrt{-1} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix}$$

A FORTRAN declaration for the array to hold this matrix and right-hand-side vector is

```
PARAMETER (N = 5, NCODA = 2, LDA = N + NCODA)
REAL A(LDA,2*NCODA + 3)
```

The matrix and right-hand-side entries are placed in the FORTRAN array A as follows:

$$A = \begin{bmatrix} \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ U_{11} & \times & \times & \times & \times & c_1 & d_1 \\ U_{22} & U_{12} & V_{12} & \times & \times & c_2 & d_2 \\ U_{33} & U_{23} & V_{23} & U_{13} & V_{13} & c_3 & d_3 \\ U_{44} & U_{34} & V_{34} & U_{24} & V_{24} & c_4 & d_4 \\ U_{55} & U_{45} & V_{45} & U_{35} & V_{35} & c_5 & d_5 \end{bmatrix}$$

Entries marked with an × do not need to be defined.

Sparse Matrix Storage Mode

The sparse linear algebraic equation solvers in Chapter 1 accept the input matrix in *sparse storage mode*. This structure consists of `INTEGER` values `N` and `NZ`, the matrix dimension and the total number of nonzero entries in the matrix. In addition, there are two `INTEGER` arrays `IROW(*)` and `JCOL(*)` that contain unique matrix row and column coordinates where values are given. There is also an array $A(*)$ of values. All other entries of the matrix are zero. Each of the arrays `IROW(*)`, `JCOL(*)`, $A(*)$ must be of size `NZ`. The correspondence between matrix and array entries is given by

$$A_{\text{IROW}(i),\text{JCOL}(i)} = A(i),\ i = 1, \ldots, \text{NZ}$$

The data type for $A(*)$ can be one of `REAL`, `DOUBLE PRECISION`, or `COMPLEX`. If your FORTRAN compiler allows, the nonstandard data type `DOUBLE COMPLEX` can also be declared.

For example, consider a real 5 × 5 sparse matrix with 11 nonzero entries. The matrix A has the form

$$A = \begin{bmatrix} A_{11} & 0 & A_{13} & A_{14} & 0 \\ A_{21} & A_{22} & 0 & 0 & 0 \\ 0 & A_{32} & A_{33} & A_{34} & 0 \\ 0 & 0 & A_{43} & 0 & 0 \\ 0 & 0 & 0 & A_{54} & A_{55} \end{bmatrix}$$

Declarations of arrays and definitions of the values for this sparse matrix are

```
PARAMETER (NZ = 11, N = 5)
DIMENSION IROW(NZ), JCOL(NZ), A(NZ)
DATA IROW /1,1,1,2,2,3,3,3,4,5,5/
DATA JCOL /1,3,4,1,2,2,3,4,3,4,5/
DATA A    /A11,A13,A14,A21,A22,A32,A33,A34, &
           A43,A54,A55/
```

Reserved Names

When writing programs accessing the STAT/LIBRARY, the user should choose FORTRAN names that do not conflict with names of IMSL subroutines, functions, or named common blocks, such as the workspace common block WORKSP (page 1522). The user needs to be aware of two types of name conflicts that can arise. The first type of name conflict occurs when a name (technically a *symbolic name*) is not uniquely defined within a program unit (either a main program or a subprogram). For example, such a name conflict exists when the name RCURV is used to refer both to a type REAL variable and to the IMSL subroutine RCURV in a single program unit. Such errors are detected during compilation and are easy to correct. The second type of name conflict, which can be more serious, occurs when names of program units and named common blocks are not unique. For example, such a name conflict would be caused by the user defining a subroutine named WORKSP and also referencing an STAT/LIBRARY subroutine that uses the named common block WORKSP. Likewise, the user must not define a subprogram with the same name as a subprogram in the STAT/LIBRARY, that is referenced directly by the user's program or is referenced indirectly by other STAT/LIBRARY subprograms.

The STAT/LIBRARY consists of many routines, some that are described in the *User's Manual* and others that are not intended to be called by the user and, hence, that are not documented. If the choice of names were completely random over the set of valid FORTRAN names, and if a program uses only a small subset of the STAT/LIBRARY, the probability of name conflicts is very small. Since names are usually chosen to be mnemonic, however, the user may wish to take some precautions in choosing FORTRAN names.

Many IMSL names consist of a root name that may have a prefix to indicate the type of the routine. For example, the IMSL single precision subroutine for fitting a polynomial by least squares has the name RCURV, which is the root name, and the corresponding IMSL double precision routine has the name DRCURV. Associated with these two routines are R2URV and DR2URV. RCURV and DRCURV are listed in the Alphabetical Index of Routines, but R2URV and DR2URV are not. The user of RCURV must consider both names RCURV and R2URV to be reserved; likewise, the user of DRCURV must consider both names DRCURV and DR2URV to be reserved. The names of *all* routines and named common blocks that are used by the STAT/LIBRARY and that do not have a numeral in the second position of the root name are listed in the Alphabetical Summary of Routines.

The careful user can avoid any conflicts with IMSL names if the following rules are observed:

- Do not choose a name that appears in the Alphabetical Summary of Routines in the *User's Manual*, nor one of these names preceded by a D, S_, D_, C_, or Z_.
- Do not choose a name of three or more characters with a numeral in the second or third position.

These simplified rules include many combinations that are, in fact, allowable. However, if the user selects names that conform to these rules, no conflict will be encountered.

Deprecated Features and Renamed Routines

Automatic Workspace Allocation

FORTRAN subroutines that work with arrays as input and output often require extra arrays for use as workspace while doing computations or moving around data. IMSL routines generally do not require the user explicitly to allocate such arrays for use as workspace. On most systems the workspace allocation is handled transparently. The only limitation is the actual amount of memory available on the system.

On some systems the workspace is allocated out of a stack that is passed as a FORTRAN array in a named common block WORKSP. A very similar use of a workspace stack is described by Fox et al. (1978, pages 116–121). (For compatiblity with older versions of the IMSL Libraries, space is allocated from the COMMON block, if possible.)

The arrays for workspace appear as arguments in lower-level routines. For example, the IMSL routine FREQ (see Chapter 1, Basic Statistics), which computes frequency tabulations, needs arrays for workspace. FREQ allocates arrays from the common area and passes them to the lower-level routine F2EQ, which does the computations. In the "Comments" section of the documentation for FREQ, the amount of workspace is noted, and the call to F2EQ is described. This scheme for using lower-level routines is followed throughout the IMSL Libraries. The names of these routines have a "2" in the second position (or in the third position in double precision routines having a "D" prefix). The user can provide workspace explicitly and call directly the "2-level" routine, which is documented along with the main routine. In a very few cases, the 2-level routine allows additional options that the main routine does not allow.

Prior to returning to the calling program, a routine that allocates workspace generally deallocates that space, so that it becomes available for use in other routines. There are some exceptions to this, as noted in the section "IDO Routines" which follows later in this chapter.

Changing the Amount of Space Allocated

This section is relevant only to those systems on which the transparent workspace allocator is not available.

By default, the total amount of space allocated in the common area for storage of numeric data is 5000 numeric storage units. (A numeric storage unit is the amount of space required to store an integer or a real number. By comparison, a double precision unit is twice this amount. Therefore the total amount of space allocated in the common area for storage of numeric data is 2500 double precision units.) This space is allocated as needed for INTEGER, REAL, or other numeric data. For larger problems in which the default amount of workspace is insufficient, the user can change the allocation by supplying the FORTRAN statements to define the array in the named common block and by informing the IMSL workspace allocation system of the new size of the common array. To request 7000 units, the statements are

```
COMMON /WORKSP/ RWKSP
REAL RWKSP(7000)
CALL IWKIN(7000)
```

If an IMSL routine attempts to allocate workspace in excess of the amount available in the common stack, the routine issues a fatal error message that indicates how much space is needed

and prints statements like those above to guide the user in allocating the necessary amount. The program below uses IMSL routine PERMA (see Chapter 19, Utilities) to permute rows or columns of a matrix. This routine requires workspace equal to the number of columns, which in this example is too large. (Note that the work vector RWKSP must also provide extra space for bookkeeping.)

```
!                                  Specifications for local variables
      INTEGER    NRA, NCA, LDA, IPERMU(6000), IPATH
      REAL A(2,6000)
!                                  Specifications for subroutines
      EXTERNAL PERMA
!
      NRA = 2
      NCA = 6000
      LDA = 2
!                                  Initialize permutation index
      DO 10 I = 1, NCA
         IPERMU(I) = NCA + 1 - I
   10 CONTINUE
      IPATH = 2
      CALL PERMA (NRA, NCA, A, LDA, IPERMU, IPATH, A, LDA)
      END
```

Output

```
*** TERMINAL ERROR 10 from PERMA.  Insufficient workspace for current
***          allocation(s). Correct by calling IWKIN from main program with
***          the three following statements: (REGARDLESS OF PRECISION)
***                COMMON /WORKSP/ RWKSP
***                REAL RWKSP(6018)
***                CALL IWKIN(6018)
*** TERMINAL ERROR 10 from PERMA.  Workspace allocation was based on NCA =
***          6000.
```

In most cases, the amount of workspace is dependent on the parameters of the problem so the amount needed is known exactly. In a few cases, however, the amount of workspace is dependent on the data (for example, if it is necessary to count all of the unique values in a vector), so the IMSL routine cannot tell in advance exactly how much workspace is needed. In such cases the error message printed is an estimate of the amount of space required.

IDO Routines

Some routines with an argument named "IDO" allocate workspace automatically and store intermediate results in elements of workspace that are referenced in subsequent calls. Typically, these routines are called in a loop. With each call, some rows of the data set are input to the routine and statistics stored in workspace are updated. In this case, the workspace must be preserved between calls.

For these routines, when IDO indicates this is the first call, the routine allocates workspace; when IDO indicates this is the last call, the routine deallocates the workspace. Because of the way this workspace is allocated and deallocated, no IMSL routine requiring additional automatic workspace can be used between these two calls. If it is necessary to call additional routines requiring workspace, use the 2-level routines and explicitly allocate the work arrays.

Not all IDO routines require workspace to be preserved between their first and last call. Some may not even use workspace. Others may allocate and deallocate workspace with each call. The statement "workspace should not be changed between calls" will be in the description of the "IDO" routine that requires that workspace be preserved. (This statement will occur in the description of one or more of the workspace arguments for the 2-level routine.)

Character Workspace

Since character arrays cannot be equivalenced with numeric arrays, a separate named common block WKSPCH is provided for character workspace. In most respects this stack is managed in the same way as the numeric stack. The default size of the character workspace is 2000 character units. (A character unit is the amount of space required to store one character.) The routine analogous to IWKIN used to change the default allocation is IWKCIN.

The routines in the following list are being deprecated in Version 2.0 of STAT/LIBRARY. A deprecated routine is one that is no longer used by anything in the library but is being included in the product for those users who may be currently referencing it in their application. However, any future versions of STAT/LIBRARY will not include these routines. If any of these routines are being called within an application, it is recommended that you change your code or retain the deprecated routine before replacing this library with the next version. Most of these routines were called by users only when they needed to set up their own workspace. Thus, the impact of these changes should be limited.

```
DHOUAP
DHOUTR
DG2DF
DG2IN
DG3DF
G2DF
G2IN
G3DF
SHOUAP
SHOUTR
```

The following routines have been renamed due to naming conflicts with other software manufacturers.

CTIME — replaced with CPSEC

DTIME — replaced with TIMDY

PAGE — replaced with PGOPT

Appendix A: GAMS Index

Description

This index lists routines in STAT/LIBRARY by a tree-structured classification scheme known as GAMS. Boisvert, Howe, Kahaner, and Springmann (1990)
give the GAMS classification scheme. The classification scheme given here is Version 2.0.

The first level of the full classification scheme is denoted by a letter A thru Z as follows:

A. Arithmetic, Error Analysis
B. Number Theory
C. Elementary and Special Functions
D. Linear Algebra
E. Interpolation
F. Solution of Nonlinear Equations
G. Optimization
H. Differentiation and Integration
I. Differential and Integral Equations
J. Integral Transforms
K. Approximation
L. Statistics, Probability
M. Simulation, Stochastic Modeling
N. Data Handling
O. Symbolic Computation
P. Computational Geometry
Q. Graphics
R. Service Routines
S. Software Development Tools
Z. Other

There are seven levels in the classification scheme. Classes in the first level are identified by a capital letter as is given above. Classes in the remaining levels are identified by alternating letter-and-number combinations. A single letter (a–z) is used with the odd-numbered levels. A number (1–26) is used within the even-numbered levels.

IMSL STAT/LIBRARY

C.....ELEMENTARY AND SPECIAL FUNCTIONS (*search also class L5*)

C3 .....Polynomials

C3a .....Orthogonal

`OPOLY` Generate orthogonal polynomials with respect to x values and specified weights.

C7 .....Gamma

C7e .....Incomplete gamma

`CHIDF` Evaluate the chi-squared distribution function.

`CHIIN` Evaluate the inverse of the chi-squared distribution function.

`GAMDF` Evaluate the gamma distribution function.

`GAMIN` Evaluate the inverse of the gamma distribution function.

C7f.....Incomplete gamma

`BETDF` Evaluate the beta probability distribution function.

`BETIN` Evaluate the inverse of the beta distribution function.

C8 .....Error functions

C8a .....Error functions, their inverses, integrals, including the normal distribution function

`ANORDF` Evaluate the standard normal (Gaussian) distribution function.

`ANORIN` Evaluate the inverse of the standard normal (Gaussian) distribution function.

K.....APPROXIMATION (*search also class L8*)

K1 .....Least squares (L_2) approximation

K1a .....Linear least squares (*search also classes D5, D6, D9*)

K1a1 .....Unconstrained

`RCOV` Fit a multiple linear regression model given the variance-covariance matrix.

`RGIVN` Fit a multivariate linear regression model via fast Givens transformations.

`RGLM` Fit a multivariate general linear model.

`RLSE` Fit a multiple linear regression model using least squares.

K1a1a ...Univariate data (curve fitting)

K1ala2 ..Polynomials

`RCURV` Fit a polynomial curve using least squares.

`RFORP` Fit an orthogonal polynomial regression model.

`RPOLY` Analyze a polynomial regression model.

K1a2 .....Constrained

K1a2a ...Linear constraints

`RLEQU` Fit a multivariate linear regression model with linear equality restrictions $HB = G$ imposed on the regression

parameters given results from IMSL routine RGIVN after IDO = 1 and IDO = 2 and prior to IDO = 3.

K1b..... Nonlinear least squares

K1b1..... Unconstrained

K1b1a... Smooth functions

K1b1a1. User provides no derivatives
RNLIN Fit a nonlinear regression model.

K1b1a2. User provides first derivatives
RNLIN Fit a nonlinear regression model.

K2..... Minimax ($L\infty$) approximation
RLMV Fit a multiple linear regression model using the minimax criterion.

K3..... Least absolute value (L_1) approximation
RLLP Fit a multiple linear regression model using the L_p norm criterion.

K4..... Other analytic approximations (e.g., Taylor polynomial, Pade)
RLLP Fit a multiple linear regression model using the L_p norm criterion.

L..... STATISTICS, PROBABILITY

L1 ..... Data summarization

L1a ..... One-dimensional data

L1a1 ..... Raw data
EQTIL Compute empirical quantiles.
LETTR Produce a letter value summary.
ORDST Determine order statistics.

L1a1a ... Location
UVSTA Compute basic univariate statistics.

L1a1b ... Disperson
UVSTA Compute basic univariate statistics.

L1a1c ... Shape
UVSTA Compute basic univariate statistics.

L1a1e ... Ties
NTIES Compute tie statistics for a sample of observations.

L1a3 ..... Grouped data
GRPES Compute basic statistics from grouped data.

L1c ..... Multi-dimensional data

L1c1 ..... Raw data

CSTAT Compute cell frequencies, cell means, and cell sums of squares for multivariate data.

L1c1b ...Covariance, correlation

CORVC Compute the variance-covariance or correlation matrix.

PCORR Compute partial correlations or covariances from the covariance or correlation matrix.

RBCOV Compute a robust estimate of a covariance matrix and mean vector.

L2 .....Data manipulation

L2a .....Transform (*search also classes L10a, N6, and N8*)

BCTR Perform a forward or an inverse Box-Cox (power) transformation.

GCSCP Generate centered variables, squares, and crossproducts.

OPOLY Generate orthogonal polynomials with respect to *x* values and specified weights.

RANKS Compute the ranks, normal scores, or exponential scores for a vector of observations.

L2b .....Tally

CSTAT Compute cell frequencies, cell means, and cell sums of squares for multivariate data.

FREQ Tally multivariate observations into a multi-way frequency table.

OWFRQ Tally observations into a one-way frequency table.

TWFRQ Tally observations into a two-way frequency table.

L2e .....Construct new variables (e.g., indicator variables)

GRGLM Generate regressors for a general linear model.

L3 .....Elementary statistical graphics (*search also class Q*)

L3a .....One-dimensional data

L3a1 .....Histograms

HHSTP Print a horizontal histogram.

VHSTP Print a vertical histogram.

L3a2 .....Frequency, cumulative frequency, percentile plots

CDFP Print a sample cumulative distribution function (CDF), a theoretical CDF, and confidence band information.

L3a3 .....EDA graphics (e.g., box plots)

BOXP Print boxplots for one or more samples.

STMLP Print a stem-and-leaf plot.

L3a4 .....Bar charts

HHSTP Print a horizontal histogram.

VHSTP Print a vertical histogram.

L3b .....Two-dimensional data (*search also class L3e*)

L3b1 .....Histograms (superimposed and bivariate)

VHS2P Print a vertical histogram with every bar subdivided into two parts.

L3b2 ..... Frequency, cumulative frequency

CDF2P Print a plot of two sample cumulative distribution functions.

L3e ..... Multi-dimensional data

L3e3 ..... Scatter diagrams

L3e3a ... Superimposed Y vs. X

PLOTP Print a plot of up to ten sets of points.

SCTP Print a scatterplot of several groups of data.

L3e4 ..... EDA

BOXP Print boxplots for one or more samples.

L4 ..... Elementary data analaysis

L4a ..... One-dimensional data

L4a1 ..... Raw data

L4a1a ... Parametric analysis

CDFP Print a sample cumulative distribution function (CDF), a theoretical CDF, and confidence band information.

L4a1a2 . Probability plots

L4a1a2e Exponential, extreme value

PROBP Print a probability plot.

L4a1a2h Halfnormal

PROBP Print a probability plot.

L4a1a21 Lambfa, logistic, lognormal

PROBP Print a probability plot.

L4a1a2n Negative binomial, normal

PROBP Print a probability plot.

L4a1a2w Weibull

PROBP Print a probability plot.

L4a1a4 . Parameter estimates and tests

L4a1a4b Binomial

BINES Estimate the parameter p of the binomial distribution.

L4a1a4p Poisson

POIES Estimate the parameter of the Poisson distribution.

L4a1b ... Nonparametric analysis

L4a1b1 . Estimates and test regarding location (e.g., median), dispersion and shape

SIGNT Perform a sign test of the hypothesis that a given value is a specified quantile of a distribution.

SNRNK Perform a Wilcoxon signed rank test.

L4a1b2 .Density function estimation

DESKN Perform nonparametric probability density function estimation by the kernel method.

DESPL Perform nonparametric probability density function estimation by the penalized likelihood method.

DESPT Estimate a probability density function at specified points using linear or cubic interpolation.

DNFFT Compute Gaussian kernel estimates of a univariate density via the fast Fourier transform over a fixed interval.

L4a1c....Goodness-of-fit tests

CHIGF Perform a chi-squared goodness-of-fit test.

KSONE Perform a Kolmogorov-Smirnov one-sample test for continuous distributions.

LILLF Perform Lilliefors test for an exponential or normal distribution.

SPWLK Perform a Shapiro-Wilk *W*-test for normality.

L4ald ....Analysis of a sequnce of numbers (*search also class L10a*)

DCUBE Perform a triplets test.

DSQAR Perform a D-square test.

NCTRD Perform the Noether test for cyclical trend.

PAIRS Perform a pairs test.

RUNS Perform a runs up test.

SDPLC Perform the Cox and Stuart sign test for trends in dispersion and location.

L4a3 .....Grouped (and/or censored) data

GRPES Compute basic statistics from grouped data.

NRCES Compute maximum likelihood estimates of the mean and variance from grouped and/or censored normal data.

L4a4 .....Data sampled from a finite population

SMPPR Compute statistics for inferences regarding the population proportion and total, given proportion data from a simple random sample.

SMPPS Compute statistics for inferences regarding the population proportion and total, given proportion data from a stratified random sample.

SMPSC Compute statistics for inferences regarding the population mean and total using single-stage cluster sampling with continuous data.

SMPSR Compute statistics for inferences regarding the population mean and total, given data from a simple random sample.

SMPSS Compute statistics for inferences regarding the population mean and total, given data from a stratified random sample.

SMPST Compute statistics for inferences regarding the population mean and total, given continuous data from a two-stage sample with equisized primary units.

L4b ..... Two dimensional data (*search also class L4c*)

L4b1 ..... Pairwise independent data

L4b1a ... Parametric analysis

L4b1a4 . Parameter estimates and hypothesis tests

TWOMV Compute statistics for mean and variance inferences using samples from two normal populations.

L4b1b ... Nonparametric analysis (e.g., tests based on ranks)

CNCRD Calculate and test the significance of the Kendall coefficient of concordance.

INCLD Perform an includance test.

KENDL Compute and test Kendall's rank correlation coefficient.

RNKSM Perform the Wilcoxon rank sum test.

L4b1c ... Goodness-of-fit tests

KSTWO Perform a Kolmogorov-Smirnov two-sample test.

L4b4 ..... Pairwise dependent grouped data

CTRHO Estimate the bivariate normal correlation coefficient using a contingency table.

TETCC Categorize bivariate data and compute the tetrachoric correlation coefficient.

L4b5 ..... Data sampled from a finite population

SMPRR Compute statistics for inferences regarding the population mean and total using ratio or regression estimation, or inferences regarding the population ratio, given a simple random sample.

SMPRS Compute statistics for inferences regarding the population mean and total using ratio or regression estimation, given continuous data from a stratified random sample.

L4c ..... Multi-dimensional data (*search also classes L4b and L7a1*)

L4c1 ..... Independent data

L4c1b ... Nonparametric analysis

BHAKV Perform a Bhapkar V test.

KRSKL Perform a Kruskal-Wallis test for identical population medians.

KTRND Perform a k-sample trends test against ordered alternatives.

MVMMT Compute Mardia's multivariate measures of skewness and kurtosis and test for multivariate normality.

QTEST Perform a Cochran Q test for related observations.

L4e ..... Multiple multi-dimensional data sets

MVIND Compute a test for the independence of k sets of multivariate normal variables.

L5 .....Function evaluation (*search also class C*)

L5a .....Univariate

L5a1 .....Cumulative distribution functions, probability density functions

L5a1b ...Beta, binomial

BETDF Evaluate the beta probability distribution function.

BINDF Evaluate the binomial distribution function.

BINPR Evaluate the binomial probability function.

L5a1c....Cauchy, chi-squared

CHIDF Evaluate the chi-squared distribution function.

CSNDF Evaluate the noncentral chi-squared distribution function.

L5a1f....*F* distribution

FDF Evaluate the *F* distribution function.

L5a1g ...Gamma, general, geometric

GAMDF Evaluate the gamma distribution function.

GCDF Evaluate a general continuous cumulative distribution function given ordinates of the density.

L5a1h ...Halfnormal, hyergeometric

HYPDF Evaluate the hypergeometric distribution function.

HYPPR Evaluate the hypergeometric probability function.

L5a1k ...Kendall *F* statistic, Kolmogorsv-Smirnov

AKS1DF Evaluate the distribution function of the one-sided Kolmogorov-Smirnov goodness-of-fit D^+ or D^- test statistic based on continuous data for one sample.

AKS2DF Evaluate the distribution function of the Kolmogorov-Smirnov goodness-of-fit D test statistic based on continuous data for two samples.

KENDP Compute the frequency distribution of the total score in Kendall's rank correlation coefficient.

L5a1n ...Negative binomial, normal

ANORDF Evaluate the standard normal (Gaussian) distribution function.

L5a1p ...Pareto, Poisson

POIDF Evaluate the Poisson distribution function.

POIPR Evaluate the Poisson probability function.

L5a1t ....*t* distribution

TDF Evaluate the Student's *t* distribution function.

TNDF Evaluate the noncentral Student's *t* distribution function.

L5a2 .....Inverse cumulative distribution functions, sparsity functions

L5a2b ... Beta, binomial

`BETIN` Evaluate the inverse of the beta distribution function.

L5a2c ... Cauchy, chi-squared

`CHIIN` Evaluate the inverse of the chi-squared distribution function.

`CSNIN` Evaluate the inverse of the noncentral chi-squared function.

L5a2f .... *F* distribution

`FIN` Evaluate the inverse of the *F* distribution function.

L5a2g ... Gamma, general, geometric

`GAMIN` Evaluate the inverse of the gamma distribution function.

`GCIN` Evaluate the inverse of a general continuous cumulative distribution function given ordinates of the density.

`GFNIN` Evaluate the inverse of a general continuous cumulative distribution function given in a subprogram.

L5a2t .... *t* distribution

`TIN` Evaluate the inverse of the Student's *t* distribution function.

`TNIN` Evaluate the inverse of the noncentral Student's *t* distribution function.

L5b ..... Multivariate

L5b1 ..... Cumulative distribution functions, probability density functions

L5b1n ... Normal

`BNRDF` Evaluate the bivariate normal distribution function.

L6 ..... Random number generation

L6a ..... Univariate

L6a2 ..... Beta, binomial, Boolean

`RNBET` Generate pseudorandom numbers from a beta distribution.

`RNBIN` Generate pseudorandom numbers from a binomial distribution.

L6a3 ..... Cauchy, chi-squared

`RNCHI` Generate pseudorandom numbers from a chi-squared distribution.

`RNCHY` Generate pseudorandom numbers from a Cauchy distribution.

L6a5 ..... Exponential, extreme value

`RNEXP` Generate pseudorandom numbers from a standard exponential distribution.

`RNEXT` Generate pseudorandom numbers from a mixture of two exponential distributions.

L6a7 ..... Gamma, general (continuous, discrete), geometric

RNGAM Generate pseudorandom numbers from a standard gamma distribution.
RNGCS Set up table to generate pseudorandom numbers from a general continuous distribution.
RNGCT Generate pseudorandom numbers from a general continuous distribution.
RNGDA Generate pseudorandom numbers from a general discrete distribution using an alias method.
RNGDS Set up table to generate pseudorandom numbers from a general discrete distribution.
RNGDT Generate pseudorandom numbers from a general discrete distribution using a table lookup method.
RNGEO Generate pseudorandom numbers from a geometric distribution.

L6a8 .....Halfnormal, hypergeometric

RNHYP Generate pseudorandom numbers from a hypergeometric distribution.

L6a12 ...Lambda, logistic, lognormal

RNLGR Generate pseudorandom numbers from a logarithmic distribution.
RNLNL Generate pseudorandom numbers from a lognormal distribution.

L6a14 ...Negative binomial, normal, normal order statistics

RNNBN Generate pseudorandom numbers from a negative binomial distribution.
RNNOA Generate pseudorandom numbers from a standard normal distribution using an acceptance/rejection method.
RNNOF Generate a pseudorandom number from a standard normal distribution.
RNNOR Generate pseudorandom numbers from a standard normal distribution using an inverse CDF method.
RNNOS Generate pseudorandom order statistics from a standard normal distribution.

L6a16 ...Pareto, Pascal, permutations, Poisson

RNNPP Generate pseudorandom numbers from a nonhomogeneous Poisson process.

RNPER Generate a pseudorandom permutation.
RNPOI Generate pseudorandom numbers from a Poisson distribution.

L6a19 ...Samples, stable distribution

RNSRI Generate a simple pseudorandom sample of indices.
RNSRS Generate a simple pseudorandom sample from a finite population.
RNSTA Generate pseudorandom numbers from a stable distribution.

L6a20 ... *t* distribution, time series, triangular

`RNARM` Generate a time series from a specified ARMA model.

`RNNPP` Generate pseudorandom numbers from a nonhomogeneous Poisson process.

`RNSTT` Generate pseudorandom numbers from a Student's *t* distribution.

`RNTRI` Generate pseudorandom numbers from a triangular distribution on the interval (0,1).

L6a21 ... Uniform (continuous, discrete), uniform order statistics

`RNUN` Generate pseudorandom numbers from a uniform (0,1) distribution.

`RNUND` Generate pseudorandom numbers from a discrete uniform distribution.

`RNUNF` Generate a pseudorandom number from a uniform (0, 1) distribution.

`RNUNO` Generate pseudorandom order statistics from a uniform (0, 1) distribution.

L6a22 ... Von Mises

`RNVMS` Generate pseudorandom numbers from a von Mises distribution.

L6a23 ... Weibull

`RNWIB` Generate pseudorandom numbers from a Weibull distribution.

L6b ..... Multivariate

`RNDAT` Generate pseudorandom numbers from a multivariate distribution determined from a given sample.

L6b3 ..... Contingency table, correlation matrix

`RNCOR` Generate a pseudorandom orthogonal matrix or a correlation matrix.

`RNTAB` Generate a pseudorandom two-way table.

L6b13 ... Multinomial

`RNMTN` Generate pseudorandom numbers from a multinomial distribution.

L6b14 ... Normal

`RNMVN` Generate pseudorandom numbers from a multivariate normal distribution.

L6b15 ... Orthogonal matrix

`RNCOR` Generate a pseudorandom orthogonal matrix or a correlation matrix.

L6b21 ... Linear L-1 (least absolute value) approximation random numbers

`FAURE_INIT` Shuffles Faure sequence initialization.

`FAURE_FREE` Frees the structure containing information about the Faure sequence.

FAURE_NEXT Computes a shuffled Faure sequence.

L6b21 ...Uniform

RNSPH Generate pseudorandom points on a unit circle or K-dimensional sphere.

L6c .....Service routines (e.g., seed)

RNGEF Retrieve the current value of the array used in the IMSL GFSR random number generator.

RNGES Retrieve the current value of the table in the IMSL random number generators that use shuffling.

RNGET Retrieve the current value of the seed used in the IMSL random number generators.

RNISD Determine a seed that yields a stream beginning 100,000 numbers beyond the beginning of the stream yielded by a given seed used in IMSL multiplicative congruential generators (with no shufflings).

RNOPG Retrieve the indicator of the type of uniform random number generator.

RNOPT Select the uniform (0, 1) multiplicative congruential pseudorandom number generator.

RNSEF Initialize the array used in the IMSL GFSR random number generator.

RNSES Initialize the table in the IMSL random number generators that use shuffling.

RNSET Initialize a random seed for use in the IMSL randomnumber generators.

L7 .....Analysis of variance (including analysis of covariance)

L7a .....One-way

L7a1 .....Parametric

AONEC Analyze a one-way classification model with covariates.

AONEW Analyze a one-way classification model.

CTRST Compute contrast estimates and sums of squares.

SCIPM Compute simultaneous confidence intervals on all pairwise differences of means.

SNKMC Perform Student-Newman-Keuls multiple comparison test.

L7b .....Two-way (*search also class L7d*)

ATWOB Analyze a randomized block design or a two-way balanced design.

FRDMN Perform Friedman's test for a randomized complete block design.

MEDPL Compute a median polish of a two-way table.

L7c .....Three-way (e.g., Latin squares) (*search also class L7d*)

ALATN Analyze a Latin square design.

L7d .....Multi-way

L7d1 ..... Balanced complete data (e.g., factorial designs)

- `ABALD` Analyze a balanced complete experimental design for a fixed, random, or mixed model.
- `ANEST` Analyze a completely nested random model with possibly unequal numbers in the subgroups.
- `ANWAY` Analyze a balanced *n*-way classification model with fixed effects.
- `CIDMS` Compute a confidence interval on a variance component estimated as proportional to the difference in two mean squares in a balanced complete experimental design.
- `ROREX` Reorder the responses from a balanced complete experimental design.

L7d2 ..... Balanced incomplete data (e.g., fractional factorial designs)

- `ABIBD` Analyze a balanced incomplete block design or a balanced lattice design.

L7d3 ..... General linear models (unbalanced data)

- `ANEST` Analyze a completely nested random model with possibly unequal numbers in the subgroups.
- `RGLM` Fit a multivariate general linear model.

L7e ..... Multivariate

- `RGLM` Fit a multivariate general linear model.

L7f..... Generate experimental designs

- `RCOMP` Generate an orthogonal central composite design.

L8 ..... Regression (*search also classes D5, D6, D9, G, K*)

L8a ..... Simple linear (e.g., $y = \beta_0 + \beta_1 x + \varepsilon$)

L8a1 ..... Ordinary least squares

- `RONE` Analyze a simple linear regression model.

L8a1a ... Parameter estimation

L8a1a1 . Unweighted data

- `RLINE` Fit a line to a set of data points using least squares.

L8a1d ... Inference (e.g., calibration) (*search also class L8a1a*)

- `RINCF` Perform response control given a fitted simple linear regression model.
- `RINPF` Perform inverse prediction given a fitted simple linear regression model.

L8a2 ..... L_p for p different from 2 (e.g., least absolute value, minimax)

- `RLAV` Fit a multiple linear regression model using the least absolute values criterion.
- `RLLP` Fit a multiple linear regression model using the L_p norm criterion.
- `RLMV` Fit a multiple linear regression model using the minimax criterion.

L8b .....Polynomial (e.g., $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon$) (*search also class L8c*)

L8b1 .....Ordinary least squares

L8b1a ...Degree determination

RFORP Fit an orthogonal polynomial regression model.
RPOLY Analyze a polynomial regression model.

L8b1b ...Parameter estimation

L8b1b2 .Using orthogonal polynomials

RCURV Fit a polynomial curve using least squares.
RFORP Fit an orthogonal polynomial regression model.
RPOLY Analyze a polynomial regression model.

L8b1c ...Analysis (*search also class L8b1b*)

RCASP Compute case statistics for a polynomial regression model given the fit based on orthogonal polynomials.
RPOLY Analyze a polynomial regression model.
RSTAP Compute summary statistics for a polynomial regression model given the fit based on orthogonal polynomials.

L8b1d ...Inference (*search also class L8b1b*)

RCASP Compute case statistics for a polynomial regression model given the fit based on orthogonal polynomials.
RPOLY Analyze a polynomial regression model.
RSTAP Compute summary statistics for a polynomial regression model given the fit based on orthogonal polynomials.

L8c .....Multiple linear (e.g., $y = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k + \varepsilon$)

L8c1 .....Ordinary least squares

L8c1a....Variable selection

L8c1a2..Using correlation or covariance data

GSWEP Perform a generalized sweep of a row of a nonnegative definite matrix.
RBEST Select the best multiple linear regression models.
RSTEP Build multiple linear regression models using forward selection, backward selection, or stepwise selection.

L8c1b ...Parameter estimation (*search also class L8c1a*)

L8c1b1 .Using raw data

RGIVN Fit a multivariate linear regression model via fast Givens transformations.
RGLM Fit a multivariate general linear model.
RLSE Fit a multiple linear regression model using least squares.

L8c1b2 . Using correlation data

RCOV Fit a multiple linear regression model given the variance-covariance matrix.

L8c1c ... Analysis (*search also classes L8c1a and L8c1b*)

RCASE Compute case statistics and diagnostics given data points, coefficient estimates $\hat{\beta}$, and the *R* matrix for a fitted general linear model.

RCOVB Compute the estimated variance-covariance matrix of the estimated regression coefficients given the *R* matrix.

RLOFE Compute a lack-of-fit test based on exact replicates for a fitted regression model.

RLOFN Compute a lack-of-fit test based on near replicates for a fitted regression model.

ROTIN Compute diagnostics for detection of outliers and influential data points given residuals and the *R* matrix for a fitted general linear model.

RSTAT Compute statistics related to a regression fit given the coefficient estimates $\hat{\beta}$ and the *R* matrix.

L8c1d ... Inference (*search also classes L8c1a and L8c1b*)

CESTI Construct an equivalent completely testable multivariate general linear hypothesis $HBU = G$ from a partially testable hypothesis $H_pBU = G_p$.

RCASE Compute case statistics and diagnostics given data points, coefficient estimates $\hat{\beta}$, and the *R* matrix for a fitted general linear model.

RHPSS Compute the matrix of sums of squares and crossproducts for the multivariate general linear hypothesis $HBU = G$ given the coefficient estimates $\hat{B}$ and the *R* matrix.

RHPTE Perform tests for a multivariate general linear hypothesis $HBU = G$ given the hypothesis sums of squares and crossproducts matrix S_H and the error sums of squares and crossproducts matrix S_E.

RSTAT Compute statistics related to a regression fit given the coefficient estimates $\hat{\beta}$ and the *R* matrix.

L8c3 ..... L_p for p different from 2

RLAV Fit a multiple linear regression model using the least absolute values criterion.

RLLP Fit a multiple linear regression model using the L_p norm criterion.

RLMV Fit a multiple linear regression model using the minimax criterion.

L8d ..... Polynomial in several variables

RCOMP Generate an orthogonal central composite design.

TCSCP Transform coefficients from a quadratic regression model generated from squares and crossproducts of centered variables to a model using uncentered variables.

L8e .....Nonlinear (i.e., $y = f(X; \theta) + \varepsilon$)

L8e1 .....Ordinary least squares

L8e1b ...Parameter estimation

RNLIN Fit a nonlinear regression model.

L8f.....Simultaneous (i.e., $Y = XB + \varepsilon$)

RCOV Fit a multiple linear regression model given the variance-covariance matrix.

RGIVN Fit a multivariate linear regression model via fast Givens transformations.

RGLM Fit a multivariate general linear model.

RHPSS Compute the matrix of sums of squares and crossproducts for the multivariate general linear hypothesis $HBU = G$ given the coefficient estimates $\hat{B}$ and the R matrix.

RHPTE Perform tests for a multivariate general linear hypothesis $HBU = G$ given the hypothesis sums of squares and crossproducts matrix S_H and the error sums of squares and crossproducts matrix S_E.

RLEQU Fit a multivariate linear regression model with linear equality restrictions $HB = G$ imposed on the regression parameters given results from IMSL routine RGIVN after IDO = 1 and IDO = 2 and prior to IDO = 3.

L8i .....Service routines (e.g., matrix manipulation for variable selection)

GCLAS Get the unique values of each classification variable.

GCSCP Generate centered variables, squares, and crossproducts.

GRGLM Generate regressors for a general linear model.

RORDM Reorder rows and columns of a symmetric matrix.

RSUBM Retrieve a symmetric submatrix from a symmetric matrix.

L9 .....Categorical data analysis

CTGLM Analyze categorical data using logistic, Probit, Poisson, and other generalized linear models.

CTRAN Perform generalized Mantel-Haenszel tests in a stratified contingency table.

L9a .....2-by-2 tables

CTTWO Perform a chi-squared analysis of a 2 by 2 contingency table.

L9b .....Two-way tables (*search also class L9d*)

CTCHI Perform a chi-squared analysis of a two-way contingency table.

CTEPR Compute Fisher's exact test probability and a hybrid approximation to the Fisher exact test probability for a contingency table using the network algorithm.
CTPRB Compute exact probabilities in a two-way contingency table.
CTRHO Estimate the bivariate normal correlation coefficient using a contingency table.
CTWLS Perform a generalized linear least squares analysis of transformed probabilities in a two-dimensional contingency table.
MEDPL Compute a median polish of a two-way table.
TWFRQ Tally observations into a two-way frequency table.

L9c ..... Log-linear model

CTASC Compute partial association statistics for log-linear models in a multidimensional contingency table.
CTLLN Compute model estimates and associated statistics for a hierarchical log-linear model.
CTPAR Compute model estimates and covariances in a fitted log-linear model.
CTSTP Build hierarchical log-linear models using forward selection, backward selection, or stepwise selection.
PRPFT Perform iterative proportional fitting of a contingency table using a loglinear model.

L9d ..... EDA (e.g., median polish)

MEDPL Compute a median polish of a two-way table.

L10 ..... Time series analysis (*search also class J*)

L10a ..... Univariate

L10a1 ... Transformations

L10a1b . Stationarity (*search also class L8a1*)

BCTR Perform a forward or an inverse Box-Cox (power) transformation.

L10a1c . Filters

L10a1c1 Difference (nonseasonal and seasonal)

DIFF Difference a time series.

L10a2 ... Time domain analysis

AUTO_FPE_MUL_AR Automatic selection and fitting of a multivariate autoregressive time series model using Akaike's Multivariate Final Prediction Error (MFPE) criteria.

AUTO_FPE_UNI_AR Automatic selection and fitting of a univariate autoregressive time series model using Akaike's Final Prediction Error (FPE) criteria.

AUTO_MUL_AR Automatic selection and fitting of a multivariate autoregressive time series model.

AUTO_UNI_AR Automatic selection and fitting of a univariate autoregressive time series model.

BAY_SEA Model allows for a decomposition of a time series into trend, seasonal, and an error component.

GARCH Computes estimates of the parameters of a GARCH(p,q) model.

MAX_ARMA Exact maximum likelihood estimation of the parameters in a univariate ARMA (auto-regressive, moving average) time series model.

L10a2a..Summary statistics

L10a2a1 Autocovariances and autocorrelations

ACF Compute the sample autocorrelation function of a stationary time series.

LOFCF Perform lack-of-fit test for a univariate time series or transfer function given the appropriate correlation function.

L10a2a2 Partial autocorrelations

PACF Compute the sample partial autocorrelation function of a stationary time series.

L10a2c..Autoregressive models

SPWF Compute the Wiener forecast operator for a stationary stochastic process.

L10a2d .ARMA and ARIMA models (including Box-Jenkins methods)

L10a2d2 Parameter estimation

ARMME Compute method of moments estimates of the autoregressive parameters of an ARMA model.

MAMME Compute method of moments estimates of the moving average parameters of an ARMA model.

NSLSE Compute least squares estimates of parameters for a nonseasonal ARMA model.

NSPE Compute preliminary estimates of the autoregressive and moving average parameters of an ARMA model.

L10a2d3 Forecasting

NSBJF Compute Box-Jenkins forecasts and their associated probability limits for a nonseasonal ARMA model.

L10a2e..State-space analysis (e.g., Kalman filtering)

KALMN Perform Kalman filtering and evaluate the likelihood function for the state-space model.

L10a3 ...Frequency domain analysis (*search also class J1*)

L10a3a . Spectral Analysis

ARMA_SPEC Calculates the rational power spectrum for an ARMA model.

L10a3a2 Periodogram analysis

PFFT Compute the periodogram of a stationary time series using a fast Fourier transform.

L10a3a3 Spectrum estimation using the periodogram

SSWD Estimate the nonnormalized spectral density of a stationary time series using a spectral window given the time series data.

SSWP Estimate the nonnormalized spectral density of a stationary time series using a spectral window given the periodogram.

SWED Estimation of the nonnormalized spectral density of a stationary time series based on specified periodogram weights given the time series data.

SWEP Estimation of the nonnormalized spectral density of a stationary time series based on specified periodogram weights given the periodogram.

L10a3a6 Spectral windows

DIRIC Compute the Dirichlet kernel.

FEJER Compute the Fejér kernel.

L10b ..... Two time series (*search also classes L10c and L10d*)

L10b2 ... Time domain analysis

L10b2a . Summary statistics (e.g., cross-correlations)

CCF Compute the sample cross-correlation function of two stationary time series.

L10b2b . Transfer function models

IRNSE Compute estimates of the impulse response weights and noise series of a univariate transfer function model.

TFPE Compute preliminary estimates of parameters for a univariate transfer function model.

L10b3 ... Frequency domain analysis (*search also class J1*)

L10b3a . Cross-spectral analysis

L10b3a3 Cross-spectrum estimation using the cross-periodogram

CSSWD Estimate the nonnormalized cross-spectral density of two stationary time series using a spectral window given the time series data.

CSSWP Estimate the nonnormalized cross-spectral density of two stationary time series using a spectral window given the spectral densities and cross periodogram.

CSWED Estimate the nonnormalized cross-spectral density of two stationary time series using a weighted cross periodogram given the time series data.

CSWEP Estimate the nonnormalized cross-spectral density of two stationary time series using a weighted cross periodogram given the spectral densities and cross periodogram.

L10c .....Multivariate time series (*search also classes J1, L3e3 and L10b*)

KALMN Perform Kalman filtering and evaluate the likelihood function for the state-space model.

OPT_DES Allows for multiple channels for both the controlled and manipulated variables

L10d .....Two multi-channel time series

MCCF Compute the multichannel cross-correlation function of two mutually stationary multichannel time series.

MLSE Compute least squares estimates of a linear regression model for a multichannel time series with a specified base channel.

MWFE Compute least squares estimates of the multichannel Wiener filter coefficients for two mutually stationary multichannel time series.

L11 .....Correlation analysis (*search also classes L4 and L13c*)

BSCAT Compute the biserial correlation coefficient for a dichotomous variable and a classification variable.

BSPBS Compute the biserial and point-biserial correlation coefficients for a dichotomous variable and a numerically measurable classification variable.

CORVC Compute the variance-covariance or correlation matrix.

COVPL Compute a pooled variance-covariance matrix from the observations.

CTRHO Estimate the bivariate normal correlation coefficient using a contingency table.

KENDP Compute the frequency distribution of the total score in Kendall's rank correlation coefficient.

PCORR Compute partial correlations or covariances from the covariance or correlation matrix.

RBCOV Compute a robust estimate of a covariance matrix and mean vector.

TETCC Categorize bivariate data and compute the tetrachoric correlation coefficient.

L12 .....Discriminant analysis

DMSCR Use Fisher's linear discriminant analysis method to reduce the number of variables.

DSCRM Perform a linear or a quadratic discriminant function analysis among several known groups.
NNBRD Perform a *k* nearest neighbor discrimination.

L13 ..... Covariance structures models

L13a ..... Factor analysis

FACTR Extract initial factor-loading estimates in factor analysis.
FCOEF Compute a matrix of factor score coefficients for input to the following IMSL routine (FSCOR).
FDOBL Compute a direct oblimin rotation of a factor-loading matrix.
FGCRF Compute direct oblique rotation according to a generalized fourth-degree polynomial criterion.
FHARR Compute an oblique rotation of an unrotated factor-loading matrix using the Harris-Kaiser method.
FIMAG Compute the image transformation matrix.
FOPCS Compute an orthogonal Procrustes rotation of a factor-loading matrix using a target matrix.
FPRMX Compute an oblique Promax or Procrustes rotation of a factor-loading matrix using a target matrix, including pivot and power vector options.
FRESI Compute commonalities and the standardized factor residual correlation matrix.
FROTA Compute an orthogonal rotation of a factor-loading matrix using a generalized orthomax criterion, including quartimax, varimax, and equamax rotations.
FRVAR Compute the factor structures and the variance explained by each factor.
FSCOR Compute a set of factor scores given the factor score coefficient matrix.

L13b ..... Principal components analysis

KPRIN Maximum likelihood or least-squares estimates for principle components from one or more matrices.
PRINC Compute principal components from a variance-covariance matrix or a correlation matrix.

L13c ..... Canonical correlation

CANCR Perform canonical correlation analysis from a data matrix.
CANVC Perform canonical correlation analysis from a variance-covariance matrix or a correlation matrix.

L14 ..... Cluster analysis

L14a ..... One-way

L14a1 ... Unconstrained

L14a1a . Nested

L14a1a1 Joining (e.g., single link)

CLINK Perform a hierarchical cluster analysis given a distance matrix.

L14a1b .Non-nested (e.g., *K* means)

KMEAN Perform a *K*-means (centroid) cluster analysis.

L14c .....Display

TREEP Print a binary tree.

L14d .....Service routines (e.g., compute distance matrix)

CDIST Compute a matrix of dissimilarities (or similarities) between the columns (or rows) of a matrix.

CNUMB Compute cluster membership for a hierarchical cluster tree.

L15 .....Life testing, survival analysis

ACTBL Produce population and cohort life tables.

HAZEZ Perform nonparametric hazard rate estimation using kernel functions. Easy-to-use version of the previous IMSL subroutine (HAZRD).

HAZRD Perform nonparametric hazard rate estimation using kernel functions and quasi-likelihoods.

HAZST Perform hazard rate estimation over a grid of points using a kernel function.

KAPMR Compute Kaplan-Meier estimates of survival probabilities in stratified samples.

KTBLE Print Kaplan-Meier estimates of survival probabilities in stratified samples.

NRCES Compute maximum likelihood estimates of the mean and variance from grouped and/or censored normal data.

PHGLM Analyze time event data via the proportional hazards model.

STBLE Estimate survival probabilities and hazard rates for various parametric models.

SVGLM Analyze censored survival data using a generalized linear model.

TRNBL Compute Turnbull's generalized Kaplan-Meier estimates of survival probabilities in samples with interval censoring.

L16 .....Multidimensional scaling

MSDBL Obtain normalized product-moment (double centered) matrices from dissimilarity matrices.

MSDST Compute distances in a multidimensional scaling model.

MSIDV Perform individual-differences multidimensional scaling for metric data using alternating least squares.

MSINI Compute initial estimates in multidimensional scaling models.

MSSTN Transform dissimilarity/similarity matrices and replace missing values by estimates to obtain standardized dissimilarity matrices.

MSTRS Compute various stress criteria in multidimensional scaling.

L17..... Statistical data sets

GDATA Retrieve a commonly analyzed data set.

N..... DATA HANDLING (*search also class L2*)

N1..... Input, output

PGOPT Set or retrieve page width and length for printing.

WRIRL Print an integer rectangular matrix with a given format and labels.

WRIRN Print an integer rectangular matrix with integer row and column labels.

WROPT Set or retrieve an option for printing a matrix.

WRRRL Print a real rectangular matrix with a given format and labels.

WRRRN Print a real rectangular matrix with integer row and column labels.

N3..... Character manipulation

ACHAR Return a character given its ASCII value.

CVTSI Convert a character string containing an integer number into the corresponding integer form.

IACHAR Return the integer ASCII value of a character argument.

ICASE Return the ASCII value of a character converted to uppercase.

IICSR Compare two character strings using the ASCII collating sequence without regard to case.

IIDEX Determine the position in a string at which a given character sequence begins without regard to case.

N5..... Searching

N5a..... Extreme value

EQTIL Compute empirical quantiles.

ORDST Determine order statistics.

N5b..... Insertion position

ISRCH Search a sorted integer vector for a given integer and return its index.

SRCH Search a sorted vector for a given scalar and return its index.

SSRCH Search a character vector, sorted in ascending ASCII order, for a given string and return its index.

N5c..... On a key

IIDEX Determine the position in a string at which a given character sequence begins without regard to case.

ISRCH Search a sorted integer vector for a given integer and return its index.

SRCH Search a sorted vector for a given scalar and return its index.

SSRCH Search a character vector, sorted in ascending ASCII order, for a given string and return its index.

N6.....Sorting

N6a.....Internal

N6a1.....Passive (i.e., construct pointer array, rank)

N6a1a...Integer

SVIGP Sort an integer array by algebraic value and return the permutations.

N6a1b...Real

RANKS Compute the ranks, normal scores, or exponential scores for a vector of observations.

SCOLR Sort columns of a real rectangular matrix using keys in rows.

SROWR Sort rows of a real rectangular matrix using keys in columns.

SVRGP Sort a real array by algebraic value and return the permutations.

N6a2.....Active

N6a2a...Integer

SVIGN Sort an integer array by algebraic value.

SVIGP Sort an integer array by algebraically increasing value and return the permutation that rearranges the array.

N6a2b...Real

SCOLR Sort columns of a real rectangular matrix using keys in rows.

SROWR Sort rows of a real rectangular matrix using keys in columns.

SVRGN Sort a real array by algebraic value.

SVRGP Sort a real array by algebraic value and return the permutations.

N8.....Permuting

MVNAN Move any rows of a matrix with the IMSL missing value code NaN (not a number) in the specified columns to the last rows of the matrix.

PERMA Permute the rows or columns of a matrix.

PERMU Rearrange the elements of an array as specified by a permutation.

RORDM Reorder rows and columns of a symmetric matrix.

Q.....GRAPHICS (*search also classes L3*)

BOXP Print boxplots for one or more samples.
CDF2P Print a plot of two sample cumulative distribution functions.
CDFP Print a sample cumulative distribution function (CDF), a theoretical CDF, and confidence band information.
HHSTP Print a horizontal histogram.
PLOTP Print a plot of up to ten sets of points.
PROBP Print a probability plot.
SCTP Print a scatterplot of several groups of data.
STMLP Print a stem-and-leaf plot.
TREEP Print a binary tree.
VHS2P Print a vertical histogram with every bar subdivided into two parts.
VHSTP Print a vertical histogram.

R..... SERVICE ROUTINES
IDYWK Compute the day of the week for a given date.
NDAYS Compute the number of days from January 1, 1900, to the given date.
NDYIN Give the date corresponding to the number of days since January 1, 1900.
TDATE Get today's date.
TIMDY Get time of day.
VERSL Obtain STAT/LIBRARY-related version, system and license numbers.

R1..... Machine-dependent constants
AMACH Retrieve machine constants.
IFNAN Check if a floating-point number is NaN (not a number).
IMACH Retrieve integer machine constants.
UMACH Set or retrieve input or output device unit numbers.

R3..... Error handling

R3b..... Set unit number for error messages
UMACH Set or retrieve input or output device unit numbers.

R3c..... Other utilities
ERSET Set error handler default print and stop actions.
IERCD Retrieve the code for an informational error.
N1RTY Retrieve an error type for the most recently called IMSL routine.

S ..... SOFTWARE DEVELOPMENT TOOLS
CPSEC Return CPU time used in seconds.

Appendix B: Alphabetical Summary of Routines

IMSL STAT/LIBRARY

ABALD	434	Analyzes a balanced complete experimental design for a fixed, random, or mixed model.
ABIBD	415	Analyzes a balanced incomplete block design or a balanced lattice design.
ACF	697	Computes the sample autocorrelation function of a stationary time series.
ACHAR	1458	Returns a character given its ASCII value.
ACTBL	1121	Produces population and cohort life tables.
AKS1DF	1257	Evaluates the distribution function of the one-sided Kolmogorov-Smirnov goodness-of-fit D^+ or D^- test statistic based on continuous data for one sample.
AKS2DF	1260	Evaluates the distribution function of the Kolmogorov-Smirnov goodness-of-fit D test statistic based on continuous data for two samples.
ALATN	422	Analyzes a Latin square design.
AMACH	1507	Retrieves machine constants.
AMILLR	1489	Evaluates Mill's ratio (the ratio of the ordinate to the upper tail area of the standardized normal distribution).
ANEST	449	Analyzes a completely nested random model with possibly unequal numbers in the subgroups.
ANORDF	1262	Evaluates the standard normal (Gaussian) distribution function.
ANORIN	1264	Evaluates the inverse of the standard normal (Gaussian) distribution function.
ANWAY	427	Analyzes a balanced n-way classification model with fixed effects.

AONEC	395	Analyzes a one-way classification model with covariates.
AONEW	392	Analyzes a one-way classification model.
ARMA_SPEC	822	Calculates the rational power spectrum for an ARMA model.
ARMME	720	Computes method of moments estimates of the autoregressive parameters of an ARMA model.
ATWOB	409	Analyzes a randomized block design or a two-way balanced design.
AUTO_FPE MUL_AR	801	Automatic selection and fitting of a multivariate autoregressive time series model using Akaike's Multivariate Final Prediction Error (MFPE) criteria.
AUTO_FPE UNI_AR	794	Automatic selection and fitting of a univariate autoregressive time series model using Akaike's Final Prediction Error (FPE) criteria.
AUTO_MUL_AR	797	Automatic selection and fitting of a multivariate autoregressive time series model.
AUTO_UNI_AR	791	Automatic selection and fitting of a multivariate autoregressive time series model.
BAY_SEA	803	Allows for a decomposition of a time series into trend, seasonal, and an error component.
BCTR	689	Performs a forward or an inverse Box-Cox (power) transformation.
BETDF	1265	Evaluates the beta probability distribution function.
BETIN	1268	Evaluates the inverse of the beta distribution function.
BHAKV	619	Performs a Bhapkar V test.
BINDF	1246	Evaluates the binomial distribution function.
BINES	48	Estimates the parameter p of the binomial distribution.
BINPR	1248	Evaluates the binomial probability function.
BNRDF	1270	Evaluates the bivariate normal distribution function.
BOXP	1219	Prints boxplots for one or more samples.
BSCAT	377	Computes the biserial correlation coefficient for a dichotomous variable and a classification variable.
BSPBS	374	Computes the biserial and point-biserial correlation coefficients for a dichotomous variable and a numerically measurable classification variable.

CANCR	957	Performs canonical correlation analysis from a data matrix.
CANVC	971	Performs canonical correlation analysis from a variance-covariance matrix or a correlation matrix.
CCF	706	Computes the sample cross-correlation function of two stationary time series.
CDF2P	1226	Prints a plot of two sample cumulative distribution functions.
CDFP	1223	Prints a sample cumulative distribution function (CDF), a theoretical CDF, and confidence band information.
CDIST	1006	Computes a matrix of dissimilarities (or similarities) between the columns (or rows) of a matrix.
CESTI	170	Constructs an equivalent completely testable multivariate general linear hypothesis $HBU = G$ from a partially testable hypothesis $H_pBU = G_p$.
CHFAC	1480	Computes an upper triangular factorization of a real symmetric nonnegative definite matrix.
CHIDF	1271	Evaluates the chi-squared distribution function.
CHIGF	638	Performs a chi-squared goodness-of-fit test.
CHIIN	1274	Evaluates the inverse of the chi-squared distribution function.
CIDMS	469	Computes a confidence interval on a variance component estimated as proportional to the difference in two mean squares in a balanced complete experimental design.
CLINK	1011	Performs a hierarchical cluster analysis given a distance matrix.
CNCRD	379	Calculates and tests the significance of the Kendall coefficient of concordance.
CNUMB	1015	Computes cluster membership for a hierarchical cluster tree.
CORVC	340	Computes the variance-covariance or correlation matrix.
COVPL	348	Computes a pooled variance-covariance matrix from the observations.
CPFFT	855	Computes the cross periodogram of two stationary time series using a fast Fourier transform.
CPSEC	1465	Returns CPU time used in seconds.
CSNDF	1275	Evaluates the noncentral chi-squared distribution function.

CSNIN	1278	Evaluates the inverse of the noncentral chi-squared function.
CSSWD	862	Estimates the nonnormalized cross-spectral density of two stationary time series using a spectral window given the time series data.
CSSWP	873	Estimates the nonnormalized cross-spectral density of two stationary time series using a spectral window given the spectral densities and cross periodogram.
CSTAT	60	Computes cell frequencies, cell means, and cell sums of squares for multivariate data.
CSWED	880	Estimates the nonnormalized cross-spectral density of two stationary time series using a weighted cross periodogram given the time series data.
CSWEP	889	Estimates the nonnormalized cross-spectral density of two stationary time series using a weighted cross periodogram given the spectral densities and cross periodogram.
CTASC	528	Computes partial association statistics for log-linear models in a multidimensional contingency table.
CTCHI	491	Performs a chi-squared analysis of a two-way contingency table.
CTEPR	504	Computes Fisher's exact test probability and a hybrid approximation to the Fisher exact test probability for a contingency table using the network algorithm.
CTGLM	557	Analyzes categorical data using logistic, Probit, Poisson, and other generalized linear models.
CTLLN	513	Computes model estimates and associated statistics for a hierarchical log-linear model.
CTPAR	522	Computes model estimates and covariances in a fitted log-linear model.
CTPRB	501	Computes exact probabilities in a two-way contingency table.
CTRAN	548	Performs generalized Mantel-Haenszel tests in a stratified contingency table.
CTRHO	367	Estimates the bivariate normal correlation coefficient using a contingency table.
CTRST	458	Computes contrast estimates and sums of squares.
CTSTP	536	Builds hierarchical log-linear models using forward selection, backward selection, or stepwise selection.

CTTWO	480	Performs a chi-squared analysis of a 2 by 2 contingency table.
CTWLS	574	Performs a generalized linear least squares analysis of transformed probabilities in a two-dimensional contingency table.
CVTSI	1464	Converts a character string containing an integer number into the corresponding integer form.
DCUBE	667	Performs a triplets test.
DESKN	1176	Performs nonparametric probability density function estimation by the kernel method.
DESPL	1172	Performs nonparametric probability density function estimation by the penalized likelihood method.
DESPT	1185	Estimates a probability density function at specified points using linear or cubic interpolation.
DIFF	693	Difference a time series.
DIRIC	818	Computes the Dirichlet kernel.
DMACH	1509	See AMACH.
DMSCR	993	Uses Fisher's linear discriminant analysis method to reduce the number of variables.
DNFFT	1180	Computes Gaussian kernel estimates of a univariate density via the fast Fourier transform over a fixed interval.
DSCRM	979	Performs a linear or a quadratic discriminant function analysis among several known groups.
DSQAR	664	Performs a D-square test.
ENOS	1487	Evaluates the expected value of a normal order statistic.
EQTIL	38	Computes empirical quantiles.
ERSET	1501	Sets error handler default print and stop actions.
FACTR	909	Extracts initial factor-loading estimates in factor analysis.
FAURE_FREE	1405	Frees the structure containing information about the Faure sequence.
FAURE_INIT	1405	Computes a shuffled Faure sequence.
FAURE_NEXT	1406	Shuffled Faure sequence initialization.
FCOEF	944	Computes a matrix of factor score coefficients for input to the following IMSL routine (FSCOR).
FDF	1280	Evaluates the *F* distribution function.

FDOBL	924	Computes a direct oblimin rotation of a factor-loading matrix.
FEJER	820	Computes the Fejér kernel.
FGCRF	936	Computes direct oblique rotation according to a generalized fourth-degree polynomial criterion.
FHARR	932	Computes an oblique rotation of an unrotated factor-loading matrix using the Harris-Kaiser method.
FIMAG	940	Computes the image transformation matrix.
FIN	1282	Evaluates the inverse of the *F* distribution function.
FOPCS	921	Computes an orthogonal Procrustes rotation of a factor-loading matrix using a target matrix.
FPRMX	927	Computes an oblique Promax or Procrustes rotation of a factor-loading matrix using a target matrix, including pivot and power vector options.
FRDMN	620	Performs Friedman's test for a randomized complete block design.
FREQ	14	Tallies multivariate observations into a multi-way frequency table.
FRESI	952	Computes commonalities and the standardized factor residual correlation matrix.
FROTA	918	Computes an orthogonal rotation of a factor-loading matrix using a generalized orthomax criterion, including quartimax, varimax, and equamax rotations.
FRVAR	942	Computes the factor structures and the variance explained by each factor.
FSCOR	949	Computes a set of factor scores given the factor score coefficient matrix.
GAMDF	1284	Evaluates the gamma distribution function.
GAMIN	1286	Evaluates the inverse of the gamma distribution function.
GARCH	745	Computes estimates of the parameters of a GARCH(p,q) model.
GCDF	1295	Evaluates a general continuous cumulative distribution function given ordinates of the density.
GCIN	1298	Evaluates the inverse of a general continuous cumulative distribution function given ordinates of the density.
GCLAS	222	Gets the unique values of each classification variable.
GCSCP	296	Generates centered variables, squares, and crossproducts.
GDATA	1474	Retrieves a commonly analyzed data set.

GFNIN	1301	Evaluates the inverse of a general continuous cumulative distribution function given in a subprogram.
GIRTS	1477	Solves a triangular (possibly singular) set of linear systems and/or compute a generalized inverse of an upper triangular matrix.
GRGLM	225	Generates regressors for a general linear model.
GRPES	56	Computes basic statistics from grouped data.
GSWEP	247	Performs a generalized sweep of a row of a nonnegative definite matrix.
HAZEZ	1195	Performs nonparametric hazard rate estimation using kernel functions. Easy-to-use version of the previous IMSL subroutine (HAZRD).
HAZRD	1188	Performs nonparametric hazard rate estimation using kernel functions and quasi-likelihoods.
HAZST	1203	Performs hazard rate estimation over a grid of points using a kernel function.
HHSTP	1213	Prints a horizontal histogram.
HYPDF	1250	Evaluates the hypergeometric distribution function.
HYPPR	1252	Evaluates the hypergeometric probability function.
IACHAR	1459	Returns the integer ASCII value of a character argument.
ICASE	1460	Returns the ASCII value of a character converted to uppercase.
IDYWK	1471	Computes the day of the week for a given date.
IERCD	1502	Retrieves the code for an informational error.
IFNAN	1509	Checks if a floating-point number is NaN (not a number).
IICSR	1462	Compares two character strings using the ASCII collating sequence without regard to case.
IIDEX	1463	Determines the position in a string at which a given character sequence begins without regard to case.
IMACH	1506	Retrieves integer machine constants.
INCLD	613	Performs an includance test.
IRNSE	758	Computes estimates of the impulse response weights and noise series of a univariate transfer function model.
ISRCH	1454	Searches a sorted integer vector for a given integer and returns its index.
KALMN	780	Performs Kalman filtering and evaluate the likelihood function for the state-space model.

KAPMR	1063	Computes Kaplan-Meier estimates of survival probabilities in stratified samples.
KENDL	383	Computes and tests Kendall's rank correlation coefficient.
KENDP	387	Computes the frequency distribution of the total score in Kendall's rank correlation coefficient.
KMEAN	1019	Performs a *K*-means (centroid) cluster analysis.
KPRIN	905	Maximum likelihood or least-squares estimates for principle components from one or more matrices.
KRSKL	616	Performs a Kruskal-Wallis test for identical population medians.
KSONE	634	Performs a Kolmogorov-Smirnov one-sample test for continuous distributions.
KSTWO	654	Performs a Kolmogorov-Smirnov two-sample test.
KTBLE	1068	Prints Kaplan-Meier estimates of survival probabilities in stratified samples.
KTRND	627	Performs a *k*-sample trends test against ordered alternatives.
LETTR	32	Produces a letter value summary.
LILLF	646	Performs Lilliefors test for an exponential or normal distribution.
LOFCF	815	Performs lack-of-fit test for a univariate time series or transfers function given the appropriate correlation function.
MAMME	**723**	Computes method of moments estimates of the moving average parameters of an ARMA model.
MAX_ARMA	741	Exacts maximum likelihood estimation of the parameters in a univariate ARMA (auto-regressive, moving average) time series model.
MCCF	711	Computes the multichannel cross-correlation function of two mutually stationary multichannel time series.
MCHOL	1483	Computes an upper triangular factorization of a real symmetric matrix A plus a diagonal matrix D, where D is determined sequentially during the Cholesky factorization in order to make $A + D$ nonnegative definite.
MEDPL	60	Computes a median polish of a two-way table.
MLSE	768	Computes least squares estimates of a linear regression model for a multichannel time series with a specified base channel.

MSDBL	1154	Obtains normalized product-moment (double centered) matrices from dissimilarity matrices.
MSDST	1146	Computes distances in a multidimensional scaling model.
MSIDV	1131	Performs individual-differences multidimensional scaling for metric data using alternating least squares.
MSINI	1159	Computes initial estimates in multidimensional scaling models.
MSSTN	1150	Transforms dissimilarity/similarity matrices and replace missing values by estimates to obtain standardized dissimilarity matrices.
MSTRS	1166	Computes various stress criteria in multidimensional scaling.
MVIND	954	Computes a test for the independence of *k* sets of multivariate normal variables.
MVMMT	649	Computes Mardia's multivariate measures of skewness and kurtosis and tests for multivariate normality.
MVNAN	1435	Moves any rows of a matrix with the IMSL missing value code NaN (not a number) in the specified columns to the last rows of the matrix.
MWFE	774	Computes least squares estimates of the multichannel Wiener filter coefficients for two mutually stationary multichannel time series.
N1RTY	1502	Retrieves an error type for the most recently called IMSL routine.
NCTRD	599	Performs the Noether test for cyclical trend.
NDAYS	1468	Computes the number of days from January 1, 1900, to the given date.
NDYIN	1470	Gives the date corresponding to the number of days since January 1, 1900.
NGHBR	1494	Searches a *k-d* tree for the *k* nearest neighbors of a key.
NNBRD	997	Performs a *k* nearest neighbor discrimination.
NRCES	53	Computes maximum likelihood estimates of the mean and variance from grouped and/or censored normal data.
NSBJF	752	Computes Box-Jenkins forecasts and their associated probability limits for a nonseasonal ARMA model.
NSLSE	733	Computes least squares estimates of parameters for a nonseasonal ARMA model.
NSPE	727	Computes preliminary estimates of the autoregressive and moving average parameters of an ARMA model.

NTIES	607	Computes tie statistics for a sample of observations.
OPOLY	292	Generates orthogonal polynomials with respect to *x* values and specified weights.
OPT_DES	810	Allows for multiple channels for both the controlled and manipulated variables
ORDST	34	Determines order statistics.
OWFRQ	3	Tallies observations into a one-way frequency table.
PACF	702	Computes the sample partial autocorrelation function of a stationary time series.
PAIRS	661	Performs a pairs test.
PCORR	353	Computes partial correlations or covariances from the covariance or correlation matrix.
PERMA	1431	Permutes the rows or columns of a matrix.
PERMU	1429	Rearranges the elements of an array as specified by a permutation.
PFFT	825	Computes the periodogram of a stationary time series using a fast Fourier transform.
PGOPT	1428	Sets or retrieves page width and length for printing.
PHGLM	1078	Analyzes time event data via the proportional hazards model.
PLOTP	1232	Prints a plot of up to ten sets of points.
POIDF	1254	Evaluates the Poisson distribution function.
POIES	50	Estimates the parameter of the Poisson distribution.
POIPR	1255	Evaluates the Poisson probability function.
PRINC	901	Computes principal components from a variance-covariance matrix or a correlation matrix.
PROBP	1229	Prints a probability plot.
PRPFT	509	Performs iterative proportional fitting of a contingency table using a loglinear model.
QTEST	625	Performs a Cochran *Q* test for related observations.
QUADT	1490	Forms a *k-d* tree.
RANKS	26	Computes the ranks, normal scores, or exponential scores for a vector of observations.
RBCOV	358	Computes a robust estimate of a covariance matrix and mean vector.
RBEST	231	Selects the best multiple linear regression models.

RCASE	204	Computes case statistics and diagnostics given data points, coefficient estimates $\hat{\beta}$, and the R matrix for a fitted general linear model.
RCASP	285	Computes case statistics for a polynomial regression model given the fit based on orthogonal polynomials.
RCOMP	268	Generates an orthogonal central composite design.
RCOV	113	Fits a multiple linear regression model given the variance-covariance matrix.
RCOVB	164	Computes the estimated variance-covariance matrix of the estimated regression coefficients given the R matrix.
RCURV	255	Fits a polynomial curve using least squares.
RFORP	272	Fits an orthogonal polynomial regression model.
RGIVN	117	Fits a multivariate linear regression model via fast Givens transformations.
RGLM	128	Fits a multivariate general linear model.
RHPSS	176	Computes the matrix of sums of squares and crossproducts for the multivariate general linear hypothesis $HBU = G$ given the coefficient estimates $\hat{B}$ and the R matrix.
RHPTE	183	Performs tests for a multivariate general linear hypothesis $HBU = G$ given the hypothesis sums of squares and crossproducts matrix S_H and the error sums of squares and crossproducts matrix S_E.
RINCF	99	Performs response control given a fitted simple linear regression model.
RINPF	103	Performs inverse prediction given a fitted simple linear regression model.
RLAV	317	Fits a multiple linear regression model using the least absolute values criterion.
RLEQU	142	Fits a multivariate linear regression model with linear equality restrictions $HB = G$ imposed on the regression parameters given results from IMSL routine `RGIVN` after `IDO` = 1 and `IDO` = 2 and prior to `IDO` = 3.
RLEQU	131	Fits a multivariate linear regression model with linear equality restrictions $HB = G$ imposed on the regression parameters given results from IMSL routine `RGIVN` after `IDO` = 1 and `IDO` = 2 and prior to `IDO` = 3.
RLINE	86	Fits a line to a set of data points using least squares.

RLLP	322	Fits a multiple linear regression model using the L_p norm criterion.
RLMV	333	Fits a multiple linear regression model using the minimax criterion.
RLOFE	189	Computes a lack-of-fit test based on exact replicates for a fitted regression model.
RLOFN	195	Computes a lack-of-fit test based on near replicates for a fitted regression model.
RLSE	107	Fits a multiple linear regression model using least squares.
RNARM	1390	Generates a time series from a specified ARMA model.
RNBET	1341	Generates pseudorandom numbers from a beta distribution.
RNBIN	1319	Generates pseudorandom numbers from a binomial distribution.
RNCHI	1343	Generates pseudorandom numbers from a chi-squared distribution.
RNCHY	1344	Generates pseudorandom numbers from a Cauchy distribution.
RNCOR	1370	Generates a pseudorandom orthogonal matrix or a correlation matrix.
RNDAT	1373	Generates pseudorandom numbers from a multivariate distribution determined from a given sample.
RNEXP	1346	Generates pseudorandom numbers from a standard exponential distribution.
RNEXT	1347	Generates pseudorandom numbers from a mixture of two exponential distributions.
RNGAM	1349	Generates pseudorandom numbers from a standard gamma distribution.
RNGCS	1351	Sets up table to generate pseudorandom numbers from a general continuous distribution.
RNGCT	1354	Generates pseudorandom numbers from a general continuous distribution.
RNGDA	1321	Generates pseudorandom numbers from a general discrete distribution using an alias method.
RNGDS	1324	Sets up table to generate pseudorandom numbers from a general discrete distribution.
RNGDT	1328	Generates pseudorandom numbers from a general discrete distribution using a table lookup method.

RNGEF	1312	Retrieves the current value of the array used in the IMSL GFSR random number generator.
RNGEO	1331	Generates pseudorandom numbers from a geometric distribution.
RNGES	1312	Retrieves the current value of the table in the IMSL random number generators that use shuffling.
RNGET	1312	Retrieves the current value of the seed used in the IMSL random number generators.
RNHYP	1333	Generates pseudorandom numbers from a hypergeometric distribution.
RNISD	1315	Determines a seed that yields a stream beginning 100,000 numbers beyond the beginning of the stream yielded by a given seed used in IMSL multiplicative congruential generators (with no shufflings).
RNKSM	609	Performs the Wilcoxon rank sum test.
RNLGR	1335	Generates pseudorandom numbers from a logarithmic distribution.
RNLIN	304	Fits a nonlinear regression model.
RNLNL	1356	Generates pseudorandom numbers from a lognormal distribution.
RNMTN	1377	Generates pseudorandom numbers from a multinomial distribution.
RNMVN	1379	Generates pseudorandom numbers from a multivariate normal distribution.
RNNBN	1336	Generates pseudorandom numbers from a negative binomial distribution.
RNNOA	1358	Generates pseudorandom numbers from a standard normal distribution using an acceptance/rejection method.
RNNOF	1359	Generates a pseudorandom number from a standard normal distribution.
RNNOR	1361	Generates pseudorandom numbers from a standard normal distribution using an inverse CDF method.
RNNOS	1386	Generates pseudorandom order statistics from a standard normal distribution.
RNNPP	1394	Generates pseudorandom numbers from a nonhomogeneous Poisson process.
RNOPG	1311	Retrieves the indicator of the type of uniform random number generator.

RNOPT	1311	Selects the uniform (0, 1) multiplicative congruential pseudorandom number generator.
RNPER	1398	Generates a pseudorandom permutation.
RNPOI	1338	Generates pseudorandom numbers from a Poisson distribution.
RNSEF	1167	Initializes the array used in the IMSL GFSR random number generator.
RNSES	1167	Initializes the table in the IMSL random number generators that use shuffling.
RNSET	1167	Initializes a random seed for use in the IMSL randomnumber generators.
RNSPH	1381	Generates pseudorandom points on a unit circle or K-dimensional sphere.
RNSRI	1399	Generates a simple pseudorandom sample of indices.
RNSRS	1401	Generates a simple pseudorandom sample from a finite population.
RNSTA	1362	Generates pseudorandom numbers from a stable distribution.
RNSTT	1364	Generates pseudorandom numbers from a Student's *t* distribution.
RNTAB	1383	Generates a pseudorandom two-way table.
RNTRI	1366	Generates pseudorandom numbers from a triangular distribution on the interval (0,1).
RNUN	1316	Generates pseudorandom numbers from a uniform (0,1) distribution.
RNUND	1340	Generates pseudorandom numbers from a discrete uniform distribution.
RNUNF	1317	Generates a pseudorandom number from a uniform (0, 1) distribution.
RNUNO	1388	Generates pseudorandom order statistics from a uniform (0, 1) distribution.
RNVMS	1367	Generates pseudorandom numbers from a von Mises distribution.
RNWIB	1368	Generates pseudorandom numbers from a Weibull distribution.
RONE	89	Analyzes a simple linear regression model.
RORDM	1433	Reorders rows and columns of a symmetric matrix.

ROREX	472	Reorders the responses from a balanced complete experimental design.
ROTIN	215	Computes diagnostics for detection of outliers and influential data points given residuals and the *R* matrix for a fitted general linear model.
RPOLY	260	Analyzes a polynomial regression model.
RSTAP	279	Computes summary statistics for a polynomial regression model given the fit based on orthogonal polynomials.
RSTAT	152	Computes statistics related to a regression fit given the coefficient estimates $\hat{\beta}$ and the *R* matrix.
RSTEP	237	Builds multiple linear regression models using forward selection, backward selection, or stepwise selection.
RSUBM	251	Retrieves a symmetric submatrix from a symmetric matrix.
RUNS	657	Performs a runs up test.
SCIPM	460	Computes simultaneous confidence intervals on all pairwise differences of means.
SCOLR	1444	Sorts columns of a real rectangular matrix using keys in rows.
SCTP	1216	Prints a scatterplot of several groups of data.
SDPLC	602	Performs the Cox and Stuart sign test for trends in dispersion and location.
SIGNT	592	Performs a sign test of the hypothesis that a given value is a specified quantile of a distribution.
SMPPR	1026	Computes statistics for inferences regarding the population proportion and total, given proportion data from a simple random sample.
SMPPS	1029	Computes statistics for inferences regarding the population proportion and total, given proportion data from a stratified random sample.
SMPRR	1032	Computes statistics for inferences regarding the population mean and total using ratio or regression estimation, or inferences regarding the population ratio, given a simple random sample.
SMPRS	1039	Computes statistics for inferences regarding the population mean and total using ratio or regression estimation, given continuous data from a stratified random sample.

SMPSC	1046	Computes statistics for inferences regarding the population mean and total using single-stage cluster sampling with continuous data.
SMPSR	1050	Computes statistics for inferences regarding the population mean and total, given data from a simple random sample.
SMPSS	1054	Computes statistics for inferences regarding the population mean and total, given data from a stratified random sample.
SMPST	1058	Computes statistics for inferences regarding the population mean and total, given continuous data from a two-stage sample with equisized primary units.
SNKMC	467	Performs Student-Newman-Keuls multiple comparison test.
SNRNK	595	Performs a Wilcoxon signed rank test.
SPWF	749	Computes the Wiener forecast operator for a stationary stochastic process.
SPWLK	644	Performs a Shapiro-Wilk *W*-test for normality.
SRCH	1452	Searches a sorted vector for a given scalar and return its index.
SROWR	1448	Sorts rows of a real rectangular matrix using keys in columns.
SSRCH	1456	Searches a character vector, sorted in ascending ASCII order, for a given string and return its index.
SSWD	831	Estimates the nonnormalized spectral density of a stationary time series using a spectral window given the time series data.
SSWP	840	Estimates the nonnormalized spectral density of a stationary time series using a spectral window given the periodogram.
STBLE	1114	Estimates survival probabilities and hazard rates for various parametric models.
STMLP	1221	Prints a stem-and-leaf plot.
SVGLM	1095	Analyzes censored survival data using a generalized linear model.
SVIGN	1442	Sorts an integer array by algebraic value.
SVIGP	1443	Sorts an integer array by algebraic value and returns the permutations.
SVRGN	1439	Sorts a real array by algebraic value.

SVRGP	1440	Sorts a real array by algebraic value and returns the permutations.
SWED	845	Estimation of the nonnormalized spectral density of a stationary time series based on specified periodogram weights given the time series data.
SWEP	851	Estimation of the nonnormalized spectral density of a stationary time series based on specified periodogram weights given the periodogram.
TCSCP	301	Transforms coefficients from a quadratic regression model generated from squares and crossproducts of centered variables to a model using uncentered variables.
TDATE	1467	Gets today's date.
TDF	1288	Evaluates the Student's *t* distribution function.
TETCC	370	Categorizes bivariate data and compute the tetrachoric correlation coefficient.
TFPE	762	Computes preliminary estimates of parameters for a univariate transfer function model.
TIMDY	1466	Gets time of day.
TIN	1289	Evaluates the inverse of the Student's *t* distribution function.
TNDF	1291	Evaluates the noncentral Student's *t* distribution function.
TNIN	1293	Evaluates the inverse of the noncentral Student's *t* distribution function.
TREEP	1236	Prints a binary tree.
TRNBL	1073	Computes Turnbull's generalized Kaplan-Meier estimates of survival probabilities in samples with interval censoring.
TWFRQ	7	Tallies observations into a two-way frequency table.
TWOMV	41	Computes statistics for mean and variance inferences using samples from two normal populations.
UMACH	1510	Sets or retrieves input or output device unit numbers.
UVSTA	18	Computes basic univariate statistics.
VERSL	1472	Obtains STAT/LIBRARY-related version, system and license numbers.
VHS2P	1210	Prints a vertical histogram with every bar subdivided into two parts.
VHSTP	1208	Prints a vertical histogram.

WRIRL	1418	Prints an integer rectangular matrix with a given format and labels.
WRIRN	1416	Prints an integer rectangular matrix with integer row and column labels.
WROPT	1421	Sets or retrieves an option for printing a matrix.
WRRRL	1412	Prints a real rectangular matrix with a given format and labels.
WRRRN	1410	Prints a real rectangular matrix with integer row and column labels.

Appendix C: References

Abraham and Ledolter

Abraham, Bovas, and Johannes Ledolter (1983), *Statistical Methods for Forecasting*, John Wiley & Sons, New York.

Abramowitz and Stegun

Abramowitz, Milton, and Irene A. Stegun (editors) (1964), *Handbook of Mathematical Functions with Formulas*, *Graphs, and Mathematical Tables*, National Bureau of Standards, Washington.

Afifi and Azen

Afifi, A.A. and S.P. Azen (1979), *Statistical Analysis : A Computer Oriented Approach*, second edition, Academic Press, New York.

Agresti, Wackerly, and Boyette

Agresti, Alan, Dennis Wackerly, and James M. Boyette (1979), Exact conditional tests for cross-classifications: Approximation of attained significance levels, *Psychometrika*, **44**, 75-83.

Ahrens and Dieter

Ahrens, J.H., and U. Dieter (1974), Computer methods for sampling from gamma, beta, Poisson, and binomial distributions, *Computing*, **12**, 223–246.

Ahrens, J.H., and U. Dieter (1985), Sequential random sampling, *ACM Transactions on Mathematical Software*, **11**, 157–169.

Aird and Howell

Aird, Thomas J., and Byron W. Howell (1991), IMSL Technical Report 9103, IMSL, Houston.

Akaike

Akaike, H. (1980) Seasonal Adjustment by Bayesian Modeling. Journal of Time Series Analysis, Vol 1, 1-13.

Akaike, H., et al

Akaike, H., Kitagawa, G., Arahata, E., and Tada, F. (1979) Computer Science Monographs, No. 13, Institute of Statistical Mathematics, Tokyo.

Akaike and Nakagawa

Akaike, H. and Nakagawa, T. (1972) *Statistical Analysis and Control of Dynamic Systems*, KTK Scientific Publishers, Tokyo.

Akima

Akima, Hirosha (1970), A new method of interpolation and smooth curve fitting based on local procedures, *Journal of the ACM*, **17**, 589–602.

Anderberg

Anderberg, Michael R. (1973), *Cluster Analysis for Applications*, Academic Press, New York.

Anderson

Anderson, T.W. (1971), *The Statistical Analysis of Time Series*, John Wiley & Sons, New York.

Anderson and Bancroft

Anderson, R.L., and T.A. Bancroft (1952), *Statistical Theory in Research*, McGraw-Hill Book Company, New York.

Anderson and Rubin

Anderson, T., and H. Rubin (1956), Statistical inference in factor analysis, *Proceedings of the Third Berkely Symposium on Mathematical Statistics and Probability*, Volume 5, University of California Press, Berkeley, 111–150.

Atkinson

Atkinson, A.C. (1973), Testing transformations to normality, *Journal of the Royal Statistical Society*, Series B: Methodological, **35**, 473–479.

Atkinson, A.C. (1979), A family of switching algorithms for the computer generation of beta random variates, *Biometrika*, **66**, 141–145.

Atkinson, A.C. (1985), *Plots, Transformations, and Regression*, Claredon Press, Oxford.

Atkinson, A.C. (1986), Diagnostic tests for transformations, *Technometrics*, **28**, 29–37.

Baker, Clarke, and Lane

Baker, R.J., M.R.B. Clarke, and P.W. Lane (1985). Zero entries in contingency tables, *Computational Statistics and Data Analysis*, **3**, 33-45.

Bartlett

Bartlett, M.S. (1935), Contingency table interactions, *Journal of the Royal Statistics Society Supplement*, **2**, 248–252.

Bartlett, M. (1937), The statistical conception of mental factors, *British Journal of Psychology*, **28**, 97–104.

Bartlett, M.S. (1946), On the theoretical specification and sampling properties of autocorrelated time series, *Supplement to the Journal of the Royal Statistical Society*, **8**, 27–41.

Bartlett, M.S. (1978), *Stochastic Processes,* 3rd. ed., Cambridge University Press, Cambridge.

Barrodale and Roberts

Barrodale, I., and F.D.K. Roberts (1973), An improved algorithm for discrete L_1 approximation, *SIAM Journal on Numerical Analysis*, **10**, 839–848.

Barrodale, I., and C. Phillips (1975), Algorithm 495. Solution of an overdetermined system of linear equations in the Chebyshev norm, *ACM Transactions on Mathematical Software*, **1**, 264–270.

Barrodale, I., and F.D.K. Roberts (1974), Solution of an overdetermined system of equations in the l_1 norm, *Communications of the ACM*, **17**, 319–320.

Barlow et al.

Barlow, R.E., D.J. Bartholomew, J.M. Bremner, and H.D. Brunk (1972), *Statistical Inference under Order Restrictions*, John Wiley & Sons, London.

Bendel and Mickey

Bendel, Robert B., and M. Ray Mickey (1978), Population correlation matrices for sampling experiments, *Communications in Statistics*, **B7**, 163–182.

Berk

Berk, Kenneth. N. (1976), Tolerance and condition in regression computations, *Proceedings of the Ninth Interface Symposium on Computer Science and Statistics*, Prindle, Weber and Schmidt, Boston, 202–203.

Best and Fisher

Best, D.J., and N.I. Fisher (1979), Efficient simulation of the von Mises distribution, *Applied Statistics*, **28**, 152–157.

Bhapkar

Bhapkar, V.P. (1961), A nonparametric test for the problem of several samples, *Annals of Mathematical Statistics*, **32**, 1108–1117.

Bishop, Feinberg, and Holland

Bishop, Yvonne M. M., Stephen E. Feinberg, and Paul W. Holland (1975), *Discrete Multivariate Analysis*, The MIT Press, Cambridge, Mass.

Bjorck and Golub

Bjorck, Ake, and Gene H. Golub (1973), Numerical Methods for Computing Angles Between Subspaces, *Mathematics of Computation,* **27**, 579–594.

Blackman and Tukey

Blackman, R.B., and J. W. Tukey (1958), *The Measurement of Power Spectra from the Point of View of Communications Engineering*, Dover Publications, New York.

Blom

Blom, Gunnar (1958), *Statistical Estimates and Transformed Beta-Variables*, John Wiley & Sons, New York.

Boisvert, Howe, and Kahaner

Boisvert, Ronald F., Sally E. Howe, and David K. Kahaner (1985), GAMS: A framework for the management of scientific software, *ACM Transactions on Mathematical Software*, **11**, 313-355.

Boisvert, Howe, Kahaner, and Springmann

Boisvert, Ronald F., Sally E. Howe, David K. Kahaner, and Jeanne L. Springmann (1990), *Guide to Available Mathematical Software*, NISTIR 90-4237, National Institute of Standards and Technology, Gaithersburg, Maryland.

Bosten and Battiste

Bosten, Nancy E., and E.L. Battiste (1974), Incomplete beta ratio, *Communications of the ACM* **17**, 156–157.

Box and Cox

Box, G.E.P., and D.R. Cox (1964), An analysis of transformations, *Journal of the Royal Statistical Society*, Series B: *Methodological*, **26**, 211–243.

Box and Jenkins

Box, George E.P., and Gwilym M. Jenkins (1976), *Time Series Analysis: Forecasting and Control*, rev. ed., Holden-Day, Oakland, Calif.

Box and Pierce

Box, G.E.P., and David A. Pierce (1970), Distribution of residual autocorrelations in autoregressive-integrated moving average time series models, *Journal of the American Statistical Association*, **65**, 1509–1526.

Box and Tidwell

Box, G.E.P., and P.W. Tidwell (1962), Transformation of the independent variables, *Technometrics*, **4**, 531–550.

Boyette

Boyette, James M. (1979), Random RC tables with given row and column totals, *Applied Statistics*, **28**, 329–332.

Bradley

Bradley, J.V. (1968), *Distribution-Free Statistical Tests*, Prentice-Hall, New Jersey.

Bradley, J.V. (1968), *Distribution-Free Statistical Inference*, Prentice-Hall, New Jersey.

Breslow

Breslow, N.E. (1974), Covariance analysis of censored survival data, *Biometrics*, **30**, 89–99.

Brillinger

Brillinger, David R. (1981), *Time Series: Data Analysis and Theory*, expanded ed., Holden-Day, San Francisco.

Bross

Bross, I. (1950), Fiducial intervals for variance components, *Biometrics*, **6**, 136−144.

Brown

Brown, Morton B. (1983), BMDP4F, two-way and multiway frequency tables measures of association and the log-linear model (complete and incomplete tables), in *BMDP Statistical Software*, *1983 Printing with Additions*, (edited by W. J. Dixon), University of California Press, Berkeley.

Brown and Benedetti

Brown, Morton B, and Jacqualine K. Benedetti (1977), Sampling behavior and tests for correlation in two-way contingency tables, *Journal of the American Statistical Association,* **42**, 309-315.

Brown and Fuchs

Brown, Morton B., and C. Fuchs (1983), On maximum likelihood estimation in sparse contingency tables, *Computational Statistics and Data Analysis*, **1**, 3−15.

Bryson and Johnson

Bryson, Maurice C. and Mark E. Johnson (1981), The incidence of monotone likelihood in the Cox model, *Technometrics*, **23**, 381–384.

Cantor

Cantor, Alan B. (1979), A computer algorithm for testing significance in M K contingency tables, *Proceedings of the Statistical Computing Section, American Statistical Association*, Washington, D.C., 220–221.

Carroll and Chang

Carroll, J.D., and J.J. Chang (1970), Analysis of individual differences in multidimensional scaling via an *n*-way generalization of "Eckart-Young" decomposition, *Psychometrika*, **35**, 283–319.

Chambers et al.

Chambers, J.M., C.L. Mallows, and B.W. Stuck (1976), A method for simulating stable random variates, *Journal of the American Statistical Association*, **71**, 340–344.

Chambers, John M., William S. Cleveland, Beat Kleiner, and Paul A. Tukey (1983), *Graphical Methods for Data Analysis*, Wadsworth, Belmont, Calif.

Chatfield

Chatfield, C. (1980), *The Analysis of Time Series: An Introduction*, 2d ed., Chapman and Hall, London.

Chiang

Chiang, Chin Long (1968), *Introduction to Stochastic Processes in Statistics*, John Wiley & Sons, New York.

Chiang, Chin Long (1972), On constructing current life tables, *Journal of the American Statistical Association*, **67**, 538–541.

Cheng

Cheng, R.C.H. (1978), Generating beta variates with nonintegral shape parameters, *Communications of the ACM*, **21**, 317–322.

Christensen

Christensen, Ronald (1989), Lack-of-fit tests based on near or exact replicates, *Annals of Statistics,* **17**, 673–683.

Clarke

Clarke, M.R.B. (1982), The Gauss-Jordan sweep operator with detection of collinearity, *Applied Statistics*, **31**, 166–168.

Clarkson

Clarkson, Douglas B. (1988a), Remark on Algorithm AS 211: The F-G diagonalization algorithm, *Applied Statistics*, **38**, 147–151.

Clarkson, Douglas B. (1988b), A least-squares version of AS 211: The F-G diagonalization algorithm, *Applied Statistics*, **38**, 317–321.

Clarkson and Fan

Clarkson, Douglas B. and Yuan-An Fan (1989), *Some improvements to the network algorithm for exact probabilities in contingency tables*, IMSL Technical Report 8903, IMSL, Houston.

Clarkson and Gentle

Clarkson, Douglas B. and James E. Gentle (1986), Methods for multidimensional scaling, in *Computer Science and Statistics*, *Proceedings of the Seventeenth Symposium on the Interface*, (D.M. Allen, editor), North-Holland, Amsterdam, 185–192.

Clarkson and Jennrich

Clarkson, Douglas B. and Robert I. Jennrich (1988), Computing extended maximum likelihood estimates for linear parameter models, IMSL *Technical Report* 8804, IMSL, Houston.

Clarkson, Douglas B. and Robert I. Jennrich (1988), Quartic rotation criteria and algorithms, *Psychometrika*, **53**, 251–259.

Clarkson, Douglas B. and Robert I. Jennrich (1991), Computing extended maximum likelihood estimates for linear parameter models, submitted to *Journal of the Royal Statistical Society, Series B*, **53**, 417-426.

Cochran

Cochran, William G. (1977), *Sampling Techniques*, 3rd ed., John Wiley & Sons, New York.

Conover

Conover, W.J. (1980), *Practical Nonparametric Statistics,* 2d ed., John Wiley & Sons, New York.

Conover and Iman

Conover, W.J., and Ronald L. Iman (1983), *Introduction to Modern Business Statistics,* John Wiley & Sons, New York.

Cook and Weisberg

Cook, R. Dennis, and Sanford Weisberg (1982), *Residuals and Influence in Regression*, Chapman and Hall, New York.

Cooper

Cooper, B.E. (1968), Algorithm AS4, An auxiliary function for distribution integrals, *(Applied Statistics*), **17**, 190–192.

Coveyou and MacPherson

Coveyou, R.R., and R.D. MacPherson (1967), Fourier analysis of uniform random number generators, *Journal of the ACM*, **14**, 100–119.

Cox

Cox, David R. (1970), *The Analysis of Binary Data*, Methuen, London.

Cox, D.R. (1972), Regression models and life tables (with discussion), *Journal of the Royal Statistical Society*, Series B, *Methodology*, **34**, 187–220.

Cox and Lewis

Cox, D.R., and P.A.W. Lewis (1966), *The Statistical Analysis of Series of Events*, Methuen, London.

Cox and Oakes

Cox, D.R., and D. Oakes (1984), *Analysis of Survival Data*, Chapman and Hall, London.

Cox and Stuart

Cox, D.R., and A. Stuart (1955), Some quick sign tests for trend in location and dispersion, *Biometrika*, **42**, 80–95.

Craddock

Craddock, J.M. (1969), *Statistics in the Computer Age*, American Elsevier, New York.

Crawford and Ferguson

Crawford, C.B. and G.A. Ferguson (1970), A general rotation criteria and its use in orthogonal rotation, *Psychometrika*, **35**, 321–332.

D'Agostino and Stevens

D'Agostino, Ralph B. and Michael A. Stevens (1986) *Goodness-of-Fit Techniques*, Marcel Dekker, New York.

Dahlquist and Bjorck

Dahlquist, Germund, and Ake Bjorck (1974), *Numerical Methods*, translated by Ned Anderson, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Dallal and Wilkinson

Dallal, Gerald E. and Leland Wilkinson (1986), An analytic approximation to the distribution of Lilliefor's test statistic for normality, *The American Statistician*, **40**, 294–296.

Davison

Davison, Mark L. (1983), *Multidimensional Scaling*, John Wiley & Sons, New York.

De Leeuw and Pruzansky

De Leeuw, Jan and Sandra Pruzansky (1978), A new computational method to fit the weighted Euclidean distance model, *Psychometrika*, **43**, 479–490.

Deming and Stephan

Deming, W.E., and F.F. Stephan (1940), On the least-squares adjustments of a sampled frequency table when the expected marginal totals are known, *Annals of Mathematical Statistics*, **11**, 427–444.

Demspter, Nan, and Rubin

Demspter, Arthur P., Nan, Laird, and Donald B. Rubin (1977), Maximum likelihood from incomplete data via the EM algorithm (with discussion), *Journal of the Royal Statistical Society*, Serie B, **39**, 1–38.

Dennis and Schnabel

Dennis, John E., Jr., and Robert B. Schnabel (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, New Jersey.

Dewey

Dewey, Michael E. (1984), A remark on Algorithm AS 169: An improved algorithm for scatter plots, *Applied Statistics*, **33**, 370–372.

Draper and Smith

Draper, N.R., and H. Smith (1981), *Applied Regression Analysis*, 2d ed., John Wiley & Sons, New York.

Durbin

Durbin, J. (1960), The fitting of time series models, *Revue Institute Internationale de Statistics*, **28**, 233–243.

Efroymson

Efroymson, M.A. (1960), Multiple regression analysis, in *Mathematical Methods for Digital Computers*, Volume 1, (edited by A. Ralston and H. Wilf), John Wiley & Sons, New York, 191–203.

Ekblom

Ekblom, Hakan (1973), Calculation of linear best L_p-approximations, *BIT*, **13**, 292–300.

Ekblom, Hakan (1987), The L_1-estimate as limiting case of an L_p or Huber-estimate, in *Statistical Data Analysis Based on the L_1-Norm and Related Methods* (edited by Yadolah Dodge), North-Holland, Amsterdam, 109–116.

Elandt-Johnson and Johnson

Elandt-Johnson, Regina C., and Norman L. Johnson (1980), *Survival Models and Data Analysis*, John Wiley & Sons, New York, 172—173.

Emerson and Hoaglin

Emerson, John D., and David C. Hoaglin (1983), Analysis of two-way tables by medians, in *Understanding Robust and Exploratory Data Analysis* (edited by David C. Hoaglin, Frederick Mosteller, and John W. Tukey), John Wiley & Sons, New York, 166–210.

Emmett

Emmett W.G. (1949), Factor analysis by Lawley's method of maximum likelihood, *British Journal of Psychology*, **2**, 90-97.

Fisher

Fisher, R.A. (1936), The use of multiple measurements in taxonomic problems, *The Annals of Eugenics*, **7**, 179–188.

Fishman

Fishman, George S. (1978), *Principles of Discrete Event Simulation*, John Wiley & Sons, New York.

Fishman et al.

Fishman, George F., and Louis R. Moore, III (1982), A statistical evaluation of multiplicative random number generators with modulus 2311, Journal of the American Statistical Association, **77**, 129–136.

Fishman, George F., and Louis R. Moore, III (1986), An exhaustive analysis of multiplicative congruential random number generators with modulus $2^{31} - 1$, *SIAM Journal on Scientific and Statistical Computing*, **7**, 24–45.

Flury

Flury, Bernard H. (1984), Common principal components in *k* groups, *Journal of the American Statistical Association*, **79**, 892–898.

Flury, Bernard H. (1988), *Common Principal Components & Related Multivariate Models*, John Wiley & Sons, New York.

Flury and Constantine

Flury, Bernard H. and Gregory Constantine (1985), The F-G diagonalization algorithm, *Applied Statistics*, **35**, 177–183.

Flury and Gautschi

Flury, Bernard H. and Walter Gautschi (1986), An algorithm for simultaneous orthogonal transformation of several positive definite symmetric matrices to nearly diagonal form, *SIAM Journal of Scientific and Statistical Computing*, **7**, 169–185.

Forsythe

Forsythe, G.E. (1957), Generation and use of orthogonal polynomials for fitting data with a digital computer, *SIAM Journal on Applied Mathematics*, **5**, 74–88.

Forthofer and Koch

Forthofer, Ronald N., and Gary G. Koch (1973), An analysis of compounded functions of categorical data, *Biometrics*, **29**, 143–157.

Fox, Hall, and Schryer

Fox, P.A., A.D. Hall, and N.L. Schryer (1978), The PORT mathematical subroutine library, *ACM Transactions on Mathematical Software*, **4**, 104–126.

Frane

Frane, James W. (1977), A note on checking tolerance in matrix inversion and regression, *Technometrics*, **19**, 513–514.

Freeman and Halton

Freeman, G.H., and J.H. Halton (1951), Note on the exact treatment of contingency, goodness of fit, and other problems of significance, *Biometrika*, **38**, 141–149.

Friedman, Bentley, and Finkel

Friedman, Jerome H., Jon Louis Bentley, and Raphael Ari Finkel (1977), An algorithm for finding best matches in logarithmic expected time, *ACM Transactions on Mathematical Software*, **3**, 209–226.

Fuller

Fuller, Wayne A. (1976), *Introduction to Statistical Time Series*, John Wiley & Sons, New York.

Furnival and Wilson

Furnival, G.M., and R.W. Wilson, Jr. (1974), *Regressions by leaps and bounds*, *Technometrics*, **16**, 499–511.

Fushimi

Fushimi, Masanori (1990), Random number generation with the recursion $X_t = X_{t-3p} \oplus X_{t-3q}$, *Journal of Computational and Applied Mathematics*, **31**, 105–118.

Gentle

Gentle, James E. (1981), Portability considerations for random number generators, in *Computer Science and Statistics: The Interface*, (edited by William F. Eddy), SpringerVerlag, New York, 158–161.

Gentle, James E. (1990), Computer implementation of random number generators, *Journal of Computational and Applied Mathematics*, **31**, 119–125.

Gentleman

Gentleman, W. Morven (1974), Basic procedures for large, sparse or weighted linear least squares problems, *Applied Statistics*, **23**, 448–454.

Gibbons

Gibbons, J.D. (1971), *Nonparametric Statistical Inference*, McGraw-Hill, New York.

Girshick

Girshick, M.A. (1939), On the sampling theory of roots of determinantal equations, *Annals of Mathematical Statistics*, **10**, 203–224.

Golub

Golub, Gene H. (1969), Matrix computations and statistical calculations, in *Statistical Computation*, (edited by Roy C. Milton and John A. Nelder), Academic Press, New York. 365–398.

Golub and Van Loan

Golub, Gene H. and Charles F. Van Loan (1983), *Matrix Computations*, The Johns Hopkins University Press, Baltimore, Maryland.

Gonin and Money

Gonin, Rene, and Arthur H. Money (1989), *Nonlinear L_p-Norm Estimation*, Marcel Dekker, New York.

Goodnight

Goodnight, James H. (1979), A tutorial on the SWEEP operator, *The American Statistician*, **33**, 149–158.

Granger and Newbold

Granger, C.W.J., and Paul Newbold (1977), *Forecasting Economic Time* Series, Academic Press, Orlando, Florida.

Graybill

Graybill, Franklin A. (1976), *Theory and Application of the Linear Model*, Duxbury Press, North Scituate, Mass.

Griffin and Redish

Griffin, R., and K.A. Redish (1970), Remark on Algorithm 347: An efficient algorithm for sorting with minimal storage, *Communications of the ACM*, **13**, 54.

Grizzle, Starmer, and Koch

Grizzle, J.E., C.F. Starmer, and G.G. Koch, (1969), Analysis of categorical data by linear models, *Biometrics*, **28**, 489-504.

Gross and Clark

Gross, Alan J., and Virginia A. Clark (1975), *Survival Distributions: Reliability Applications in the Biomedical Sciences*, John Wiley & Sons, New York.

Gruenberger and Mark

Gruenberger, F., and A.M. Mark (1951), The d^2 test of random digits, *Mathematical Tables and Other Aids in Computation*, **5**, 109–110.

Guerra et al.

Guerra, Victor O., Richard A. Tapia, and James R. Thompson (1976), A random number generator for continuous random variables based on an interpolation procedure of Akima, in *Proceedings of the Ninth Interface Symposium on Computer Science and Statistics*, (edited by David C. Hoaglin and Roy E. Welsch), Prindle, Weber & Schmidt, Boston, 228–230.

Haberman

Haberman, S.J. (1972), Log-linear fit for contingency tables, *Applied Statistics*, **21**, 218–225.

Haldane

Haldane, J.B.S. (1939), The mean and variance of χ^2 when used as a test of homogeneity, when expectations are small, *Biometrika*, **31**, 346.

Hancock

Hancock, T.W. (1975), Remark on Algorithm 434: Exact probabilities for $R \times C$ contingency tables, *Communications of the ACM*, **18**, 117–119.

Hand

Hand, D.J. (1981), *Discrimination and Classification*, John Wiley & Sons, New York.

Harman

Harman, Harry H. (1976), *Modern Factor Analysis*, 3rd. ed. revised, University of Chicago Press, Chicago.

Harris and Kaiser

Harris, C., and H. Kaiser (1964), Oblique factor analysis solutions by orthogonal transformations, *Psychometrika*, **29**, 347–362.

Hart, et al.

Hart, John F., E.W. Cheney, Charles L. Lawson, Hans J. Maehly, Charles K. Mesztenyi, John R. Rice, Henry G. Thacher, Jr., and Christoph Witzgall (1968), *Computer Approximations*, John Wiley & Sons, New York.

Hartigan

Hartigan, John A. (1975), *Clustering Algorithms*, John Wiley & Sons, New York.

Hartigan and Wong

Hartigan, J.A., and M.A. Wong (1979), Algorithm AS 136: A *K*-means clustering algorithm, *Applied Statistics*, **28**, 100–108.

Harvey

Harvey, A.C. (1981a), *The Econometric Analysis of Time Series*, Philip Allen Publishers, Deddington, England.

Harvey, A.C. (1981b), *Time Series Models*, John Wiley & Sons, New York.

Hayter

Hayter, Anthony J. (1984), A proof of the conjecture that the Tukey-Kramer multiple comparisons procedure is conservative, *Annals of Statistics*, **12**, 61–75.

Heiberger

Heiberger, Richard M. (1978), Generation of random orthogonal matrices, *Applied Statistics*, **27**, 199–206.

Hemmerle

Hemmerle, William J. (1967), *Statistical Computations on a Digital Computer*, Blaisdell Publishing Company, Waltham, Mass.

Hendrickson and White

Hendrickson, A., and P. White (1964), PROMAX: A quick method for rotation to oblique simple structure, *British Journal of Statistical Psychology*, **17**, 65–70.

Herraman

Herraman, C. (1968), Sums of squares and products matrix, *Applied Statistics*, **17**, 289–292.

Hill

Hill, G.W. (1970), Student's *t*-distribution, *Communications of the ACM*, **13**, 617–620.

Hinkley

Hinkley, David (1977), On quick choice of power transformation, *Applied Statistics*, **26**, 67–69.

Hoaglin

Hoaglin, David C. (1983), Letter values: A set of selected order statistics, in *Understanding Robust and Exploratory Data Analysis* (edited by David C. Hoaglin, Frederick Mosteller, and John W. Tukey), John Wiley & Sons, New York, 33–57.

Hoaglin et al.

Hoaglin, David C., Frederick Mosteller, and John W. Tukey (editors) (1983), *Understanding Robust and Exploratory Data Analysis*, John Wiley & Sons, New York.

Hoaglin and Welsch

Hoaglin, D.C., and R. Welsch (1978), The hat matrix in regression and ANOVA, *American Statistician*, **32**, 17–22.

Hocking

Hocking, R.R. (1972), Criteria for selection of a subset regression: Which one should be used?, *Technometrics*, **14**, 967–970.

Hocking, R.R. (1973), A discussion of the two-way mixed model, *The American Statistician*, **27**, 148–152.

Hocking, R.R. (1985), *The Analysis of Linear Models*, Brooks/Cole Publishing Company, Monterey, California.

Huber

Huber, Peter J. (1977), Robust covariances, in *Statistical Decision Theory and Related Topics*, S.S. Gupta and D.S. Moore (editors), Academic Press, New York.

Huber, Peter J. (1981), *Robust Statistics*, John Wiley & Sons, New York.

Hughes and Saw

Hughes, David T., and John G. Saw (1972), Approximating the percentage points of Hotelling's generalized T_0^2 statistic, *Biometrika*, **59**, 224–226.

Hurley and Cattell

Hurley, J., and R. Cattell (1962), The Procrustes program: Producing direct rotation to test a hypothesized factor structure, *Behavioral Science*, **7**, 258–262.

IEEE

ANSI/IEEE Std 754-1985 (1985), *IEEE Standard for Binary Floating-Point Arithmetic*, The IEEE, Inc., New York.

Iman and Davenport

Iman, R.L., and J.M. Davenport (1980), Approximations of the critical region of the Friedman statistic, *Communications in Statistics*, **A9(6)**, 571–595.

Isogai

Isogai, Takafumi (1983), On measures of multivariate skewness and kurtosis, *Mathematica Japonica*, **28**, 251–261.

Jenkins and Watts

Jenkins, Gwilym M., and Donald G. Watts (1968), *Spectral Analysis and Its Applications*, Holden-Day, Oakland, Calif.

Jennrich

Jennrich, Robert I. (1973), Standard errors for obliquely rotated factor loadings, *Psychometrika*, **38**, 593–604.

Jennrich and Robinson

Jennrich, R.I., and S.M. Robinson (1969), A Newton-Raphson algorithm for maximum likelihood factor analysis, *Psychometrika,* **34**, 111–123.

Jennrich and Sampson

Jennrich, R.I. and P.F. Sampson (1966), Rotation for simple loadings, *Psychometrika*, **31**, 313–323.

John (1971)

John, Peter W.M. (1971), *Statistical Design and Analysis of Experiments*, Macmillan Company, New York.

Johnk

Johnk, M.D. (1964), Erzeugung von Betaverteilten und Gammaverteilten Zufallszahlen, *Metrika*, **8**, 5–15.

Johnson and Kotz

Johnson, Norman L., and Samuel Kotz (1969), *Discrete Distributions*, Houghton Mifflin Company, Boston.

Johnson, Norman L., and Samuel Kotz (1970a), *Continuous Univariate Distributions*-1, John Wiley & Sons, New York.

Johnson, Norman L., and Samuel Kotz (1970b), *Continuous Univariate Distributions*-2, John Wiley & Sons, New York.

Johnson and Welch

Johnson, D.G., and W.J. Welch (1980), The generation of pseudo-random correlation matrices, *Journal of Statistical Computation and Simulation*, **11**, 55–69.

Jonckheere

Jonckheere, A.R. (1954), A distribution-free *k*-sample test against ordered alternatives, *Biometrika*, **41**, 133–143.

Joreskog

Joreskog, K.G. (1977), Factor analysis by least squares and maximum-likelihood methods, in *Statistical Methods for Digital Computers*, (edited by Kurt Enslein, Anthony Ralston, and Herbert S. Wilf), John Wiley & Sons, New York, 125–153.

Kachitvichyanukul

Kachitvichyanukul, Voratas (1982), *Computer generation of Poisson, binomial, and hypergeometric random variates*, Ph.D. dissertation, Purdue University, West Lafayette, Indiana.

Kaiser

Kaiser, H.F. (1958), The varimax criterion for analytic rotation in factor analysis, *Psychometrika*, **23**, 187–200.

Kaiser, H.F. (1963), Image analysis, in *Problems in Measuring Change*, (edited by C. Harris), University of Wisconsin Press, Madison, Wisconsin.

Kaiser and Caffrey

Kaiser, H.F., and J. Caffrey (1965), Alpha factor analysis, *Psychometrika*, **30**, 1–14.

Kalbfleisch and Prentice

Kalbfleisch, John D., and Ross L. Prentice (1980), *The Statistical Analysis of Failure Time Data*, John Wiley & Sons, New York.

Kalman

Kalman, R. E. (1960), A new approach to linear filtering and prediction problems, *Journal of Basic Engineering*, **82**, 35–45.

Kelly

Kelly, L.G. (1967), *Handbook of Numerical Methods and Applications,* Addison-Wesley, Reading, Mass.

Kemp

Kemp, A.W., (1981), Efficient generation of logarithmically distributed pseudo-random variables, *Applied Statistics,* **30**, 249–253.

Kempthorne

Kempthorne, Oscar (1975), *The Design and Analysis of Experiments*, Robert E. Krieger Publishing Company, Huntington, New York.

Kendall

Kendall, Maurice G. (1962), *Rank Correlation Methods*, Charles Griffin & Company, 94–100.

Kendall, Stuart, and Ord

Kendall, Maurice G., Alan Stuart, and J. Keith Ord (1983), *The Advanced Theory of Statistics*, Volume 3: *Design and Analysis, and Time Series*, 4th ed., Oxford University Press, New York.

Kendall, Maurice G., Alan Stuart, and J. Keith Ord (1987), *The Advanced Theory of Statistics,* Volume 1: Distribution Theory, 5th ed., Oxford University Press, New York.

Kendall and Stuart

Kendall, Maurice G., and Alan Stuart (1979), *The Advanced Theory of Statistics*, Volume 2: *Inference and Relationship*, 4th ed., Oxford University Press, New York.

Kennedy and Gentle

Kennedy, William J., and James E. Gentle (1980), *Statistical Computing*, Marcel Dekker, New York.

Kim and Jennrich

Kim, P.J., and R.I. Jennrich (1973), Tables of the exact sampling distribution of the two sample Kolmogorov-Smirnov criterion D_{mn} ($m < n$), in *Selected Tables in Mathematical Statistics*,

Volume 1, (edited by H. L. Harter and D.B. Owen), American Mathematical Society, Providence, Rhode Island.

Kinderman

Kinderman, A.J., and J.G. Ramage (1976), Computer generation of normal random variables, *Journal of the American Statistical Association*, **71**, 893−896.

Kinderman, A.J., J.F. Monahan, and J.G. Ramage (1977), Computer methods for sampling from Student's *t* distribution, *Mathematics of Computation* **31**, 1009−1018.

Kinnucan and Kuki

Kinnucan, P., and H. Kuki (1968), A single precision inverse error function subroutine, Computation Center, University of Chicago. Strecok, Anthony J. (1968), On the calculation of the inverse of the error function, *Mathematics of Computation*, **22**, 144–158.

Kirk

Kirk, Roger E. (1982), *Experimental Design: Procedures for the Behavioral Sciences*, 2d. ed., Brooks/Cole Publishing Company, Monterey, Calif.

Kitagawa and Akaike

Kitagawa, G. and Akaike, H. (1978) A procedure for the modeling of non-stationary time series. Ann. Inst. Statist. Math., Vol 30, B, 351-363.

Knuth

Knuth, Donald E. (1973), *The Art of Computer Programming,* Volume 3: *Sorting and Searching*, Addison-Wesley Publishing Company, Reading, Mass.

Knuth, Donald E. (1981), *The Art of Computer Programming*, Volume 2: *Seminumerical Algorithms*, 2d ed., Addison-Wesley, Reading, Mass.

Koch, Amara, and Atkinson

Koch, G.G., I.A. Amara, and S.S. Atkinson (1983), Mantel-Haenszel and related methods in analyzing ordinal categorical data with concomitant information, 39th Annual Conference on Applied Statistics, Newark, New Jersey.

Kotz and Johnson

Kotz, Samuel, and Norman L. Johnson (Editors) (1986), *Encyclopedia of Statistical Sciences*, **7**, John Wiley & Sons, New York.

Kronmal and Peterson

Kronmal, Richard A., and Arthur J. Peterson, Jr. (1979), On the alias method for generating random variables from a discrete distribution, *The American Statistician*, **33**, 214–218.

Kruskal

Kruskal, J.B. (1964), Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis, *Psychometrika*, **29**, 1–27.

Kruskal, Young, and Seery

Kruskal J.B., F.W. Young, and J.B. Seery (1977), How to use KYST, a very flexible program to do multidimensional scaling and unfolding, Unpublished manuscript, Bell Telephone Laboratories, Murray Hill, New Jersey.

Kshirsagar

Kshirsagar, Anant M. (1972), *Multivariate Analysis*, Marcel Dekker, New York.

Lachenbruch

Lachenbruch, Peter A. (1975), *Discriminant Analysis*, Hafner Press, London.

Landis, Cooper, Kennedy, and Koch

Landis, J. Richard, Murray M. Cooper, Thomas Kennedy, and Gary G. Koch (1979), A computer program for testing average partial association in three-way contingency tables (PARCAT), *Computer Programs in Biomedicine*, **9**, 223–246.

Landis, Stanish, Freeman, and Koch

Landis, J. Richard, William M. Stanish, Jean L. Freeman, and Gary G. Koch (1976), A computer program for the generalized chi-square analysis of categorical data using weighted least squares (GENCAT), *Computer Programs in Biomedicine*, **6**, 196–231.

Lawless

Lawless, J.F. (1982), *Statistical Models and Methods for Lifetime Data*, John Wiley & Sons, New York.

Lawley and Maxwell

Lawley, D.N., and A.E. Maxwell (1971), *Factor Analysis as a Statistical Method*, 2d ed., Butterworth, London.

Learmonth et al.

Learmonth, G.P., and P.A. W. Lewis (1973a), *Naval Postgraduate School Random Number Generator Package LLRANDOM, NPS55LW73061A*, Naval Postgraduate School, Monterey, Calif.

Learmonth, G. P., and P. A. W. Lewis (1973b), Statistical tests of some widely used and recently proposed uniform random number generators, in *Computer Science and Statistics: 7th Annual Symposium on the Interface*, (edited by William J. Kennedy), Statistical Laboratory, Iowa State University, Ames, Iowa, 163–171.

Lee (1977)

Lee, S. Keith (1977), On the asymptotic variances of $\hat{\mu}$ terms in log-linear models of multidimensional contingency tables, *Journal of the American Statistical Association*, **72**, 412–419.

Lee (1980)

Lee, Elisa T. (1980), *Statistical Methods for Survival Data Analysis*, Lifetime Learning Publications, Belmont, Calif.

Lehmann

Lehmann, E.L. (1975), *Nonparametrics: Statistical Methods Based on Ranks*, Holden-Day, San Francisco.

Levenberg

Levenberg, K. (1944), A method for the solution of certain problems in least squares, *Quarterly of Applied Mathematics*, **2**, 164–168.

Levin and Marascuilo

Levin, J.R., and L.A. Marascuilo (1983), *Multivariate Statistics in the Social Sciences:* A Researcher's Guide, Wadsworth, Inc., California.

Lewis et al.

Lewis, P.A.W., A. S. Goodman, and J.M. Miller (1969), A pseudorandom number generator for the System/360, *IBM Systems Journal*, **8**, 136–146.

Lewis, P.A.W., and G.S. Shedler (1979), Simulation of nonhomogeneous Poisson processes by thinning, *Naval Logistics Quarterly*, **26**, 403–413.

Lilliefors

Lilliefors, H.W. (1967), On the Kolmogorov-Smirnov test for normality with mean and variance unknown, *Journal of the American Statistical Association*, **62**, 534–544.

Lilliefors, H.W. (1969), On the Kolmogorov-Smirnov test for the exponential distribution with mean unknown, *Journal of the American Statistical Association*, **64**, 387–389.

Lin and Bendel

Lin, Shang P., and Robert B. Bendel (1985), Generation of population correlation matrices with specified eigenvalues, *Applied Statistics*, **34**, 193–198.

Longley

Longley, James W. (1967), An appraisal of least-squares programs for the electronic computer from the point of view of the user, *Journal of the American Statistical Association*, **62**, 819-841.

Ljung and Box

Ljung, G.M., and G.E.P. Box (1978), On a measure of lack of fit in time series models, *Biometrika*, **65**, 297–303.

McCormack

McCormack, R.L. (1965), Extended tables of the Wilcoxon matched pair signed rank test, *Journal of the American Statistical Association*, **60**, 96–102.

McCullagh and Nelder

McCullagh, P., and J.A. Nelder, (1983), *Generalized Linear Models*, Chapman and Hall, London.

McKean and Schrader

McKean, Joseph W., and Ronald M. Schrader (1987), Least absolute errors analysis of variance, in *Statistical Data Analysis Based on the L_1-Norm and Related Methods* (edited by Yadolah Dodge), North-Holland, Amsterdam, 297–305.

McKeon

McKeon, James J. (1974), *F* approximations to the distribution of Hotelling's T_0^2, *Biometrika*, **61**, 381–383.

Maindonald

Maindonald, J.H. (1984), *Statistical Computation*, John Wiley & Sons, New York.

Mandel

Mandel, J. (1961), Non-additivity in two-way analysis of variance, *Journal of the American Statistical Association,* **69**, 859–866.

Marazzi

Marazzi, Alfio (1985), Robust affine invariant covariances in ROBETH, ROBETH-85 document No. 6, Division de Statistique et Informatique, Institut Universitaire de Medecine Sociale et Preventive, Laussanne.

March

March, D.L. (1972), Algorithm 434: *Exact probabilities for R × C contingency tables, Communications of the ACM*, **15**, 991–992.

Mardia et al.

Mardia, K.V. (1970), Measures of multivariate skewness and kurtosis with applications, *Biometrics*, **57**, 519–530.

Mardia, K.V., J.T. Kent, J.M. Bibby (1979), *Multivariate Analysis*, Academic Press, New York.

Mardia and Foster

Mardia, K.V. and K. Foster (1983), Omnibus tests of multinormality based on skewness and kurtosis, *Communications in Statistics A, Theory and Methods*, **12**, 207–221.

Marquardt

Marquardt, D. (1963), An algorithm for least-squares estimation of nonlinear parameters, *SIAM Journal on Applied Mathematics*, **11**, 431–441.

Marsaglia

Marsaglia, George (1964), Generating a variable from the tail of a normal distribution, *Technometrics*, **6**, 101–102.

Marsaglia, G. (1968), Random numbers fall mainly in the planes, *Proceedings of the National Academy of Sciences*, **61**, 25–28.

Marsaglia, G. (1972), The structure of linear congruential sequences, in *Applications of Number Theory to Numerical Analysis*, (edited by S. K. Zaremba), Academic Press, New York, 249–286.

Marsaglia, George (1972), Choosing a point from the surface of a sphere, *The Annals of Mathematical Statistics*, **43**, 645–646.

Marsaglia and Bray

Marsaglia, G. and T.A. Bray (1964), A convenient method for generating normal variables, *SIAM Review*, **6**, 260–264.

Marsaglia et al.

Marsaglia, G., M.D. MacLaren, and T.A. Bray (1964), A fast procedure for generating normal random variables, *Communications of the ACM*, **7**, 4–10.

Martinson and Hamdan

Martinson, E.O., and M.A. Hamdan (1972), Maximum likelihood and some other asymptotically efficient estimators of correlation in two way contingency tables, *Journal of Statistical Computation and Simulation*, **1**, 45–54.

McLeod and Bellhouse

McLeod, A.I., and D.R. Bellhouse (1983), A convenient algorithm for drawing a simple random sample, *Applied Statistics*, **32**, 182–184.

Mehta and Patel

Mehta, Cyrus R., and Nitin R. Patel (1983), A network algorithm for performing Fisher's exact test in *r c* contingency tables, *Journal of the American Statistical Association*, **78**, 427–434.

Mehta, C.R. and N.R. Patel (1986a), Algorithm 643: FEXACT: A FORTRAN subroutine for Fisher's exact test on unordered $r \times c$ contingency tables, *ACM Transactions on Mathematical Software*, **12**, 154–161.

Mehta, C.R. and N.R. Patel (1986b), A hybrid algorithm for Fisher's exact test in unordered $r \times c$ contingency tables, *Communication in Statistics, Series A*, **15**, 387–404.

Merle and Spath

Merle, G., and H. Spath (1974), Computational experiences with discrete L_p approximation, *Computing*, **12**, 315–321.

Meyers

Meyers, Raymond H. (1971), *Response Surface Methodology*, Allyn and Bacon, Boston.

Miller

Miller, Rupert G., Jr. (1980), *Simultaneous Statistical Inference,* 2d ed., SpringerVerlag, New York.

Milliken and Johnson

Milliken, George A., and Dallas E. Johnson (1984), *Analysis of Messy Data: Volume 1*, Designed Experiments, Van Nostrand Reinhold, New York.

Moran

Moran, P.A.P. (1947), Some theorems on time series `I`, *Biometrika*, **34**, 281–291.

More and Hillstrom

More, J.J., B.S. Garbow, and K. E. Hillstrom (1980), *User Guide for MINPACK-1,* Argonne National Labs Report ANL-80-74, Argonne, Ill.

Morrison

Morrison, Donald F. (1976), *Multivariate Statistical Methods*, 2nd. ed. McGraw-Hill Book Company, New York.

Mosier

Mosier, C. (1939), Determining a simple structure when loadings for certain tests are known, *Psychometrika*, **4**, 149–162.

Muller

Muller, M.E. (1959), A note on a method for generating points uniformly on N-dimensional spheres, *Communications of the ACM*, **2**, 19–20.

Neter and Wasserman

Neter, John, and William Wasserman (1974), *Applied Linear Statistical Models,* Richard D. Irwin, Homewood, Ill.

Neter, Wasserman, and Kutner

Neter, John, William Wasserman, and Michael H. Kutner (1983), *Applied Linear Regression Models*, Richard D. Irwin, Homewood, Ill.

Noether

Noether, G.E. (1956), Two sequential tests against trend, *Journal of the American Statistical Association*, **51**, 440–450.

Null and Sarle

Null, Cynthia H., and Warren S. Sarle (1982), Multidimensional Scaling by Least Squares, in *Proceedings of the Seventh Annual SAS Users Group International Conference*, SAS Institute Inc., Cary, North Carolina.

Owen

Owen, D.B. (1962), *Handbook of Statistical Tables*, Addison-Wesley Publishing Company, Reading, Mass.

Owen, D.B. (1965), A special case of the bivariate non-central *t*-distribution, *Biometrika*, **52**, 437–446.

Pagano and Halvorsen

Pagano, Marcello, and Katherine Taylor Halvorsen (1981), An algorithm for finding the exact significance levels in $r \times c$ contingency tables, *Journal of the American Statistical Association*, **76**, 931–934.

Park and Miller

Park, Stephen K., and Keith W. Miller (1988), Random number generators: good ones are hard to find, *Communications of the ACM*, **31**, 1192–1201.

Patefield

Patefield, W.M. (1981), An efficient method of generating $R \times C$ tables with given row and column totals, *Applied Statistics*, **30**, 91–97.

Patefield and Tandy

Patefield, W.M. (1981), and Tandy D. (2000) Fast and Accurate Calculation of Owen's T-Function, *J. Statistical Software*, **5**, Issue 5.

Peixoto

Peixoto, Julio L. (1986), Testable hypotheses in singular fixed linear models, Communications in Statistics: *Theory and Methods*, **15**, 1957–1973.

Peto

Peto, R. (1973), Experimental survival curves for interval-censored data, *Applied Statistics*, **22**, 86–91.

Petro

Petro, R. (1970), Remark on Algorithm 347: An efficient algorithm for sorting with minimal storage, *Communications of the ACM*, **13**, 624.

Pike

Pike, M.C. (1966), A method of analysis of a certain class of experiments in carcinogenesis, *Biometrics*, **22**, 1–39.

Pillai

Pillai, K.C.S. (1985), Pillai's trace, in *Encyclopedia of Statistical Sciences*, *Volume 6*, (edited by Samuel Kotz and Norman L. Johnson), John Wiley & Sons, New York, 725–729.

Pregibon

Pregibon, Daryl (1981), Logistic regression diagnostics, *The Annals of Statistics*, **9**, 705–724.

Priestley

Priestley, M.B. (1981), *Spectral Analysis and Time Series*, Volumes 1 and 2, Academic Press, New York.

Prentice

Prentice, Ross L. (1976), A generalization of the probit and logit methods for dose response curves, *Biometrics*, **32**, 761–768.

Ramsey

Ramsey, James O. (1977), Maximum likelihood estimation in multidimensional scaling, *Psychometrika*, **42**, 241–266.

Ramsey, J.O. (1978), Confidence regions for multidimensional scaling analysis, *Psychometrika*, **43**, 145–160.

Ramsey, J.O. (1983), *Multiscale II Manual*, Unpublished manuscript, McGill University, Montreal, Quebec, Canada.

Rao

Rao, C. Radhakrishna (1973), *Linear Statistical Inference and Its Applications*, 2d ed., John Wiley & Sons, New York.

Robinson

Robinson, Enders A. (1967), *Multichannel Time Series Analysis with Digital Computer Programs*, Holden-Day, San Francisco.

Romesburg and Marshall

Romesburg, C., and K. Marshall (1974), *LIFE: A computer program for stochastic life table analysis*, US/IBP Desert Research Memorandum 74-68, Utah State University, Logan, Utah.

Royston

Royston, J.P. (1982a), An extension of Shapiro and Wilk's *W* test for normality to large samples, *Applied Statistics*, **31**, 115–124.

Royston, J.P. (1982b), The *W* test for normality, *Applied Statistics*, **31**, 176–180.

Royston, J.P. (1982c), Expected normal order statistics (exact and approximate), *Applied Statistics*, **31**, 161–165.

Sallas

Sallas, William M. (1988), Some Remarks on Algorithm AS 164. Least squares subject to linear constraints, *Applied Statistics*, **37**, 484–489.

Sallas, William M. (1990), An algorithm for an L_p norm fit of a multiple linear regression model, *American Statistical Association 1990 Proceedings of the Statistical Computing Section*, 131–136.

Sallas and Harville

Sallas, William M., and David A. Harville (1981), Best linear recursive estimation for mixed linear models, *Journal of American Statistical Association*, **76**, 860–869.

Sallas, William M., and David A. Harville (1988), Noninformative priors and restricted maximum likelihood estimation in the Kalman filter, in *Bayesian Analysis of Time Series and Dynamic Models* (edited by James C. Spall), Marcel Dekker, New York, 477–508.

Sallas and Lionti

Sallas, William M. and Abby M. Lionti (1988), Some useful computing formulas for the nonfull rank linear model with linear equality restrictions, IMSL Technical Report 8805, IMSL, Houston.

Satterthwaite

Satterthwaite, F.E. (1946), An approximate distribution of estimates of variance components, *Biometrics Bulletin*, **2**, 110–114.

Savage

Savage, I. Richard (1956), Contributions to the theory of rank order statistics|the twosample case, *Annals of Mathematical Statistics*, **27**, 590–615.

Scheffe

Scheffe, Henry (1959), *The Analysis of Variance*, John Wiley & Sons, New York.

Schiffman, Reynolds, and Young

Schiffman, Susan S., M. Lance Reynolds, and Forrest W. Young (1981), *Introduction to Multidimensional Scaling: Theory*, *Methods*, *and Applications*, Academic Press, New York.

Schmeiser et al.

Schmeiser, Bruce W., and A.J.G. Babu (1980), Beta variate generation via exponential majorizing functions, *Operations Research*, **28**, 917–926.

Schmeiser, Bruce W., and Ram Lal (1980), Squeeze methods for generating gamma variates, *Journal of the American Statistical Association*, **75**, 679–682.

Schmeiser, Bruce, and Voratas Kachitvichyanukul (1981), *Poisson Random Variate Generation*, Research Memorandum 81-4, School of Industrial Engineering, Purdue University, West Lafayette, Indiana.

Schmeiser, Bruce (1983), Recent advances in generating observations from discrete random variates, in *Computer Science and Statistics: Proceedings of the Fifteenth Symposium on the Interface*, (edited by James E. Gentle), North-Holland Publishing Company, Amsterdam, 154–160.

Schoneman

Schoneman, P.H. (1966), A generalized solution of the orthogonal Procrustes problem, *Psychometrika*, **31**, 1–10.

Scott

Scott, David W. (1976), Nonparametric probability density estimation by optimization theoretic techniques, *Technical Report* **476** 131-1, Rice University, Houston, Texas.

Scott, Tapia, and Thompson

Scott, D.W., R.W. Tapia, and J.R. Thompson (1980), Nonparametric probability density estimation by discrete penalized-likelihood criteria, *The Annals of Statistics*, **8**, 820– 832.

Searle

Searle, S.R. (1971), *Linear Models*, John Wiley & Sons, New York.

Seber

Seber, G.A.F. (1984), *Multivariate Observations*, John Wiley & Sons, New York.

Shampine

Shampine, L.F. (1975), Discrete least-squares polynomial fits, *Communications of the ACM,* **18**, 179–180.

Siegel

Siegel, Sidney (1956), *Nonparametric Statistics for the Behavioral Sciences*, McGraw-Hill, New York.

Silverman

Silverman, Bernard W. (1982), Kernel density estimation using the fast Fourier transform, *Applied Statistics*, **31**, 93–99.

Silverman, Bernard W. (1986), *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London.

Singleton

Singleton, R.C. (1969), Algorithm 347: An efficient algorithm for sorting with minimal storage, *Communications of the ACM*, **12**, 185–187.

Smirnov

Smirnov, N.V. (1939), Estimate of deviation between empirical distribution functions in two independent samples (in Russian), *Bulletin of Moscow University*, **2**, 3–16.

Snedecor and Cochran

Snedecor, George W., and William G. Cochran (1967), *Statistical Methods*, 6th. ed., Iowa State University Press, Ames, Iowa.

Sposito

Sposito, Vincent A. (1989), Some properties of L_p-estimators, in *Robust Regression*: *Analysis and Applications* (edited by Kenneth D. Lawrence and Jeffrey L. Arthur), Marcel Dekker, New York, 23–58.

Spurrier and Isham

Spurrier, John D. and Steven P. Isham (1985), Exact simultaneous confidence intervals for pairwise comparisons of three normal means, *Journal of the American Statistical Association*, **80**, 438–442.

Stablein, Carter, and Novak

Stablein, D.M, W.H. Carter, and J.W. Novak (1981), Analysis of survival data with nonproportional hazard functions, *Controlled Clinical Trials*, **2**, 149–159.

Stahel

Stahel, W. (1981), Robuste Schatzugen: Infinitesimale Opimalitat und Schatzugen von Kovarianzmatrizen, Dissertation no. 6881, ETH, Zurich.

Stephens

Stephens, M.A. (1974), EDF statistics for goodness of fit and some comparisons, *Journal of the American Statistical Association*, **69**, 730–737.

Stirling

Stirling, W.D. (1981), Algorithm AS 164. Least squares subject to linear constraints, *Applied Statistics*, **30**, 204–212 (see correction, page 357).

Stirling, W.D. (1981), Algorithm AS 169: An improved algorithm for scatter plots, *Applied Statistics*, **30**, 345—349.

Stoline

Stoline, Michael R. (1981), The status of multiple comparisons: simultaneous estimation of all pairwise comparisons in one-way ANOVA designs, *The American Statistician*, **35**, 134–141.

Swan

Swan, A.V. (1969a), Computing maximum-likelihood estimates for parameters of the normal distribution from grouped and censored data, *Applied Statistics*, **18**, 65–69.

Swan, A.V. (1969b), Maximum likelihood estimation from grouped and censored normal data, *Applied Statistics*, **18**, 110–114.

Takane and Carroll

Takane, Yoshio, and J. Douglas Carroll (1981), Nonmetric maximum likelihood multidimensional scaling from directional ranking of similarities, *Psychometrika*, **46**, 389–405.

Takane, Young, and De Leeuw

Takane, Y., F.W. Young, and J. De Leeuw (1977), Nonmetric individual differences multidimensional scaling: An alternating least-squares method with optimal scaling features, *Psychometrika*, **42**, 7–67.

Tanner

Tanner, Martin A. (1983), A note on the variable kernel estimator of the hazard function from censored data, *Annals of Statistics*, **11**, 994–998.

Tanner and Thisted

Tanner, Martin A., and Ronald A. Thisted (1982), Generation of random orthogonal matrices, *Applied Statistics*, **31**, 190–192.

Tanner and Wong

Tanner, Martin A., and Wing H. Wong (1983), The estimation of the hazard function from randomly censored data by the kernel method, *Annals of Statistics*, **11**, 989–993.

Tanner, Martin A., and Wing H. Wong (1984), Data-based nonparametric estimation of the hazard function with applications to model diagnostics and exploratory analysis, *Journal of the American Statistical Association*, **79**, 123–456.

Tapia

Tapia, R.A. (1974), A stable approach to Newton's method for general mathematical programming problems in R^n, *Journal of Optimization Theory and Applications*, **14**, 453– 476.

Tapia and Thompson

Tapia, Richard A., and James R. Thompson (1978), *Nonparametric Probability Density Estimation*, Johns Hopkins University Press, Baltimore.

Tatsuoka

Tatsuoka, Maurice M. (1971), *Multivariate Analysis: Techniques for Educational and Psychological Research*, John Wiley & Sons, New York.

Taylor and Thompson

Taylor, Malcolm S., and James R. Thompson (1986), Data based random number generation for a multivariate distribution via stochastic simulation, *Computational Statistics & Data Analysis*, **4**, 93–101.

Thompson

Thompson, James R, (1989), *Empirical Model Building*, John Wiley & Sons, New York.

TIMSAC

TIMSAC, *TIMe Series Analysis and Control* (TIMSAC-72, TIMSAC-74, TIMSAC-78, TIMSAC-84). Developed by the Institute of Statistical Mathematics, Japan.

Tucker and Lewis

Tucker, Ledyard, and Charles Lewis (1973), A reliability coefficient for maximum likelihood factor analysis, *Psychometrika*, **38**, 1–10.

Tukey

Tukey, J.W. (1949), One degree of freedom for nonadditivity, *Biometrics*, **5**, 232.

Tukey, John W. (1962), The future of data analysis, *Annals of Mathematical Statistics*, **33**, 1–67.

Tukey, John W. (1977), *Exploratory Data Analysis*, Addison-Wesley Publishing Company, Reading, Mass.

Turnbull

Turnbull, Bruce W. (1976), The empirical distribution function with arbitrary grouped, censored, and truncated data, *Journal of the Royal Statistical Society*, Series B: *Methodology*, **38**, 290–295.

Van de Geer

Van de Geer, John P. (1971), *Introduction to Multivariate Analysis for the Social Sciences*, W.H. Freeman and Company, San Francisco.

Velleman and Hoaglin

Velleman, Paul F., and David C. Hoaglin (1981), *Applications, Basics, and Computing of Exploratory Data Analysis,* Duxbury Press, Boston.

Verdooren

Verdooren, L.R. (1963), Extended tables of critical values for Wilcoxon's test statistic, *Biometrika*, **50**, 177–186.

Walker

Walker, A.J. (1974), New fast method for generating discrete random numbers with arbitrary frequency distributions, *Electronics Letters*, **10**, 127–128.

Wallace

Wallace, D.L. (1959), Simplified Beta-approximations to the Kruskal-Wallis H-test, *Journal of the American Statistical Association*, **54**, 225–230.

Weisberg

Weisberg, S. (1985), *Applied Linear Regression*, 2d ed., John Wiley & Sons, New York.

Weeks and Bentler

Weeks, David G., and P.M. Bentler (1982), Restricted multidimensional scaling models for asymmetric proximities, *Psychometrika*, **47**, 201–208.

Wilks

Wilks, S.S. (1935), On the independence of *k* sets of normally distributed statistical variables, *Econometrika*, **3**, 309–326.

Williams

Williams, J.S. (1962), A confidence interval for variance component, *Biometrika,* **49**, 278– 281.

Woodfield

Woodfield, Terry J. (1990), Some notes on the Ljung-Box portmanteau statistic, *American Statistical Association 1990 Proceedings of the Statistical Computing Section*, 155–160.

Young and Lewyckyj

Young, F.W., Y. Takane, and R. Lewyckyj (1978), Three notes on ALSCAL, *Psychometrika*, **43**, 433–435.

Young, Forrest W., and Rostyslaw Lewyckyj (1979), *ALSCAL-4 Users Guide*, second edition, Data Analysis and Theory Associates, Chapel Hill, North Carolina.

Product Support

Contacting Visual Numerics Support

Users within support warranty may contact Visual Numerics regarding the use of the IMSL Libraries. Visual Numerics can consult on the following topics:

- Clarity of documentation
- Possible Visual Numerics-related programming problems
- Choice of IMSL Libraries functions or procedures for a particular problem
- Evolution of the IMSL Libraries

Not included in these consultation topics are mathematical/statistical consulting and debugging of your program.

Consultation

Contact Visual Numerics Product Support by faxing 713/781-9260 or by emailing:

`support@houston.vni.com`.

The following describes the procedure for consultation with Visual Numerics.

1. Include your serial (or license) number
2. Include the product name and version number: IMSL Fortran Library Version 5.0
3. Include compiler and operating system version numbers
4. Include the name of the routine for which assistance is needed and a description of the problem

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W