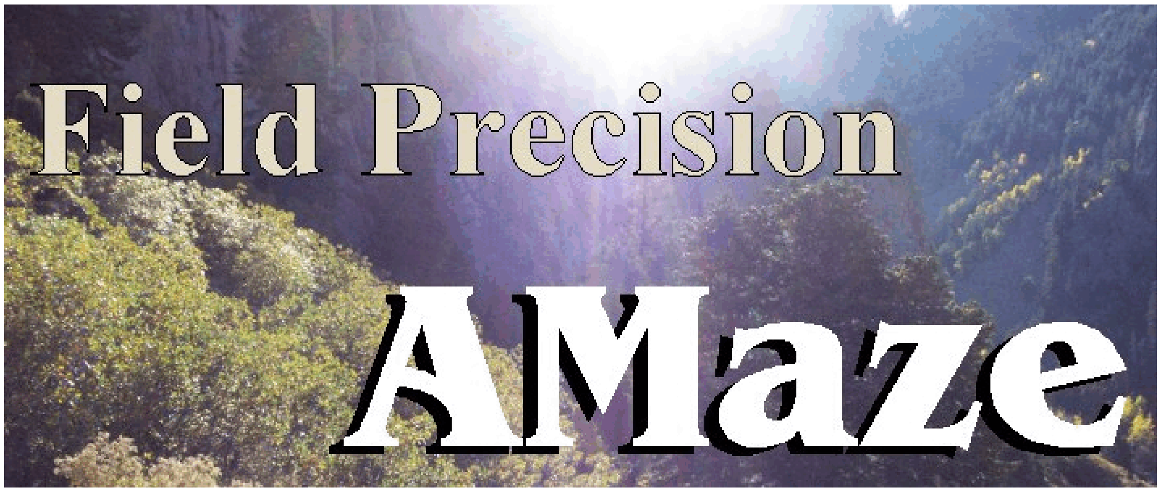


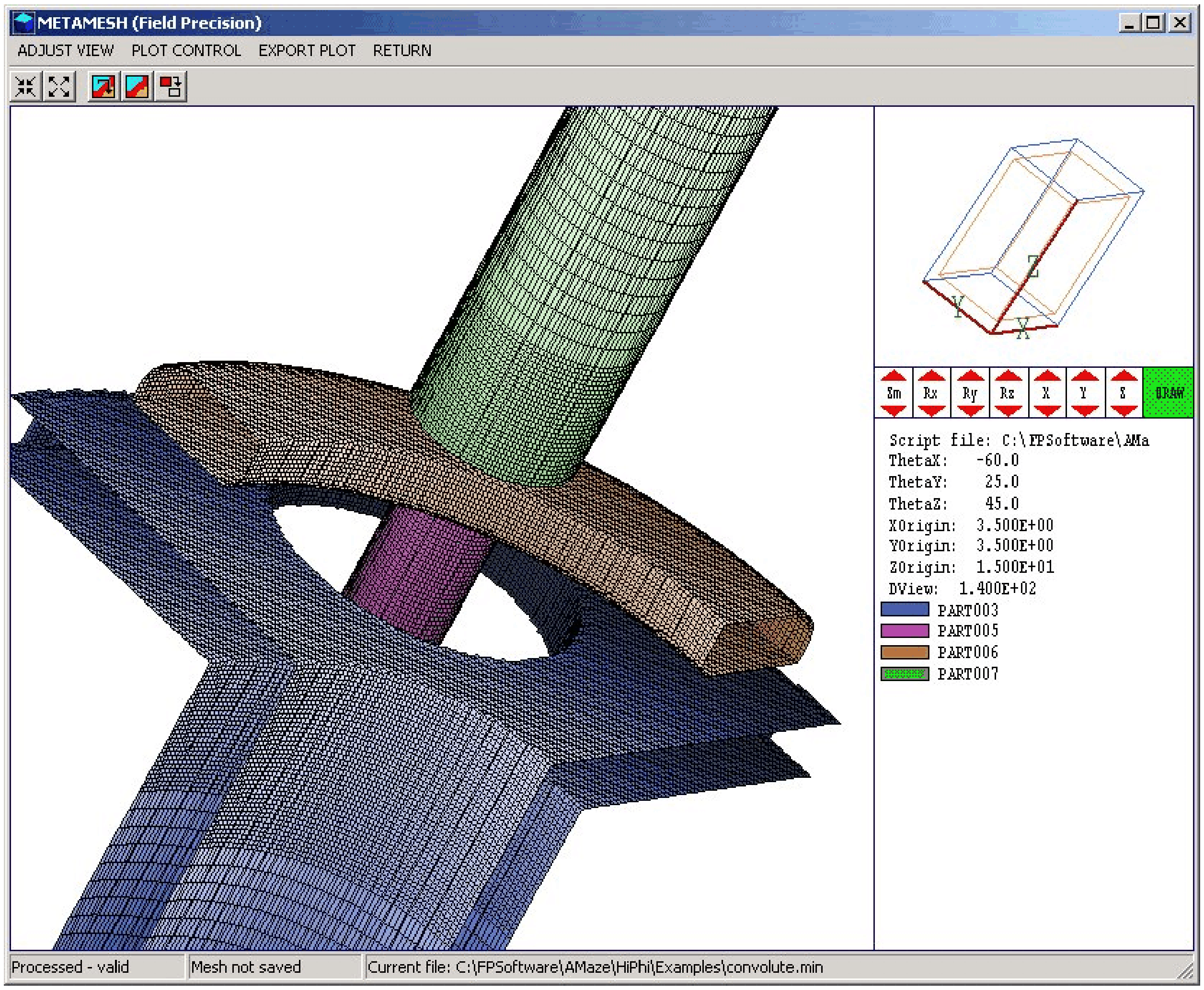
MetaMesh

Three-dimensional Conformal Mesh Generator

Field Precision
Copyright 2003
Internet: www.fieldp.com
E Mail: techninfo@fieldp.com
PO Box 13595, Albuquerque, New Mexico 87192 U.S.A.
Telephone: 505-220-3975, FAX: 505-294-0222

MetaMesh
Three-dimensional Conformal Mesh Generator

Table of contents



Script file: C:\FPSoftware\AMa
ThetaX: -60.0
ThetaY: 25.0
ThetaZ: 45.0
XOrigin: 3.500E+00
YOrigin: 3.500E+00
ZOrigin: 1.500E+01
DView: 1.400E+02
🟦 PART003
🟪 PART005
🟫 PART006
🟩 PART007

Processed - valid
Mesh not saved
Current file: C:\FPSoftware\AMaze\HiPhi\Examples\convolute.min

Figure 1.1. **MetaMesh** screen shot — three-dimensional mesh plot

Chapter 1. Introduction to MetaMesh

1.1 Program function

MetaMesh is the core of the **AMaze** series of 3D finite-element programs. The function of **MetaMesh** is to divide solution spaces into conformal, hexahedron elements to match user-specified geometries. (A *hexahedron* is defined as a generalized six-sided solid figure, basically a box with arbitrary facets.) The program creates files of geometric information for **HiPhi**, **Magnum** and other **AMaze** solution programs. **MetaMesh** can also be used as a stand-alone mesh generator for your own finite-element programs.

The **AMaze** programs handle a wide variety of solutions for variations of physical quantities in space. Examples include thermal transport in solids and current flow in resistive media. These processes are described by continuous differential or integral equations. These equations cannot be solved directly on digital computers. The strategy of both the finite-difference and finite-element methods is first to divide a solution space into small pieces and then to apply the governing equations locally. The technique yields a large set of coupled linear equations that closely approximate the physical content of the continuous equations. Coupled linear equations can be solved easily on a digital computer.

The finite-difference method is based on the direct conversion of differential operators to difference operators that apply over small lengths. In contrast, the finite-element approach uses integrals over small volumes (or *elements*) of the solution space. While finite-difference techniques are usually limited to orthogonal coordinate systems, the flexible integral approach can be applied to elements of any shape. Given the latitude to set element characteristics, we want to ensure that each element has an unambiguous material identity. For example, in an electrostatic simulation an element should be part of an electrode or a dielectric but not a part of both. This constraint implies that the boundaries between material regions should be coincide with the surfaces of elements. Therefore, elements should have shapes and sizes that follow the material divisions of the solution volume. A mesh with this property is called *conformal* because the element shapes conform to the system geometry.

The function of **MetaMesh** is to create sets of conformal elements to represent any user-specified geometry. The program generates output files that can be utilized by all of the **AMaze** solution programs. The following goals guided our development of **MetaMesh**:

- Perform automatic and reliable mesh generation with no required user intervention.
- Maintain a simple input format that minimizes preparation time.
- Implement a mesh organization that gives the fastest and most efficient operation of the solution programs.

Chapter 2 describes how **MetaMesh** compares to other three-dimensional mesh generators. Our unique structured approach offers significant advantages in computational capabilities. Three-dimensional **AMaze** simulations with large meshes run in minutes on high-performance personal computers.

1.2 Learning MetaMesh

Three-dimensional solutions are challenging and **MetaMesh** is not a black box. The quality of the mesh and the accuracy of subsequent field solutions depend on your insight into the problem and careful preparation of input information. This caution aside, we have streamlined the process as much as possible. We guarantee that you will not waste hours manually tweaking a mesh. With a well-prepared script, the process runs automatically in a few seconds.

Scripts are text files that list mesh parameters and geometric specifications. This manual concentrates on the direct preparation of scripts using text editors. The **Geometer** utility to create scripts in an interactive, graphical environment is described in a separate manual. There are several reasons why we use scripts as the basic input medium for all Field Precision programs:

■ Nothing is hidden — all information to generate a mesh is directly accessible.

■ Scripts form succinct and permanent records — with the script you can regenerate a mesh in seconds.

■ It is easy to exchange results with colleagues by E Mail — short ASCII scripts can control the regeneration of large binary files.

■ Scripts are self-documenting, particularly if you add comments that describe simulation features and goals.

■ Small changes can be made with an editor in a few seconds — it is not necessary to backtrack through an involved menu structure.

■ You can create a library of standard assemblies that can be quickly pasted into the script.

The mesh generation process is fast and easy if you have a firm grasp of a few basic concepts. We recommend that you set aside several hours to work through the manual. We have included examples in each chapter along with ready-to-run input files. The walkthrough example in the next section of this chapter will help you test the program installation and get an overview of **MetaMesh** capabilities.

Chapter 2 describes the nature of meshes created by **MetaMesh** and introduces some important terminology. The chapter covers the advantages of a structured conformal mesh and the motivations for hexahedron elements. The chapter also describes the basics of how **Metamesh** operates.

Chapter 3 reviews the structure and syntax of the **MetaMesh** input script while Chapters 4 through 8 address classes of commands that may appear in the file. The *global* commands of Chapter 4 control program operation and the definition of the foundation mesh. Here the term *foundation mesh* applies to the original set of box elements that fills the solution volume. Chapter 5 discusses commands to define *parts*, the basic mesh building blocks of solids. *Regions* of the solution volume may represent electrodes or dielectrics in electrostatic solutions. A region with a complex shape may be built from several parts. The chapter describes how to construct a part, how to orient it in space, and how to move it to its correct position. Chapters 6, 7 and 8 review different classes of parts. Chapter 6 covers simple solids like spheres and cylinders while Chapter 7 covers *extrusions*, *turnings*, and *transitions*, complex solids with arbitrary cross-sections defined by a closed set of line and arc vectors. Chapter 8 discusses the neon model, a method to create tubes that follow arbitrary paths in three-dimensional space. Chapter 9 discusses open parts with zero volume. Creation of an open part involves setting the positions and region numbers of nodes with no changes in the surrounding elements. Open parts are useful to represent filamentary wires, sheets and grids.

The next three chapters concentrate on operation of the **MetaMesh** program. Chapter 10 describes how to launch the program either interactively or in the background under batch file control.. The chapter also covers the commands of the *File menu*. These commands are used to load, to save and to edit data files. Finally, the chapter gives details on the mesh processing operation and the information created in the diagnostic listing file. Chapters 11 and 12 cover plotting capabilities. Plots are invaluable aids to detect errors and to confirm the validity of the mesh. Chapter 11 describes *slice* plots, two-dimensional orthographic views in a logical plane normal to one of the Cartesian axes. Chapter 12 describes *surface* plots, three-dimensional isometric views of the facets of regions or parts (Fig. 1.1). It is also possible to display the nodes of open regions. Finally, Chapter 13 documents the format of the **MetaMesh** output file.

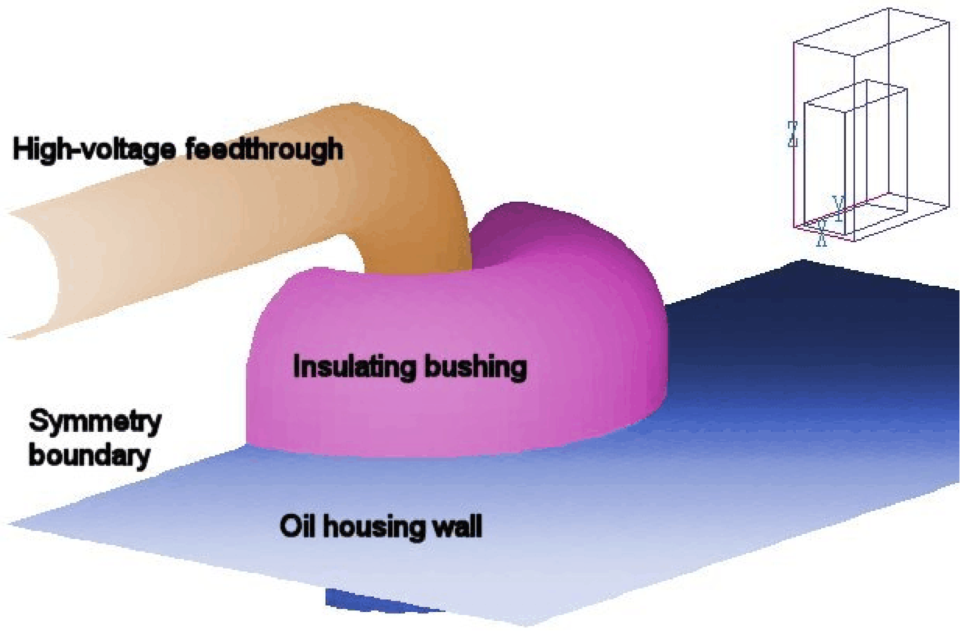
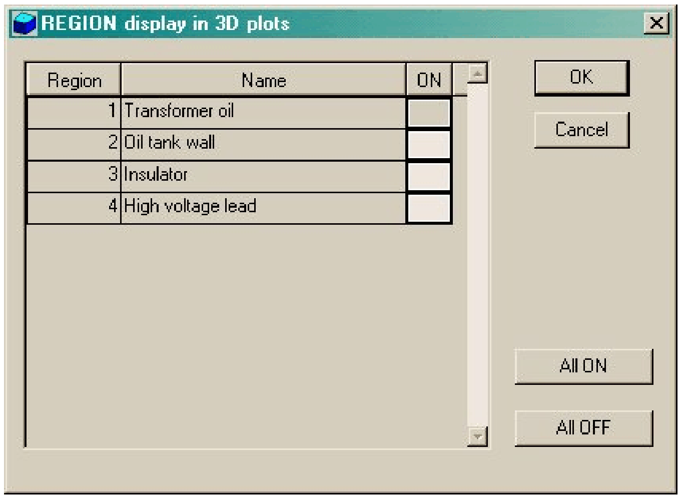


Figure 1.2. Geometry of the application example

1.3 Walkthrough example

The simulation shown in Fig. 1.2 illustrates the creation of a mesh and the steps involved in a three-dimensional solution. A high-voltage bus bar enters an oil housing through a shaped insulator. We want to find values of the electric field magnitude in the oil region near the penetration point. Because the rear plane is a symmetry boundary, we can model only half the system.

First we will briefly review the example and then return to consider some of the details. Copy the file `WALKTHROUGH.MIN` from `\AMAZE\EXAMPLES` to a temporary data storage location such as `\AMAZE\BUFFER`. Run `METAMESH.EXE` and click on *Load MIN file* in the *File* menu. Go to the data directory, pick `WALKTHROUGH.MIN`, and click *OK*. The status bar at the bottom of the window (Fig. 1.1) shows that a file has been loaded but not processed. Click on the command *Process mesh*. **MetaMesh** starts work, reporting the progress of the analysis in a text box. The program analyzes the file content and performs the following operations: 1) create a foundation mesh, 2) assign elements of the foundation mesh to approximate the specified geometry, 3) shift nodes so that element faces fit on material boundaries, and 4) check the integrity of the resulting elements. The main purpose of the rapidly-scrolling screen display is to give you the sense that the program is busy. The information is also recorded in a listing file, `WALKTHROUGH.MLS`, so you can view it


REGION display in 3D plots

Region	Name	ON
1	Transformer oil	
2	Oil tank wall	
3	Insulator	
4	High voltage lead	

Figure 1.3. Region display dialog.

at leisure with a text editor. Mesh generation should take a few seconds. At the end of the analysis click a mouse button or press any key to proceed

The plot menus become active when the mesh has been successfully processed. Click on *Plot3D*. **MetaMesh** makes a default surface plot of Region 2 (in this case, the metal wall of the oil tank). The plot shows facets of elements on surfaces of selected regions. Pick the command *Region/part display* in the *Plot control* menu to call up the dialog of Fig. 1.3. Activate Regions 3 and 4 to display the insulator and high-voltage rod. You should see a plot similar to that of Fig. 1.4. It will be necessary to rotate and to shift the plot to achieve the same view. The orientation tools with red arrows on the right-hand side of the screen are used to rotate about the Cartesian axes or to translate the viewpoint. You can also zoom in and out. Move the mouse over an arrow and click the left mouse button to move the thumbnail display in the upper-right corner of the screen. Click the right mouse button to update the main plot.

At this point you can experiment with the three-dimensional plot capabilities. Pick the *Plot type* command in the *Plot control* menu and try the wireframe display. Although the display may be more difficult to comprehend for complex meshes, the plot regenerates quickly. You can use the wireframe view to orient the plot and then switch back to the hidden surface view. You can also try plotting by parts rather than regions. In this case, the program does color coding on the basis of the part number rather than the region number. The high voltage rod will have three colored zones because it was constructed from three different parts.

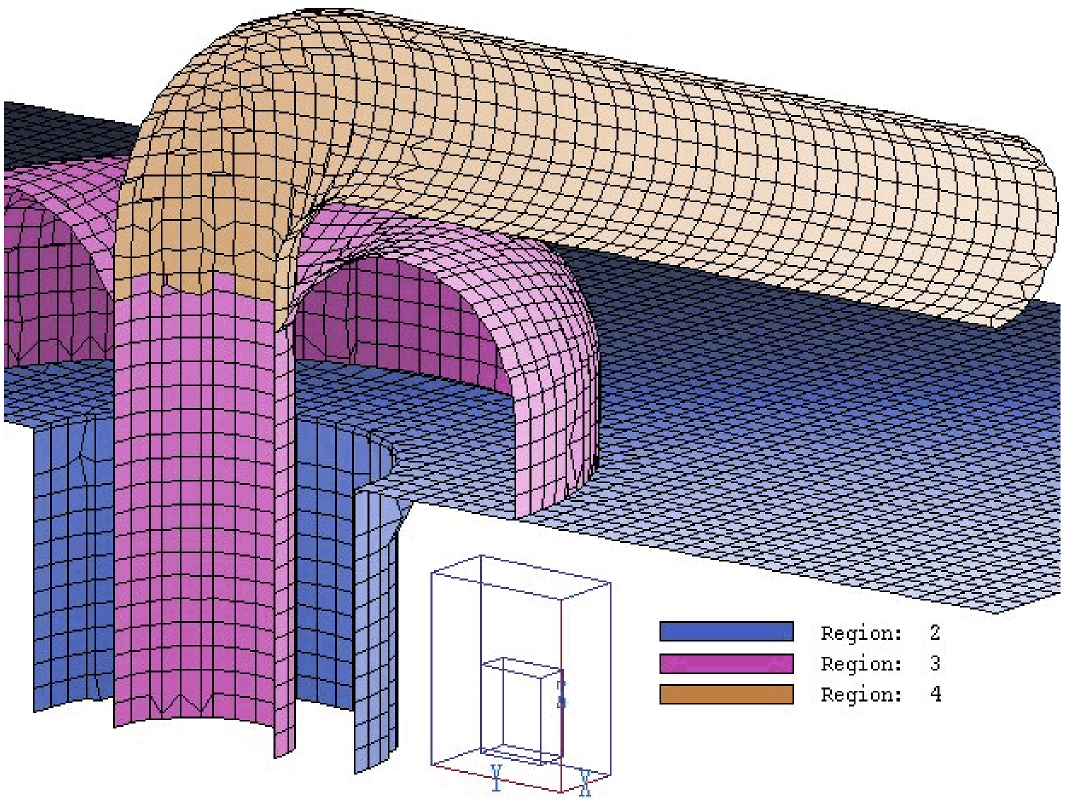
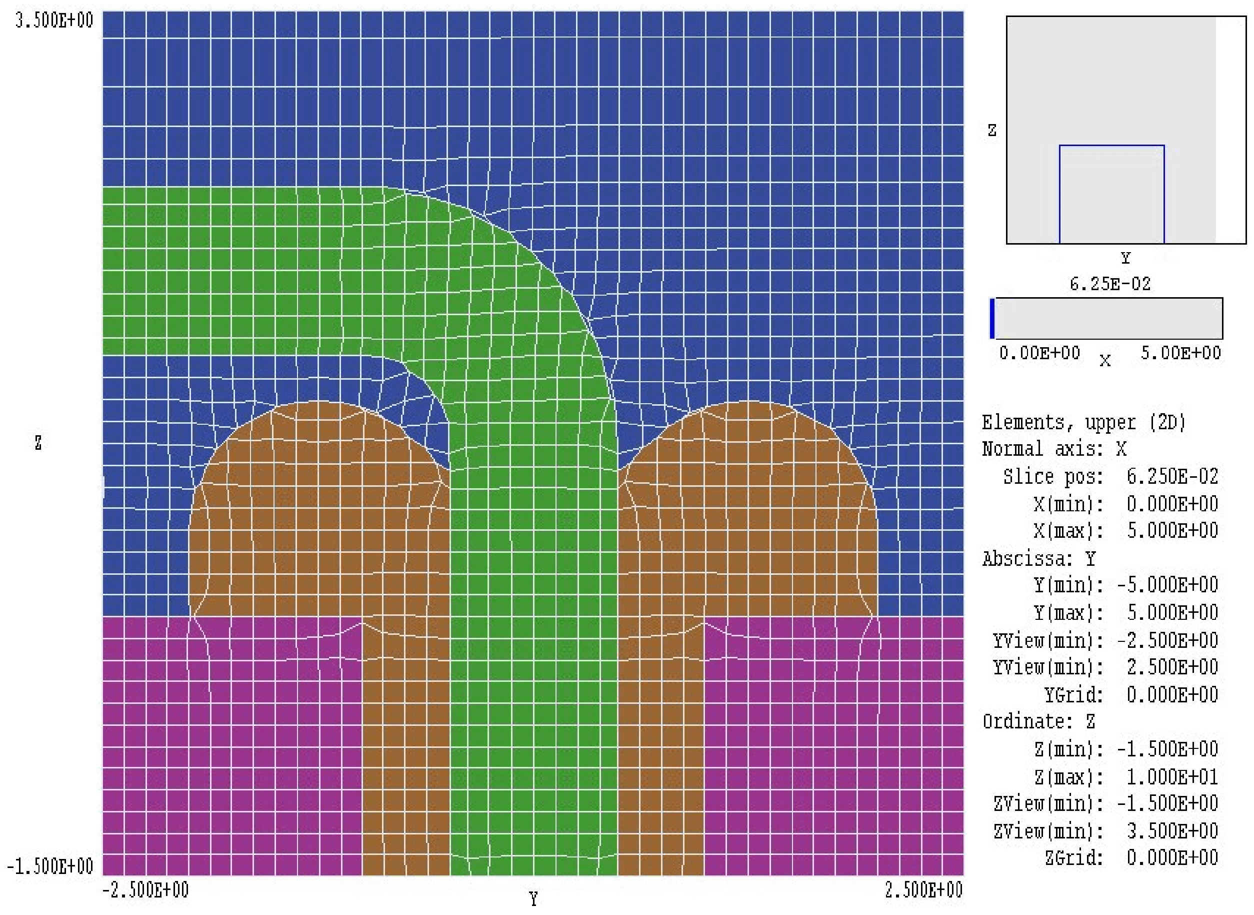


Figure 1.4. 3D view of region surfaces in the completed mesh

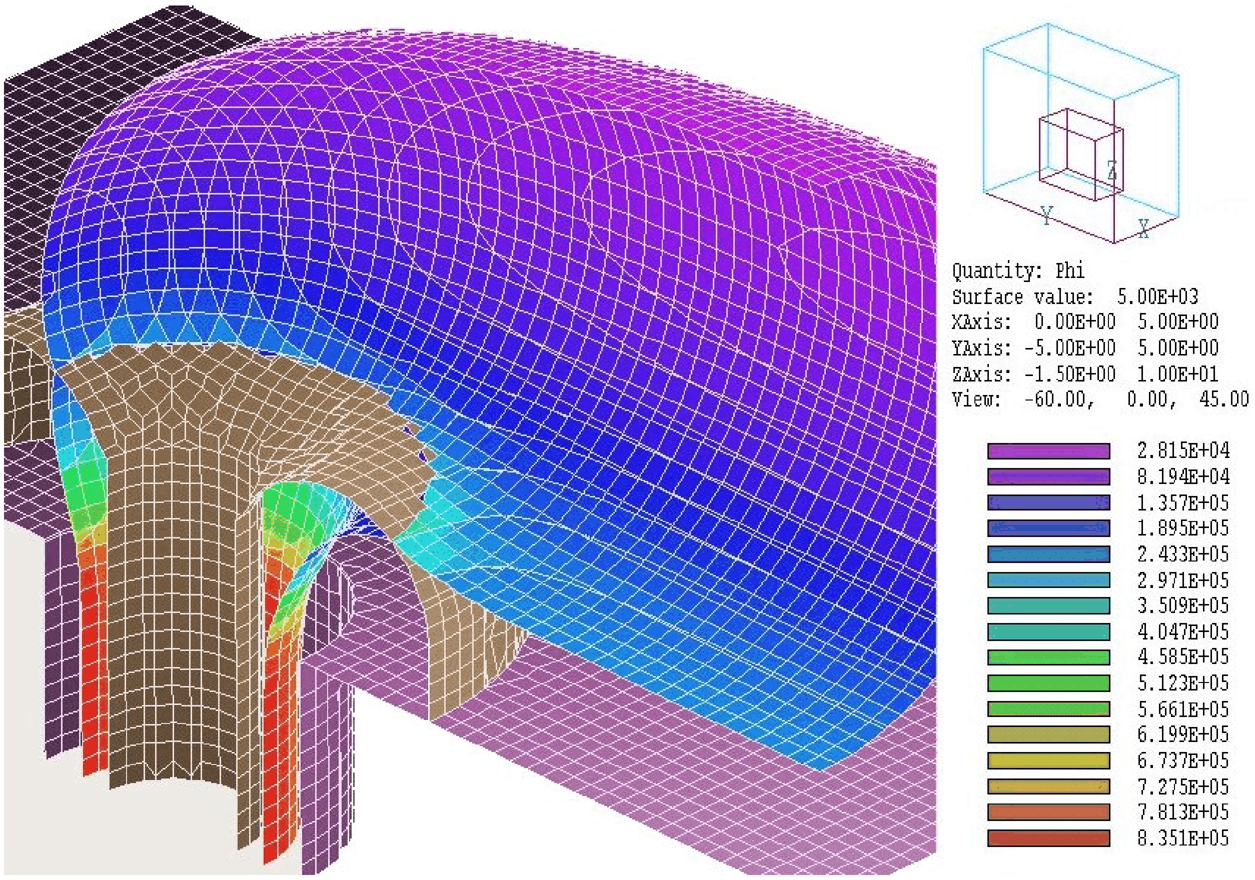
Next we shall check the slice plot capability. Click the *Return* command to return to the main menu and then click *Plot2D*. The default plot is normal to the *z* axis at a slice position near the middle of the solution volume. In the *Adjust view* menu click on *Plot slice* to display a dialog. Use the radio button to pick the *x* axis and move the slider to *x* = 0.0. You can use the *Zoom window* command in the *Adjust view* menu to get a closeup view like that of Fig. 1.5. The red arrow tools take you up and down in the *x* plane. At this point, you can experiment with different view and plot options. In some cases regions may appear to have ragged edges. Chapter 12 gives a complete description of the slice plot algorithms and their limitations. The *Hardcopy* command sends plot information to the Windows default printer. Be sure the default is set correctly before clicking on the command.



```
Elements, upper (2D)
Normal axis: X
  Slice pos:  6.250E-02
     X(min):  0.000E+00
     X(max):  5.000E+00
Abscissa: Y
     Y(min): -5.000E+00
     Y(max):  5.000E+00
  YView(min): -2.500E+00
  YView(min):  2.500E+00
      YGrid:  0.000E+00
Ordinate: Z
     Z(min): -1.500E+00
     Z(max):  1.000E+01
  ZView(min): -1.500E+00
  ZView(min):  3.500E+00
      ZGrid:  0.000E+00
```

Figure 1.5. View of the mesh in a slice normal to the x-axis.

The plot of Fig. 1.5 illustrates the remolding of the foundation mesh boxes by the fitting process. The region where the high-voltage rod enters the insulator (center of plot) is noteworthy. The insulator profile does not exactly follow the theoretical shape because the element size was not small enough to resolve the sharp corner. The example illustrates useful properties of **MetaMesh**: tolerance and flexibility. If you don't give the code enough elements to work with, it will do the best job possible. In this case, the sharp corner has a negligible effect on the fields of interest so finer zoning is unnecessary. Finally, note the change of element resolution in the *z* direction. The volume $z \geq 4.0$ was included to approximate a free-space boundary condition. Figure 1.6 shows an electrostatic solution by the **HiPhi** program using the mesh. The oil inside the tank has a dielectric constant $\epsilon_r = 1.5$ and the insulator has $\epsilon_r = 5.3$. The tank is grounded and the conductor has an applied potential $V_o = 120$ kV. The calculated surface is an equipotential at $\phi = 5$ kV. The color coding shows the magnitude of the electric field.



```
Quantity: Phi
Surface value:  5.00E+03
XAxis:  0.00E+00  5.00E+00
YAxis: -5.00E+00  5.00E+00
ZAxis: -1.50E+00  1.00E+01
View:  -60.00,   0.00,  45.00
```

🟪	2.815E+04
🟪	8.194E+04
🟦	1.357E+05
🟦	1.895E+05
🟦	2.433E+05
🟦	2.971E+05
🟩	3.509E+05
🟩	4.047E+05
🟩	4.585E+05
🟩	5.123E+05
🟩	5.661E+05
🟨	6.199E+05
🟨	6.737E+05
🟧	7.275E+05
🟧	7.813E+05
🟥	8.351E+05

Figure 1.6. Electrostatic solution using the **HiPhi** program.
Surface of ϕ = 5 kV color-coded by |**E**|.

At this point you may wonder how much work was required to specify the mesh geometry. To conclude the example we shall inspect the input script. Press *Return* to go back to the main menu and then click on *Edit MIN file* in the *File* menu. The program loads `WALKTHROUGH.MIN` into the internal text editor. The full text is listed at the end of this section. Here we will step through some parts.

Optional commands of the form:

```
   RegName 1 TransformerOil
```

are helpful to organize the mesh construction session. Note that the names appear in the dialog of Fig. 1.3.

The following statements define the foundation mesh:

```
XMesh
   0.000  5.000  0.125
End
YMesh
  -5.000  5.000  0.125
End
ZMesh
  -1.500  3.000   0.125
   3.000 10.000   0.500
End
```

The commands give the limits of the solution volume along each axis and the element dimensions in the foundation mesh. Note there are two entries for the *z* axis to define regions of different element sizes.

The remaining command groups set the geometries of parts and regions. Some of the sections are simple:

```
Part 2
  Type Box
  Region 2
  Fab   10.000  10.000    1.500
  Shift  0.000   0.000   -0.750
  Surface Region 1
End
```

The section defines the oil tank wall as a box fabricated with lengths 10.0" in *x* and *y* and 1.5" in *z*. The box is constructed in the workshop (reference frame) and then shifted -0.75" in *z* to its proper position. The statement `Surface Region 1` indicates that all nodes on the boundary with Region 1 (the oil inside the tank) should be shifted to and clamped at the border of the box at *z* = 0.0.

You can also specify complex custom shapes. The following section describes the insulator. The part penetrates the tank wall and has a shaped protrusion in the oil region:

```
Part 3
  Type Turning
     L  -1.50  0.50  0.50  0.50
     A   0.50  0.50  1.25  1.25  0.50  1.25 S
     A   1.25  1.25  0.50  2.00  0.50  1.25 S
     L   0.50  2.00  0.00  2.00 SE
     L   0.00  2.00  0.00  1.00 SE
```

```
        L   0.00  1.00 -1.50  1.00
        L  -1.50  1.00 -1.50  0.50
     End
     Region 3
     Fab    0.0  360.0
     Surface Region 1 Edge
     Surface Region 2 Edge
     Coat 2 2
   End
```

The lines beginning with *A* and *L* are a set of line and arc vectors that outline the insulator shape in *r*-*z* space (Fig. 1.5). The outline is rotated about the *z* axis of the reference frame from 0.0° to 360.0° to form a solid. The portion of the part outside the solution volume ($x < 0.0$) is ignored – **MetaMesh** performs automatic clipping. After identifying elements of the foundation mesh that comprise the volume, **MetaMesh** works to shift nodes bordering on the tank (Region 2) and the oil (Region 1) to the specified surfaces.

The part that requires the most effort is the elbow on the high voltage lead. It is a turning with a circular cross-section that extends from 0.0° to 90.0°:

```
   Part 6
     Type Turning SideFit
        A   0.00  0.50  0.50  1.00  0.00  1.00
        A   0.50  1.00  0.00  1.50  0.00  1.00
        A   0.00  1.50 -0.50  1.00  0.00  1.00
        A  -0.50  1.00  0.00  0.50  0.00  1.00
     End
     Region 4
     Fab    0.0  90.0
     Shift  0.000   -1.000  1.000
     Rotate 90.0 0.0 90.0 XYZ
     Surface Region 1
   End
```

The orientation (as fabricated in the reference frame) must be adjusted by two 90° rotations in *x* and *z*. The position must then be adjusted by shifts along *y* and *z*. In this case, we used the graphical capabilities of **MetaMesh** to situate the part correctly in a few tries.

Table 1.1. Complete listing of the file WALKTHROUGH.MIN

```
Global
* Region names
  RegName 1 TransformerOil
  RegName 2 OilTankWall
  RegName 3 Insulator
  RegName 4 HighVoltageLead
  XMesh
     0.000  5.000  0.125
  End
  YMesh
    -5.000  5.000  0.125
  End
  ZMesh
    -1.500  3.000   0.125
     3.000 10.000   0.500
  End
  AxisSmooth Z 5
End
* ----------------------------------------------
Part 1
  Type Box
  Region 1
  Fab   10.000  10.000   20.000
End
* ----------------------------------------------
Part 2
  Type Box
  Region 2
  Fab   10.000  10.000    1.500
  Shift  0.000   0.000   -0.750
  Surface Region 1
End
* ----------------------------------------------
Part 3
  Type Turning
     L  -1.500  0.500  0.500  0.500
     A   0.500  0.500  1.250  1.250  0.500  1.250 S
     A   1.250  1.250  0.500  2.000  0.500  1.250 S
     L   0.500  2.000  0.000  2.000 SE
     L   0.000  2.000  0.000  1.000 SE
     L   0.000  1.000 -1.500  1.000 S
     L  -1.500  1.000 -1.500  0.500
  End
  Region 3
  Fab    0.0  360.0
  Surface Region 1 Edge
  Surface Region 2 Edge
  Coat 2 2
End
```

```
* ----------------------------------------------
Part 4
  Type Cylinder
  Region 4
  Fab   0.500  2.50
  Shift  0.000   0.000  -0.250
  Surface Region 3 Edge
  Surface Region 1 Edge
End
* ----------------------------------------------
Part 5
  Type Cylinder
  Region 4
  Fab  0.500  4.000
  Rotate 90.0 0.0 0.0
  Shift  0.000   -3.000  2.000
  Surface Region 1 Edge
End
* ----------------------------------------------
Part 6
  Type Turning SideFit
     A   0.000  0.500  0.500  1.000  0.000  1.000 S
     A   0.500  1.000  0.000  1.500  0.000  1.000 S
     A   0.000  1.500 -0.500  1.000  0.000  1.000 S
     A  -0.500  1.000  0.000  0.500  0.000  1.000 S
  End
  Region 4
  Fab    0.0  90.0
  Shift  0.000   -1.000  1.000
  Rotate 90.0 0.0 90.0
  Surface Region 1
End
* ----------------------------------------------
EndFile
```

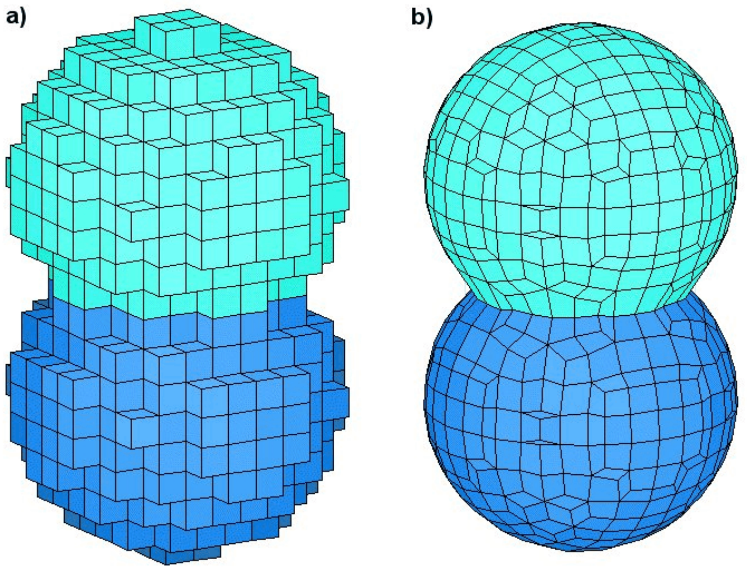


Figure 2.1. Representation of adjoining spheres with (*a*) a regular mesh and (*b*) a conformal mesh.

Chapter 2. Mesh generation basics

2.1 Approaches to three-dimensional mesh generation

This chapter covers terminology and reviews methods that form the basis for **MetaMesh**. A clear understanding will save you time and reduce frustration. This section has two purposes: 1) to review some concepts of three-dimensional mesh generation, and 2) to outline advantages and drawbacks of the **MetaMesh** approach.

As described in Chap. 1, mesh generation for a finite-element solution consists of the division of a solution space into small volumes (or *elements*). In this limit, the governing integral equations can be converted to a large set of coupled linear equations. In a *non-conformal* or *regular* mesh the elements have similar shapes (like the boxes of Fig. 2.1*a*). The elements are then assigned material identities for the best possible representation of physical structures. In general, the element surfaces do not closely follow the physical surfaces. Therefore, curved boundaries have the characteristic stair-step pattern of Fig. 2.1*a*. Regular meshes are easy to construct, and it is simple to determine linear equation coefficients in the subsequent finite-element solution programs. Although a calculation on a regular mesh can provide good accuracy for fields in regions

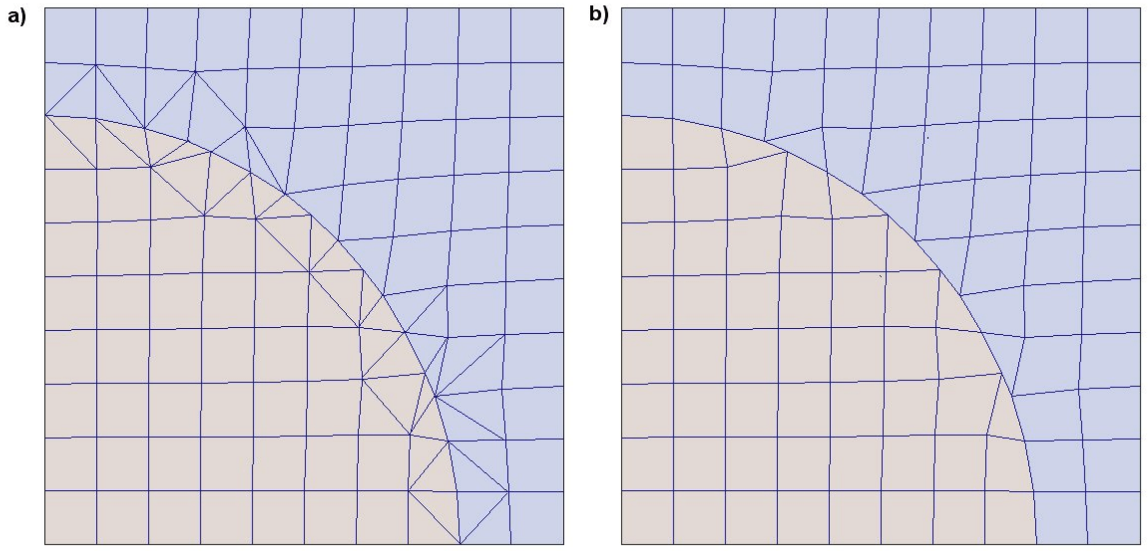


Figure 2.2. Two-dimensional representation of a curved surface with (*a*) an unstructured mesh and (*b*) a structured mesh.

well-removed from material boundaries, the values may be quite inaccurate on surfaces. Regular meshes generally require a large number of elements for a good representation, and hence the solution run times are long.

In contrast, the elements in a conformal mesh (Fig. 2.1*b*) have unique shapes that follow the boundaries of material surfaces. As a result the field calculations achieve better accuracy with fewer elements. The price for this advantage is that it is considerably more difficult to construct the mesh and to find the coefficients of the linear equation set. Fortunately, you do not have to worry about the extra work. **MetaMesh** handles all tasks without your intervention.

Conformal meshes divide into two categories: *structured* and *unstructured*. Figure 2.2 illustrates the difference. Both of the two-dimensional meshes conform to a section of a circular boundary. The unstructured mesh mixes element shapes (quadrilaterals and triangles). The number of elements surrounding a node and the number of neighboring nodes may vary throughout the mesh. In contrast all elements of the structured mesh are quadrilaterals although some may look like triangles because two sides are parallel. Four elements surround each node and each node connects to four neighbors.

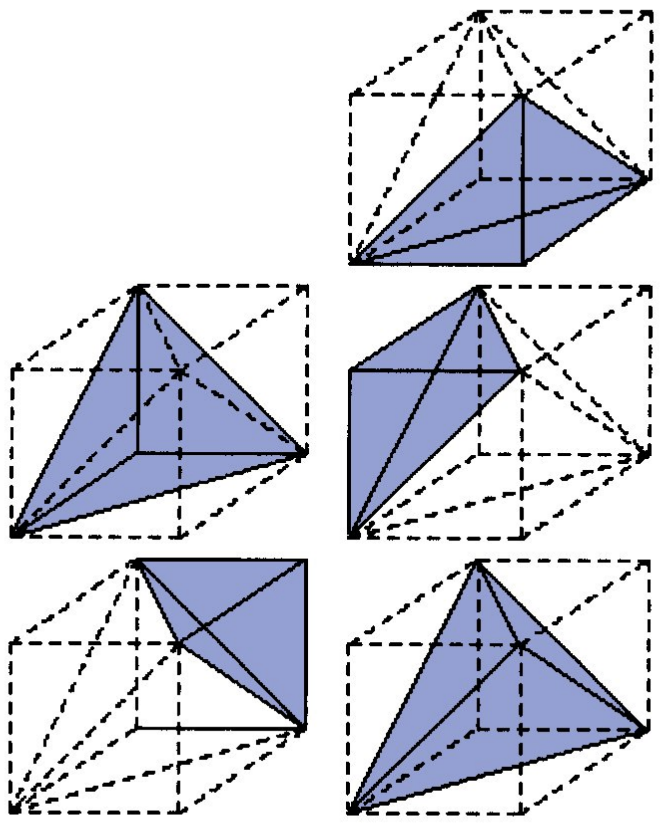


Figure 2.3. Filling a box with five dissimilar tetrahedrons.

With the exception of **MetaMesh**, commercially-available three-dimensional conformal mesh generators create unstructured meshes. The primary reason is that it is quite difficult to fit an arbitrary three-dimensional geometry while preserving strict logical structure. It is easier to patch elements where they are needed. The unstructured approach has one advantage. Because a volume can be divided into an unlimited number of elements, there is more flexibility to employ *variable resolution* (differences in average element sizes) to represent small features. This flexibility comes at a price. The nonuniform interconnections between elements adds to the complexity of solution programs and increases their run time. A structured mesh minimizes the matrix bandwidth of the linear equation set and allows more efficient use of random-access memory. Furthermore, mesh searches can be based on simple index operations. Fast searches are critical in applications such as particle tracking that require repetitive field interpolations.

The final choice is the shape of the three-dimensional elements. There are two practical options: tetrahedrons and hexahedrons. The mathematical operations associated with more complex shapes are unwieldy and are therefore mainly limited to specialized research codes. Tetrahedrons are a popular choice for finite-element programs because the field expansions within an element can be expressed as simple analytic functions. In contrast, the field expansions in hexahedron elements must be determined by three-dimensional numerical integrations.

A drawback of tetrahedrons for structured meshes is that there is no simple way to fill up space. For example, it is not possible to fill a box with a set of similar tetrahedrons. Figure 2.3 illustrates one possible choice for the division using five tetrahedrons. Clearly the indexing scheme of a structured mesh based on tetrahedrons will be complex. We decided to use hexahedron elements to achieve maximum efficiency in the solution programs. Although individual element shapes may be complex, the indexing scheme follows the simple Cartesian logic of a cubic mesh. The required numerical integrations (performed once at the beginning of a solution) do not substantially extend the run time. Furthermore, the finite-element equations of a hexahedron mesh with coupling to 26 neighbors ensure higher-order accuracy than those derived from tetrahedrons with linear shape functions.

2.2 MetaMesh strategy

MetaMesh addresses the challenge of creating ideally-structured conformal meshes by proceeding in ordered steps to the final refined product. We will review the procedure in the section and give details in following sections. For illustration we will consider an example: a mesh that represents an elbow in a high-pressure hydraulic system (`ELBOW_DEMO.MIN`). The fitting is constructed by superimposing simple shapes.

Stage 1 - logical mesh definition. Using information in the global section of the input script, **MetaMesh** defines a solution volume and divides it into box elements. The regular mesh is called the *foundation mesh*. The quality of the final mesh depends on the initial division of the solution volume. Clearly the elements should be small enough to resolve significant details. On the other hand, **MetaMesh** is forgiving and will usually give a useful starting mesh even if the element sizes are not ideal. You can tune the mesh later. For the example, we pick a solution volume in the range [$0.0 \geq x,y \geq 5.0$, $0.0 \geq z \geq 2.0$] and divide it into cubes with sides of length 0.125.

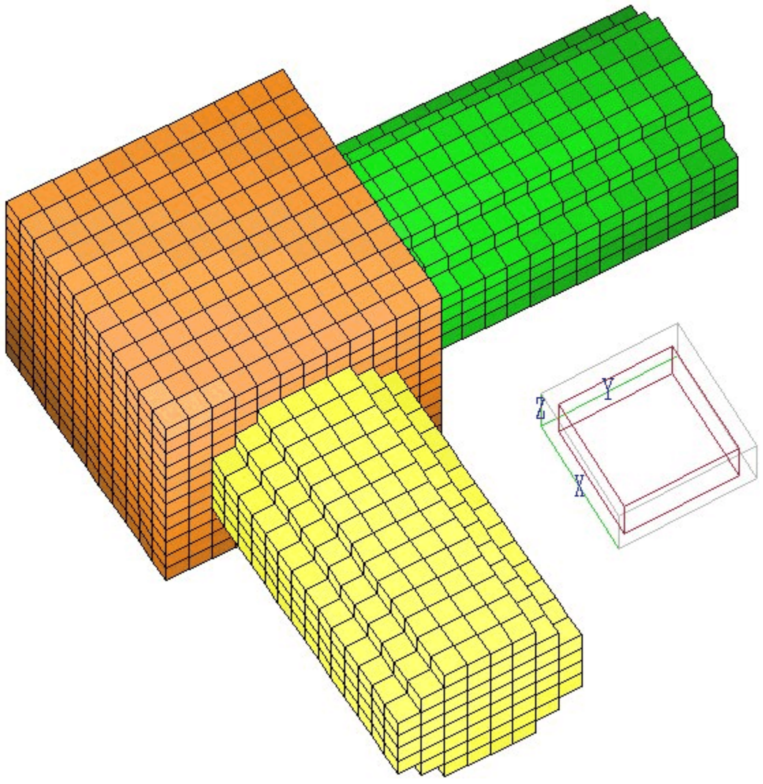


Figure 2.4. Example `EBLOW_DEMO` at the volume-fitting stage

Stage 2 - volume fitting. Based on geometric specifications in the input script, **MetaMesh** assigns region numbers to the elements of the foundation mesh for the best representation of objects. Complex shapes are built from elementary volumes called *parts*. The program checks each element to see if its center-of-mass lies inside the geometric definition of the part. The procedure is identical to that used in **Mesh3** (of the **Maze** program series) for regular (box) meshes. In fact, you could apply **MetaMesh** as a regular mesh generator by stopping at this stage. You could also create mixed meshes with regular and conformal features by applying the operations of subsequent stages only to selected parts. Figure 2.4 shows the example mesh after volume fitting. The region representing the elbow is constructed of three parts - the cubical block (orange) and two cylindrical tubes (yellow and green). By coincidence, the surfaces of the foundation-mesh elements constituting the cube correspond to the desired surface. This condition would not hold if, for instance, the cube were tipped at an angle. In contrast, the elements of the cylinders give poor representations of the tube surfaces.

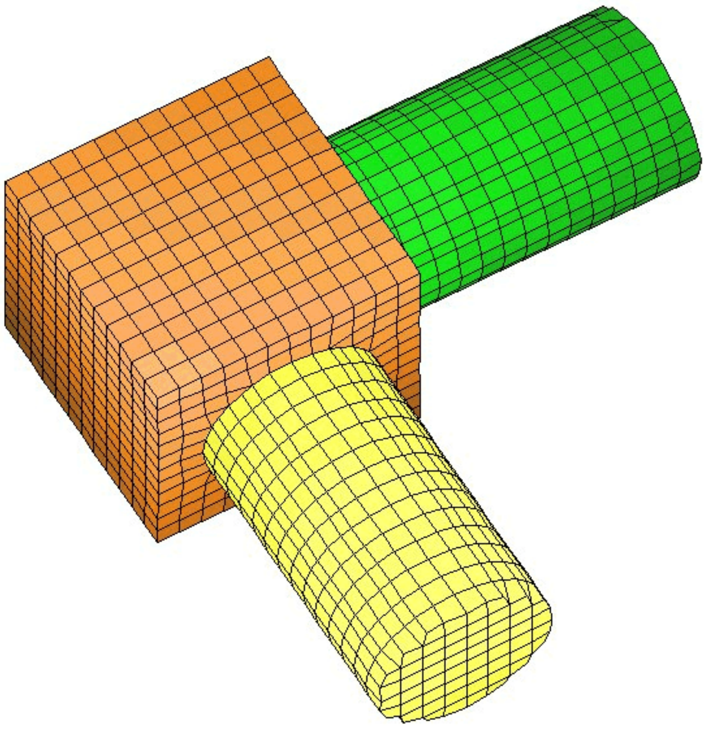


Figure 2.5. Simulation `ELBOW_DEMO` after surface fitting.

Stage 3 - surface fitting. A statement of the type `Surface Region 5` in the specification of a part signals that **MetaMesh** should proceed to the surface-fitting stage. The sample statement indicates that the program should fit all nodes of the part that lie on the shared boundary with elements that have region number equal to 5. The program first identifies affected nodes, and then moves them toward the surface of the part. Shifts are combined with smoothing operations on nearby nodes to minimize element distortions. In the example, the definitions of both tubes contain the statements `Surface Region 1`. Here, the elements of Region 1 constitute the part of the solution volume outside the elbow (air). Figure 2.5 shows the state of the mesh after surface fitting.

Stage 4 - edge fitting. Surface fitting involves moving nodes to the closest geometric surface of the part. As shown in Fig. 2.5, the procedure may leave bevels on the edges. The effect is generally unimportant for simulations that involve the Poisson, diffusion or Helmholtz equations because calculated quantities are undefined on sharp edges. Nonetheless, **MetaMesh** has an additional capability for occasions where you want a precisely-defined edge. When the option `Edge` appears in the Surface

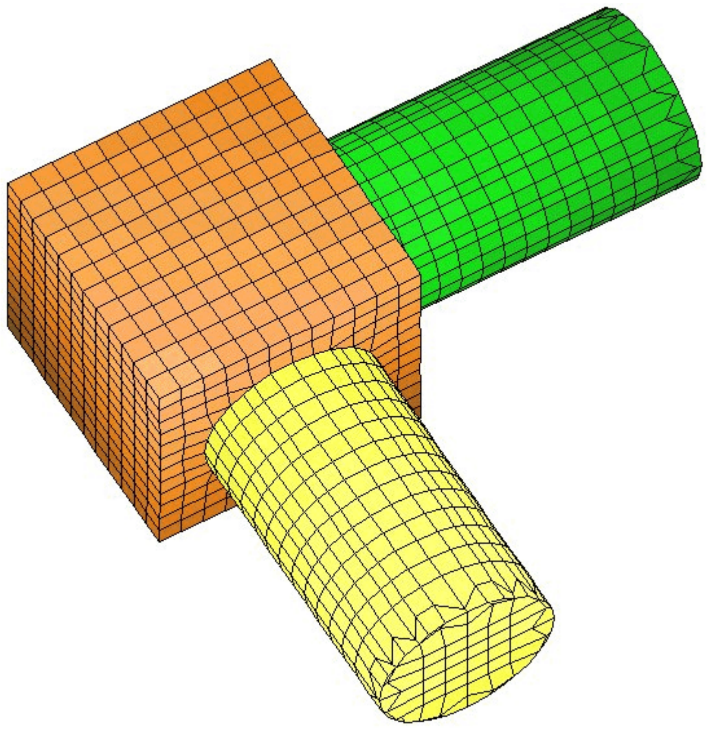


Figure 2.6. Simulation `ELBOW_DEMO` after surface and edge fitting.

statement, **MetaMesh** generates a list of nodes adjacent to the edges and then shifts them incrementally to reduce the bevel effect. Figure 2.6 shows the state of the mesh with edge fitting on both cylindrical parts.

Stage 5 - node fitting. It is also possible to set the region identity and positions of individual nodes in the mesh after processing of volumetric parts is complete. This feature is useful to define special boundary conditions and to represent wires, grids and foils in electrostatic solutions.

Stage 6 - integrity checks and corrections. After all fitting and smoothing operations, **MetaMesh** analyzes the element set. The program detects any elements that have been logically inverted during the fitting process and corrects node positions to ensure a valid mesh. **MetaMesh** also reports the results of volume and surface integrals over regions and the solution volume as an indication of fitting accuracy.

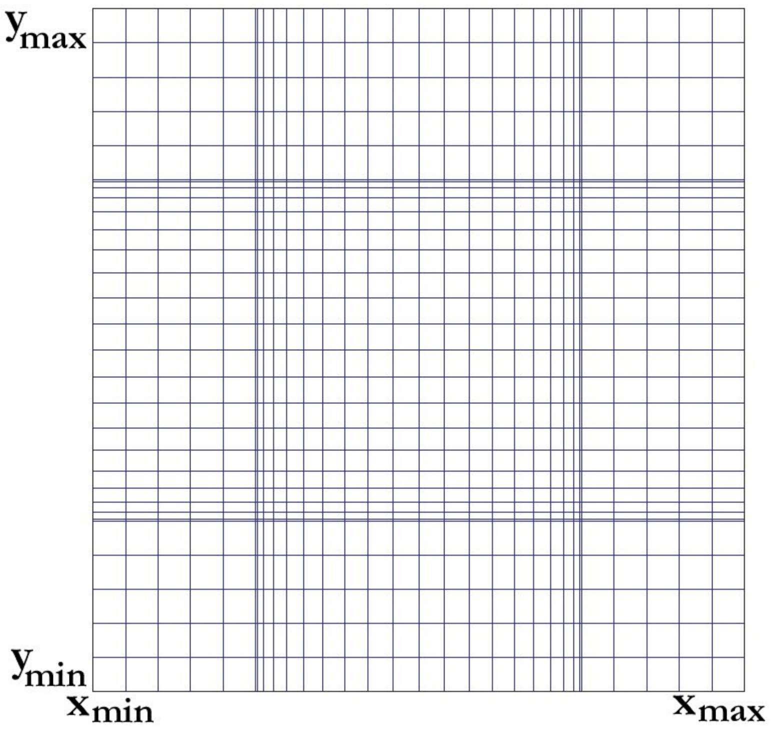


Figure 2.7. Foundation mesh projected in the *z* plane

2.3 Foundation mesh quantities

Finite-element solutions in the **AMaze** programs are performed in a bounded solution volume with sides parallel to the Cartesian axes. The size of the box along the three axes is given by the quantities x_{min}, x_{max}, y_{min}, y_{max}, z_{min} and z_{max}. The circumscribing box may have different lengths along the axes. Figure 2.7 shows a slice of a mesh normal to the z-axis.

The mesh is initially divided into brick elements that constitute the foundation mesh. Element boundaries along the three axes are defined by the following arrays:

```
x(0),...,x(I),...,x(IMax),
y(0),...,y(J),...,y(JMax),
z(0),...,z(K),...,z(KMax).
```

The total number of elements in the solution box is

```
NMax = IMax*JMax*KMax.
```

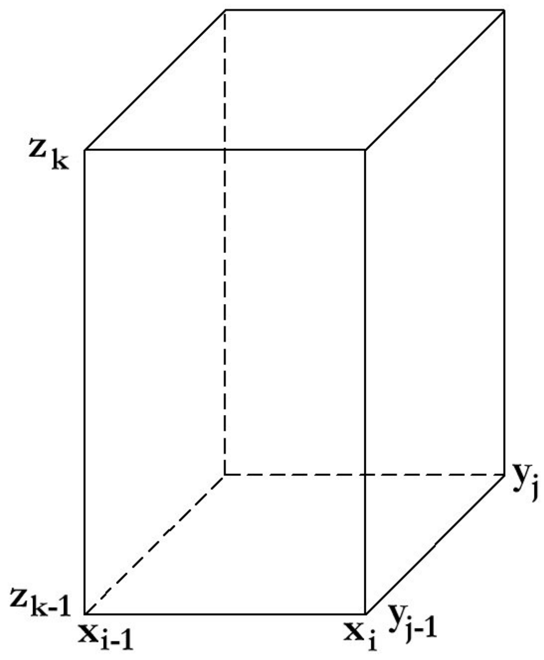


Figure 2.8. Boundaries of the foundation element `[I,J,K]`.

The **AMaze** programs use dynamic memory allocation, so the size of a mesh is limited only by the installed memory of your computer.

In both the foundation and conformal mesh stages, an element has a unique set of indices, `[I,J,K]`. Each element has eight nodes:

```
[I-1,J-1,K-1], [I,J-1,K-1], [I-1,J,K-1],
[I,J,K-1], [I-1,J-1,K], [I,J-1,K],
[I-1,J,K], [I,J,K]
```

Each element shares a boundary with the following six elements:

```
[I+1,J,K], [I-1,J,K], [I,J+1,K],
[I,J-1,K], [I,J,K-1], [I,J,K+1]
```

As shown in Fig. 2.8, element `[I,J,K]` of the foundation mesh extends from `x(I-1)` to `x(I)` along *x*, from `y(J-1)` to `y(J)` along *y* and from `z(K-1)` to `z(K)` along *z*. The sizes of elements of the foundation mesh may vary along the axes. Size variations can be used to improve resolution in critical areas and to provide a better match to the boundaries of objects.

2.3 Assigning element and node region numbers

One of the primary tasks in the mesh generation process is to assign region numbers to elements and nodes. The region number, an integer in the range 1 to 127, identifies the element or node. Physical properties are associated with the region numbers in subsequent **AMaze** solution programs. For example, a set of elements with the same region number may represent a dielectric support in an electrostatic solution. A set of nodes and elements with the same region number may represent a shaped electrode. Finally, a set of nodes with the same region number may represent a wire, sheet or grid.

During volume-fitting, **MetaMesh** processes the parts in the script in the order in which they appear. Part numbers are assigned sequentially. For each part, the program checks the elements of the foundation mesh to determine whether the center-of-mass of an element lies within the specified part boundaries. If the element is inside, **MetaMesh** assigns the part number and associated region number to the element. The program also assigns the current region number to nodes connected to the element. ***It is important to note that region and part numbers over-write any previous definitions***.

2.4 Application example

A good way to illustrate the logic of mesh processing in **MetaMesh** is to walk through a specific example. Figure 2.9 shows the geometry defined by the file `MESH_LOGIC.MIN`, an offset spherical electrode inside a shaped vacuum chamber. We have included a hole through the electrode to illustrate how to drill parts. The chamber has radius 5.0 cm and an axis that lies on the z axis of the simulation space. Table 2.1 lists the complete input script.

The foundation mesh is described by the statements:

```
XMesh
  -5.25  5.25  0.25
End
YMesh
  -5.25  5.25  0.25
End
ZMesh
 -0.25 10.00  0.25
 10.00 20.00  0.50
End
```

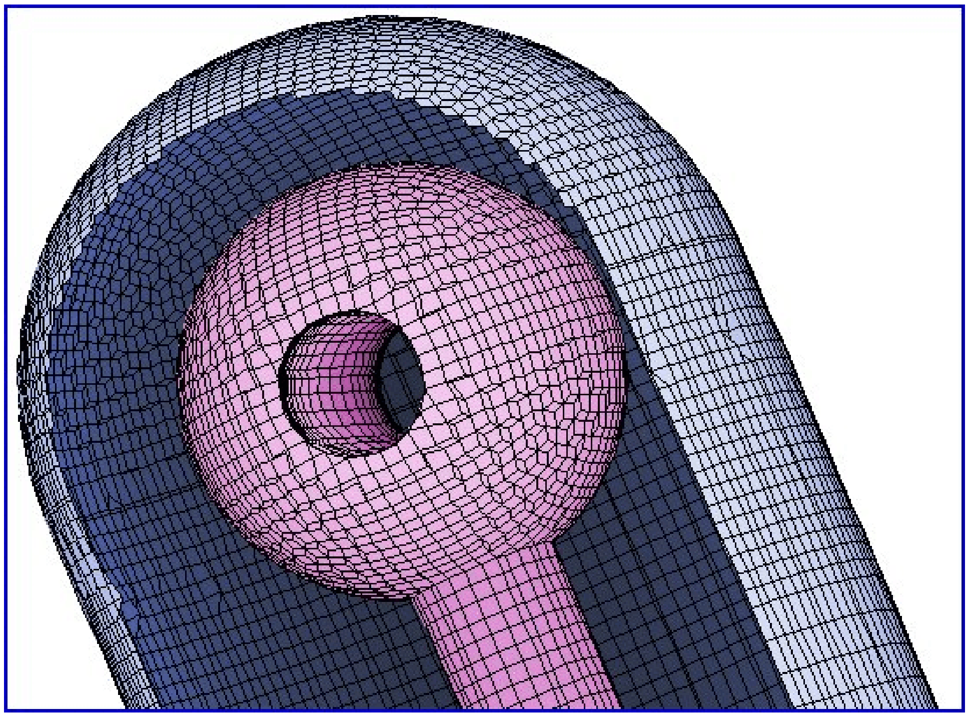


Figure 2.6. Cutaway view of the geometry of simulation `MESH_LOGIC.MIN`. Blue facets represent the inner wall of the vacuum chamber. Violet facets show the surface of the electrode.

Although the inside wall of the vacuum chamber is cylindrical, the solution volume must be a box. We choose the half-widths of the box in *x* and *y* to be one element larger than the chamber radius. The element size of 0.25 cm is sufficient to resolve the smallest objects in the simulation (1.0 cm scale size). Along the *z* axis the elements have width 0.25 cm near the electrode and increase to 0.50 cm at larger values of *z*.

The remainder of the file defines five *Parts*. The first part is a box that represents the grounded vacuum chamber. It fills the entire solution volume:

```
* Vacuum chamber (ground)
Part 1
  Type Box
  Name GroundedWall
  Region 1
  Fab 10.50 10.50 100.0
End
```

Note that we have used the `Name` command and a comment line to identify the part. Good documentation is valuable if you return to a

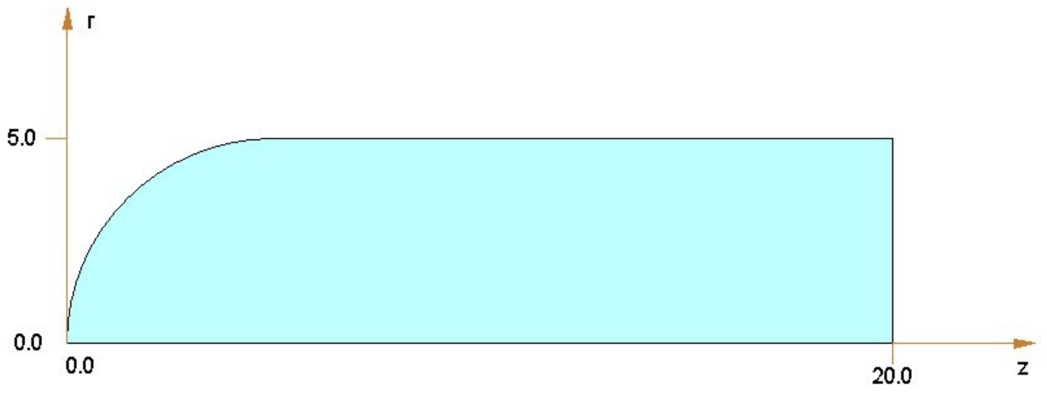


Figure 2.10. Turning cross-section, Part 2 of `MESH_LOGIC.CIN`.

simulation or if you use the script as a template for future work. In processing the Part, **MetaMesh** sets *RegNo* = 1 on all elements and nodes in the solution volume. In the **HiPhi** program, *RegNo* 1 will be associated with the fixed potential $\phi = 0.0$. In the `Fab` command, note that the length of the box along *z* is larger than the solution space. **MetaMesh** ignores any sections of a part that extend outside the solution volume.

The next part is the inner volume of the shaped vacuum chamber. This part is more sophisticated and illustrates the versatility of **MetaMesh** input scripts.

```
* Vacuum
Part 2
  Type Turning
    L 20.00 0.00 20.00  5.00 SE
    L 20.00  5.00  5.00  5.00 SE
    A  5.00  5.00  0.00  0.00  5.00  0.00 SE
    L  0.00  0.00 20.00  0.00 SE
  End
  Region 2
  Fab 0.0  360.0
  Surface Region 1 Edge
  Coat 1 1
End
```

The processing operation assigns elements and nodes to *RegNo* = 2. This region is treated as a dielectric with $\epsilon_r = 1.0$ in **HiPhi**. The part is a *Turning* — the type of object that you would create on a lathe. The data lines beginning with *L* and *A* are a series of arc and line vectors that outline a shape in *z*-*r* space. The shape is then rotated about the *z* axis of the assembly space to define a volume. In the absence of `Shift` or

`Rotate` commands, the *z* axes of the assembly and simulation space are coincident. Figure 2.10 shows the shape defined by the five data lines. Processing of Part 2 will overwrite any previously-defined parts. The effect is to carve elements out of the metal block (*RegNo* = 1) to define an approximately cylindrical volume with 5.0 cm radius with a hemispherical termination near *z* = 0.0. The command `Surface Region 1 Edge` instructs the program to shift nodes to match the surfaces and edges of the turning. The operation applies only to nodes connected to element facets that lie on the border between elements of the current part and elements with *RegNo* = 1. Furthermore, the shifts are performed only on nodes lying on vectors of the turning marked *S* or *SE*.

After volume-fitting all nodes connected to elements of Part 2 are marked as *RegNo* = 2. We want nodes on the inner face of the vacuum chamber to be marked *RegNo* = 1 so that they assume the fixed-potential condition in **HiPhi**. The command `Coat 1 1` directs the program to change the region number of all nodes connected to facets of the current part on the border with elements marked *RegNo* = 1 to *RegNo* = 1. The `Coat` command appears in many hole-drilling operations. A statement like `Coat 1 3` can be used to define special node surfaces such as the emission regions used in **OmniTrak**.

The next two parts define the high-voltage electrode (*RegNo* = 3), a support rod with a spherical tip.

```
* Support rod
Part 3
  Type Cylinder
  Region 3
  Fab 1.00 15.00
  Shift 0.50 0.00 12.50
  Surface Region 2
End

* Spherical electrode
Part 4
  Type Sphere
  Region 3
  Fab 3.00
  Shift 0.50 0.00 5.00
  Surface Part 2
End
```

`Shift` commands are applied to Parts 3 and 4 to move them to their correct locations along *z* and to shift the entire assembly a distance 0.50 cm in *x*. Note that the rod and sphere overlap. This is not a problem because **MetaMesh** creates smooth transitions where parts intersect. The rod contains the command `Surface Region 2`. The command instructs the program to fit nodes on facets that border the vacuum region and to leave any other nodes undisturbed. When processing is complete all nodes connected to elements with *RegNo* = 3 will have same region number. This is the correct condition for the subsequent electrostatic solution where *RegNo* = 3 is a fixed potential region (ϕ = 50 kV). In this case the `Coat` command is unnecessary. Note that the sphere (*PartNo* = 4) has an alternate form of the fitting command `SURFACE PART 2`. In this case the code fits only nodes that lie on the boundary with elements that have *PartNo* = 2, the *external* vacuum region. If we used the command `SURFACE REGION 2`, then **MetaMesh** would have included nodes along the internal cylindrical hole. An attempt to fit these nodes to the surface of the sphere would have resulting in severe mesh distortion.

The final operation (Part 5) is to drill a hole through a diameter of the spherical electrode:

```
* Hole through electrode
Part 5
  Type Cylinder
  Region 2
  Fab 1.0 7.0
  Rotate 0.0 90.0 0.0
  Shift 0.5 0.0 5.0
  Surface 3 1.00
  Coat 3 3
End
```

The elements of this part have the region number of vacuum (*RegNo* = 2). The `Fab` statement defines a cylinder of radius 1.0 cm and length 7.0 cm. We make the length larger than the 6.0 cm diameter of the sphere to ensure that the hole penetrates completely. By convention, cylinders are created in **MetaMesh** with their axis parallel to the *z* axis of assembly space. The command `Rotate 0.0 90.0 0.0` rotates the Part 90° about the *y* axis so that cylinder is parallel to the *x* axis of the simulation space. The `Shift` operation, performed after the rotation, moves the Part to the same axial location as the center of the sphere and offsets it 0.5 cm in *x*. The command `Surface 3 1.00` shifts nodes on facets bordering the metal electrode to match the cylindrical shape. The extra parameter controls the surface and edge fitting tolerance. The value 1.00 corresponds

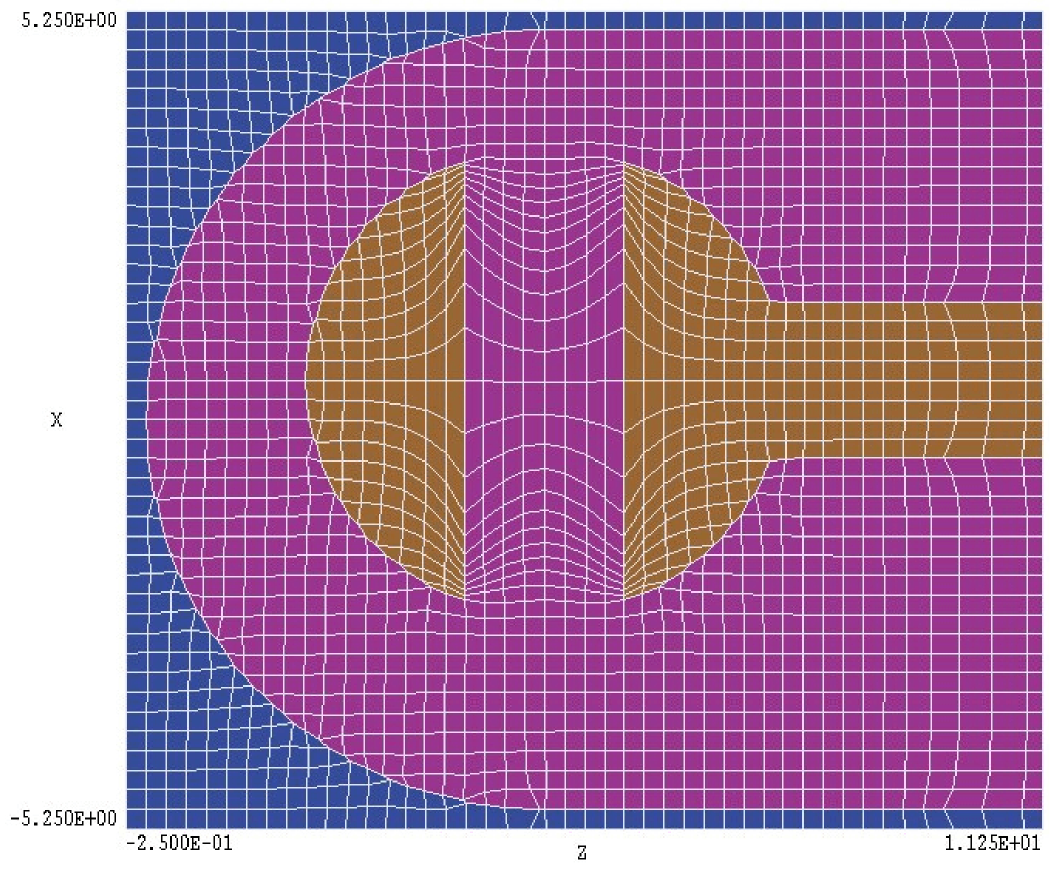


Figure 2.11. Orthographical view of a portion of the finished mesh in the plane y = 0.0.

to an exact fit. The default tolerance of 0.90 gives some flexibility to the fitting process. In the present case, exact fitting is necessary to prevent bowing of the hole inside the electrode. Finally, the command `Coat 3 3` assigns all nodes on the inside of the hole to the fixed potential *RegNo* = 3. Figure 2.11 shows an orthographic view of the finished product.

2.5 Symbolic part and region numbers

The direct use of region and part numbers in the `Part` commands `Region`, `Surface` and `Coat` is acceptable for relatively simple assemblies. On the other hand, dealing with numbers can be inconvenient if you want to reorder the parts of a complex assembly. In this case, you would need to change all part references to reflect the new order. Reordering is also a problem if you want to insert part assemblies from a library to represent standard objects. For these situations **MetaMesh** has a useful capability to recognize with symbolic region and part numbers.

Table 2.1. Application example script MESH_LOGIC.MIN

```
* File Mesh_Logic.MIN
Global
  XMesh
    -5.25  5.25  0.25
  End
  YMesh
    -5.25  5.25  0.25
  End
  ZMesh
   -0.25 10.00  0.25
   10.00 20.00  0.50
  End
  RegName 1 Vacuum chamber
  RegName 2 Vacuum
  RegName 3 Electrode
End

* Vacuum chamber (ground)
Part 1
  Type Box
  Region 1
  Fab 10.50 10.50 100.0
End
* Vacuum

Part 2
  Type Turning
    L 20.00 -5.00 20.00  5.00 SE
    L 20.00  5.00  5.00  5.00 SE
    A  5.00  5.00  0.00  0.00  5.00
0.00 SE
    A  0.00  0.00  5.00 -5.00  5.00
0.00 SE
    L  5.00 -5.00 20.00 -5.00 SE
  End
  Region 2
  Fab 0.0  360.0
  Surface 1 Edge
  Coat 1 1
End
```

```
* Support rod
Part 3
  Type Cylinder
  Region 3
  Fab 1.00 15.00
  Shift 0.50 0.00 12.50
  Surface 2 Edge
End

* Spherical electrode
Part 4
  Type Sphere
  Region 3
  Fab 3.00
  Shift 0.50 0.00 5.00
  Surface 2
End

* Hole through electrode
Part 5
  Type Cylinder
  Region 2
  Fab 1.0 7.0
  Rotate 0.0 90.0 0.0
  Shift 0.5 0.0 5.0
  Surface 3 1.00
  Coat 3 3
End

EndFile
```

To illustrate the association of names with region numbers, consider the following statements in the WALKTHROUGH example of Table 1.1:

```
RegName 1 TransformerOil
RegName 2 OilTankWall
RegName 3 Insulator
RegName 4 HighVoltageLead
```

The first statement defines a string, *TransformerOil*, that can be used in place of the number 1 in commands in subsequent PART sections where a region number is required. For example the following statements would have the same effect:

```
Surface Region 1 Edge
Surface Region TransformerOil Edge
Surface Region TRANSFORMEROIL Edge

Coat 2 2
Coat OilTankWall OilTankWall
```

The first set of commands illustrates that the interpretation of region and part names is case insensitive. Region names may be up to 20 characters in length and may not contain any of the standard delimiters discussed in Chap. 3. The underscore symbol is acceptable.

Symbolic forms are useful if you want to change the order of regions or add new regions. In this case, any changes in the REGNAME commands will automatically be updated in subsequent PART operations. Symbolic fitting commands are generally more readable in a long, complex script. There are two addition motivations to include a complete set of REGNAME commands in the GLOBAL section:

- ■ Region names are used within **MetaMesh** in dialog boxes and 2D and 3D plots.
- ■ **MetaMesh** uses name information to make a formatted listing of regions in the file RunName.MLS. You can paste this information into **HiPhi** or **Magnum** script to document the identity of regions.

The following list was generated in WALKTHROUGH.MLS for the WalkThrough example:

```
Analysis of Regions in the script
   Number of Regions:   4
   Assignment of Region names
*  NReg      RegName
*  ==============================
*     1      TRANSFORMEROIL
*     2      OILTANKWALL
*     3      INSULATOR
*     4      HIGHVOLTAGELEAD
```

Note that the program has placed comment symbols at the beginning of the listing lines.

You can assign a name to a part by including the `NAME` command in the `PART` section. For example, suppose the following section is the fifth to appear in the script:

```
Part
  Name HVLeadInsulator
  Type Cylinder
  Region HighVoltageLead
  Fab   0.500  2.50
  Shift  0.000   0.000  -0.250
  Surface Region Insulator Edge 0.90
  Surface Region TransformerOil Edge 0.90
End
```

In this case, the string *HVLeadInsulator* would assume the value of 5 in *SURFACE* commands referring to part numbers. For example, the following two statements (in a different *PART* section) would have the same effect:

```
Surface Part 5
Surface Part HVLeadInsulator
```

The difference is that if you change the order of parts in the script, the second form ensures that the reference to the part is automatically updated. **MetaMesh** summarizes information on the association of part names with numbers in the listing file `RunName.MLS`. For example, the following entry is created for the input file `WalkThroughSymbolic.MIN`.

```
Analysis of Parts in the script
   Number of Part sections:   6
   Assignment of Part names
   NPart     PartName
   ===========================
      1      OILINSIDETANK
      2      OILTANKWALL
      3      INSULATOR
      4      HVLEADINSULATOR
      5      HVLEADEXTENSION
      6      HVLEADELBOW
```

2.6 Mesh generation guidelines

The MESH_LOGIC example illustrates that you can create sophisticated objects in **MetaMesh** with moderate effort. Inspection of the script in Table 2.1 shows that few geometric calculations were necessary to achieve the complex shapes of Fig. 2.9. Preparation of input scripts can be easy and even pleasant if you have experience with mechanical assemblies. Here are a few suggestions for efficient mesh generation:

■ Build the script step-by-step. Add and test a new part before proceeding to the next.

■ Use comments to organize the assembly. You can also add any amount of descriptive text after the ENDFILE command.

■ Use the cut and paste operations of your editor to build repetitive structures, changing the SHIFT and ROTATE parameters for specific instances.

■ Save important subassemblies in a file library.

■ In fitted regions, allow a sufficient number of elements to resolve shapes. Attempts to fit single elements to complex surfaces will result in distortion. In this case, **MetaMesh** may report logically-inverted elements, or you may observe numerical instabilities in a subsequent solution program.

■ Avoid fitting surfaces whose shape is not essential for the accuracy of the solution. This precaution reduces chances for error and speeds convergence of solution programs.

■ Remember that the order of parts in the script is important.

■ Naming regions and parts improves readability and makes it easier to modify a script.

Chapter 3. Structure of the assembly script

The **MetaMesh** assembly script contains commands to set the sizes of elements in the foundation mesh and to define the shapes of a series of *Parts* in the solution space. The script is a text file that can be created or inspected with any ASCII editor. You can use the built-in editor of **MetaMesh** to create or to modify scripts. Assembly files have a name of the form `RunName.MIN`, where `RunName` is a descriptive file prefix from 1 to 20 characters in length.

Assembly scripts consist of a number of active lines with standard commands and parameters. The entries in active lines can be separated by the following delimiters

```
Space
Comma [,]
Tab
Colon [:]
Equal sign [=]
Left parenthesis [(]
Right parenthesis [)]
```

Any number of delimiters may be used in a line. Blank lines and comment lines are ignored. The file may also contain any number of comment lines that begin with an asterisk `[*]`. Parameters may be strings, integers or real numbers. The input of real numbers is flexible. The following formats are valid:

```
1.000
5.67E6
6.8845E+09
5
```

The last number is interpreted as 5.0.

The **MetaMesh** assembly script has the following general form:

```
GLOBAL
 (Global commands)
END

PART 1
 (Part 1 commands)
END

PART 2
 (Part 2 commands)
END

...

PART N
 (Part N commands)
END

ENDFILE
```

The purpose of the `GLOBAL` section is to set up the foundation mesh. The file may contain only one `GLOBAL` section and it must appear at the beginning. The following set of nine commands may appear in the `GLOBAL` section:

```
XMesh
  -5.0 5.0 0.5
End
YMesh
  -5.0 5.0 0.5
End
ZMesh
  -10.0 10.0 0.5
End
AxisSmooth X 10
PreSmooth 15
Smooth 25
ASCII
RegName 1 SolutionVolume
```

Chapter 4 describes the functions of the commands. Within the `GLOBAL` section the commands may appear in any order. **MetaMesh** reads the full set of commands before starting construction of the foundation mesh.

The assembly file may contain any number of `PART` sections (limited by the installed memory on the computer). Parts are the basic 3D building blocks in **MetaMesh**. Several parts may be combined to construct a complex *Region*. The order in which `PART` sections appear in the file is ***important***, because a part overwrites the definition of any previously-defined parts in the shared space. The over-write principle is used to drill holes, bevel edges, and join complex shapes. The following set of commands may appear in a `PART` section

```
Type Cylinder
Region 15
Shift 0.00 1.54 0.00
Rotate 90.0 45.0 0.0
Fab 5.56 20.00
Surface Region 4 Edge 0.95
Surface Part 5
Coat 1 3
```

Within a single part section, the commands may appear in any order. Chapters 5 through 8 give detailed information on commands of the `PART` sections. The `ENDFILE` command follows all `PART` sections and signifies that all data have been entered. The program ignores lines after `ENDFILE`, so you can append any amount of text at the end of the file.

The **MetaMesh** script represents a highly efficient method to define complex three-dimensional shapes. The geometry construction process requires a set of only thirteen core commands: `XMESH, YMESH, ZMESH, AXISSMOOTH, PRESMOOTH, SMOOTH, TYPE, REGION, SHIFT, ROTATE, FAB, SURFACE` and `COAT`

Chapter 4. Global commands

4.1 Building the foundation mesh

Because **MetaMesh** creates structured meshes, the *solution space* must enclose a box (brick-shaped region). Following Chap. 2, the box has dimensions $x_{\min}$ to $x_{\max}$ along x, $y_{\min}$ to $y_{\max}$ along y and $z_{\min}$ to $z_{\max}$ along z. All elements of the solution space must be filled. The *active solution space* is the set of elements and nodes that actually enter into the physical solution in subsequent programs. For example, suppose you want to simulate electrostatic fields inside a grounded spherical chamber. In this case, the total solution space is grounded metal cube, while the active solution space is an enclosed spherical dielectric region. The arrangement wastes some memory, but has no effect on the solution speed — the programs of the **AMaze** series skip fixed-potential nodes during the solution process. Note that the total solution space need only be large enough to circumscribe the active solution space.

Commands to build the foundation mesh appear in the `GLOBAL` section of the assembly script. The `XMESH`, `YMESH` and `ZMESH` commands define the initial division of the total solution space into box elements. The commands are part of a data set with the following format:

```
XMesh
  -5.00  5.00  0.20
End
```

In the example the single data line states that the solution box extends from -5.0 to 5.0 in the x direction with uniform element size approximately equal to 0.20. Note that dimensions in **MetaMesh** are relative. For example, if you are simulating microelectronic devices you can use dimensions of μm or mils. All **AMaze** solution programs have provisions for converting the coordinates of nodes from **MetaMesh** to the standard SI unit of meters.

A script must include one instance each of the `XMESH`, `YMESH` and `ZMESH` commands. The following example completely defines a foundation mesh:

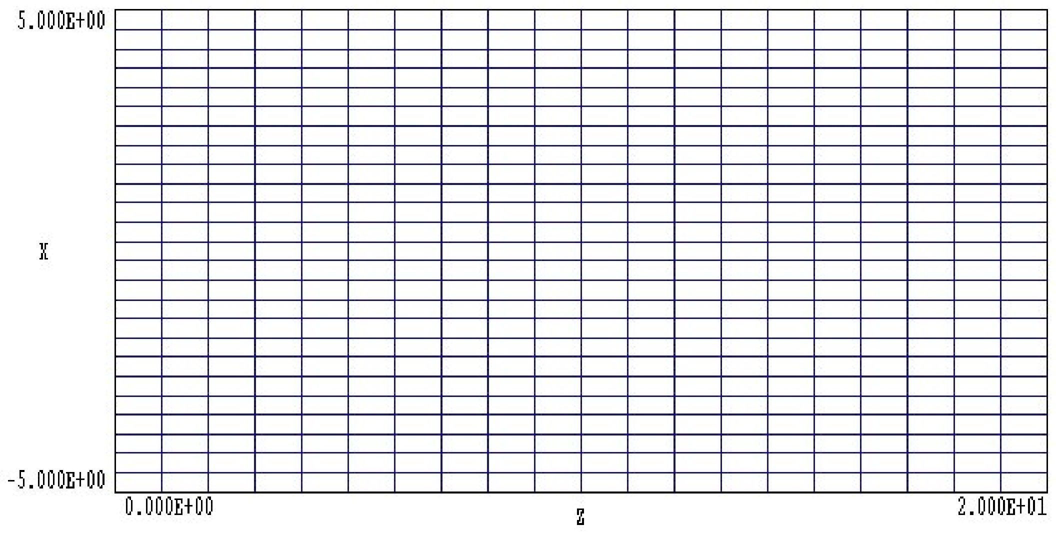


Figure 4.1. Projection of foundation mesh normal to *y* axis - uniform element sizes along *x* and *z*.

```
XMesh
  -5.00  5.00  0.20
End
YMesh
  -5.00  5.00  0.20
End
ZMesh
  0.00 40.00 0.50
End
```

The example shows that you can use different element sizes along each axis. Figure 4.1 shows a projection of the resulting foundation mesh normal to the *y* axis. The following alternate form for the *z* axis data line can be used if you want to specify the number of elements rather than the approximate size:

```
ZMesh
  0.00 40.00 N 80
End
```

The string parameter *N* signals that the following number is an integer equal to the number of uniform elements in the interval.

The `XMesh`, `YMesh` and `ZMesh` command sets may contain multiple data lines to create a variable-resolution mesh:

```
XMesh
 0.00  2.50 0.10
 2.50 20.00 0.25
End

XMesh
 0.00  2.50 N 25
 2.50 20.00 N 70
End

XMesh
 0.00  2.50 N 25
 2.50 20.00 0.25
End
```

The three examples result in the same action. They specify an element size along *x* of 0.10 from 0.00 to 2.50 and a size of 0.25 and from 2.50 to 20.0. The resulting boundaries of the solution volume are x_{min} = 0.0 and x_{max} = 20.0. When there are multiple data lines, it is essential that the intervals appear in order and that they fill a continuous span from x_{min} to x_{max}. As an example, the two command sets

```
XMesh
  -5.00  0.00  0.20
   0.00  5.00  0.40
End
YMesh
  -5.00  0.00  0.20
   0.00  5.00  0.40
End
```

create the mesh (normal to the *z* axis) of Figure 4.2*a*.

It is important to choose good element sizes to ensure successful mesh generation. The finite-element techniques in the **AMaze** solution programs are quite accurate, so the elements need not be extremely small. On the other hand, picking element sizes that are too large can lead to problems.

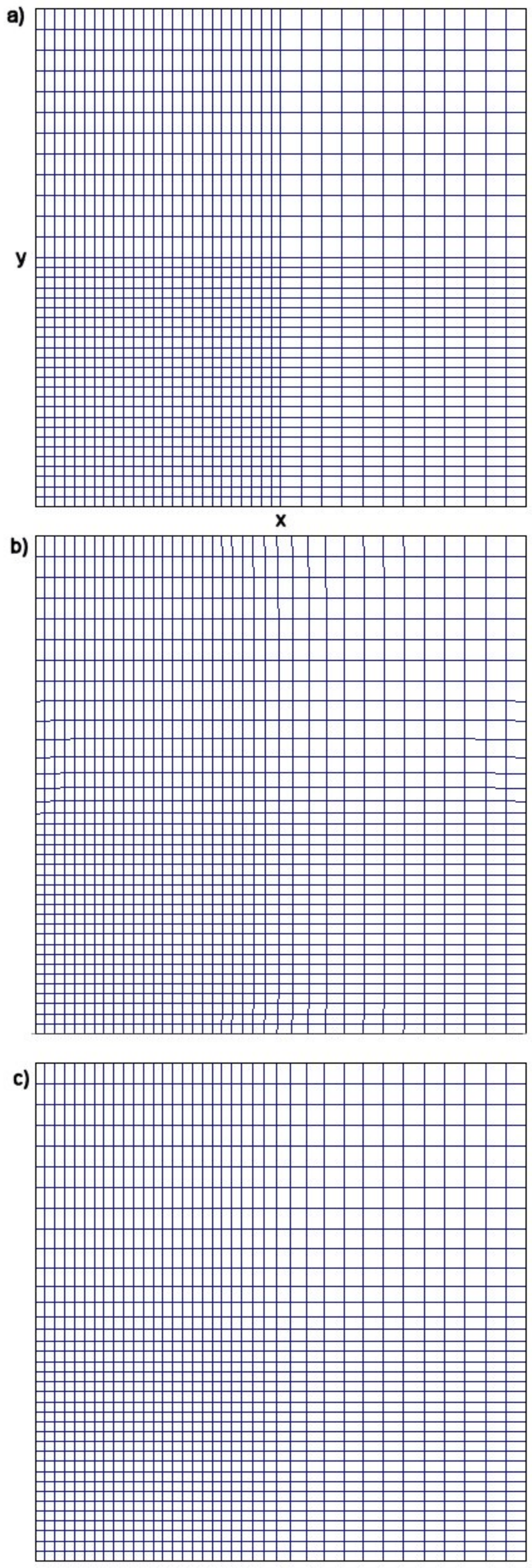


Figure 4.2. Effects of smoothing on a variable-resolution mesh. (*a*) No smoothing. (*b*) 20 cycles of general presmoothing. (*c*) 20 smoothing cycles along the *x* and *y* axes.

Errors may occur if you try to represent a complex part with several surfaces with only one or a few elements. **MetaMesh** will attempt to comply with your request, sometimes leading to distorted or inverted elements. Large elements are not of great concern in regions where there is no surface fitting. In these areas the mesh is approximately regular (*i.e.* box elements). Although small details may be lost, the mesh is always valid and there are no errors in subsequent solution programs.

In summary, the following command is used to define the size of the total solution space and to create the elements of the foundation mesh along the *x* direction:

```
XMesh
  xmin1 xmax1 dx1
  xmin2 xmax2 dx2
 ...
  xminn xmaxn dxn
End
```

Defines the division of the total solution space along *x*. The space extends from $x_{min} = x_{min}^1$ to $x_{max} = x_{max}^n$. Within the region from x_{min}^1 to x_{max}^1, the approximate element size is dx^1, and so forth. The maximum number of data lines is $n = 25$. The intervals along *x* defined by each data line must appear in order from x_{min} to x_{max} and define a continuous set.

Similar definitions apply to the `YMESH` and `ZMESH` commands

```
XMesh
  x1min x1max N N1
  x2min x2max N N2
 ...
  xnmin xnmax N Nn
End
```

Alternate form of the data lines of the `XMESH` command. The string *N* designates that the following integer gives the number of elements in the interval. Note that both forms of data lines can be mixed in a single `XMESH` set.

4.2 Refining the foundation mesh

The next set of commands controls refinement of the foundation mesh and smoothing of the final mesh.

`PRESMOOTH  NPSmooth`

This command has an effect only when variable resolution is defined along one or more axes. (*i.e.*, multiple data lines in the `XMESH`, `YMESH` or `ZMESH` command sets). It response, **MetaMesh** applies smoothing operations to the foundation mesh before volume and surface fitting operations are performed. The smoothing eliminates sharp transitions between regions with different element sizes. Pre-smoothing often helps the fitting process, leading to meshes that give improved accuracy in the solution program. This command gives spatial smoothing of node positions along all axes simultaneously. The integer parameter *NPSmooth* is the number of smoothing cycles. More cycles give more gradual transitions. The default is no pre-smoothing (*NPSmooth* = 0). Figure 4.2*b* illustrates the effect of the command.

`AXISSMOOTH X  NSmoothX`

This command has an effect similar to `PRESMOOTH`. The difference is that smoothing is applied along a single axis. The first parameter is a string that designates the axis [*X*,*Y* or *Z*] and the second integer parameter is the number of smoothing cycles. Larger values give more gradual transitions. Up to three `AXISSMOOTH` commands may appear in the `GLOBAL` section, one for each axis. The default is no axis smoothing. Figure 4.2*c* illustrates the effect of the command.

`SMOOTH NSmooth`

This command instructs **MetaMesh** to perform *NSmooth* cycles of smoothing *after* volume and surface fitting has been completed. Smoothing applies only to nodes that have not been clamped by fitting operations. The choice *NSmooth* = 0 gives no smoothing. Larger values of *NSmooth* give more smoothing. The default value is *NSmooth* = 10.

4.3 Miscellaneous commands

The `REGNAME` command is useful to document your runs and to organize you work within **MetaMesh**.

`REGNAME  RegNo  String`

The parameter *RegNo* is the region number, an integer from 1 to 127. The string is a descriptive word (no delimiters) that you want associated with the region (maximum of 20 characters).

The next command controls whether **MetaMesh** performs automatic mesh corrections (Section 9.4). After all fitting processes, the program checks whether excessive displacements were required to fit elements of the foundation mesh to surfaces and edges of parts. The nodes of distorted elements are relaxed in position to ensure valid shapes. The process may result in smoothed transitions on the edges of parts.

`AUTOCORRECT OFF`
`AUTOCORRECT ON 9`

This command controls automatic correction of meshes by element relaxation. The default mode is *ON* with seven cycles of element relaxation. The option *OFF* deactivates the correction process. For meshes with large numbers of distorted elements, you can increase the number of relaxation cycles by supplying an integer parameter after the string parameter *ON*.

The final command controls format of the output file created by **MetaMesh**. The file has name of the form `FPrefix.MDF` where `FPrefix` is same file prefix used for the assembly script and `MDF` stands for **M**esh **D**efinition **F**ile. To conserve disk space, the file is normally written in a binary format (Chap. 14) that is compatible with all **AMaze** solution programs. If you are using **MetaMesh** as a stand-along utility, it may be more convenient to write the file in ASCII format.

`VOLUMECHECK [OFF, ON]`
`VOLUMECHECK ON`

After generating and smoothing a conformal mesh, **MetaMesh** checks of the integrity of elements by performing surface and (optionally) volume integrals. During the check, the program keeps track of the surface area and volume of regions and records the result in the `MLS` file. The volume integrals are not as sensitive to element errors and take longer. Therefore they are turned *OFF* by default. You can use this command to activate volume integrals to make a more stringent check of new mesh geometries or if you need the volume information. (Default: *OFF*)

`FITTING [ON, OFF]`
`FITTING OFF`

When building a mesh it is sometimes useful to deactivate surface fitting to check the validity of the volume-fitting process. When this command is issued, **MetaMesh** ignores *SURFACE* commands for all parts. Note: use the *#NOFIT* script directive to deactivate fitting of specific parts. (Default: *ON*)

`FORMAT ASCII`

This *GLOBAL* command controls the format of the output file. The string parameter options are *ASCII* and *BINARY*. If the *FORMAT* command does not appear, the default is *BINARY*.

`ELEMFILE [ON, OFF]`
`ELEMFILE ON`

Finite-element calculations using hexahedron elements and the minimum-weighted-residual method involve the precalculation of element integrals. The integrals depend only on the mesh geometry, and are therefore independent of the specific solution program. By default, **AMaze** solution programs compute element integrals on-the-fly for each solution. This activity takes time and may be redundant if you make several solutions with the same mesh geometry. Use the *ELEMFILE ON* command to create a file of element integrals that can be transferred to the **AMaze** solution programs. If an *ELEMFILE* command also appears in the control script for the solution program, the code will read values from the file instead of calculating them. The drawback is that files of element integrals are usually very large (>100 MB) and may consume excessive space on your hard disk if they accumulate. Default: *ELEMFILE OFF*.

Element integral files are quite useful if you want to create your own finite-element programs using **MetaMesh** as a standalone mesh generator. Please contact us for a report on the element file format.

Chapter 5. Commands to define parts

5.1 Three-dimensional assembly procedure

Complex three-dimensional objects (*Regions*) are built in **MetaMesh** by combining simple volumes called *Parts*. Figure 5.1 shows an example where we construct a bulb for a bathroom light fixture from a geometric sum of simple shapes. This chapter covers basic concepts for the volume construction technique. Chapters 6, 7 and 8 discuss specific components of the **MetaMesh** parts bin.

The **MetaMesh** program and script syntax was designed so that the mesh creation process closely resembles the physical fabrication of assemblies in a machine shop. The script is equivalent to the blueprint for the assembly. We manufacture and add parts one-by-one. We must take care to assemble the parts in the correct order. Our workbench is the *assembly space*, a reference coordinate system where parts are created with a fixed orientation. In order to fabricate the part we need to know the geometric type and the dimensions. We then orient the part by applying rotations and move it to its correct position in the solution space.

We need the following information to add a part to solution space:

Part number

An integer that gives the order in which the part is added to the simulation geometry. This number is assigned automatically by the program according to the part's order in the script.

Region number

An integer that identifies the physical object that contains the part. For example, an electrode region with a radius on the end may be constructed from two parts: a cylinder and a sphere. The region number is similar to a specification of the material used to fabricate the part.

Type

A string that specifies the geometric class of the part (*i.e.*, sphere, box, turning,...). Just as we would use different tools to construct parts of different shapes in a machine shop (lathe, milling machine,...), so **MetaMesh** uses different routines to process different part types.

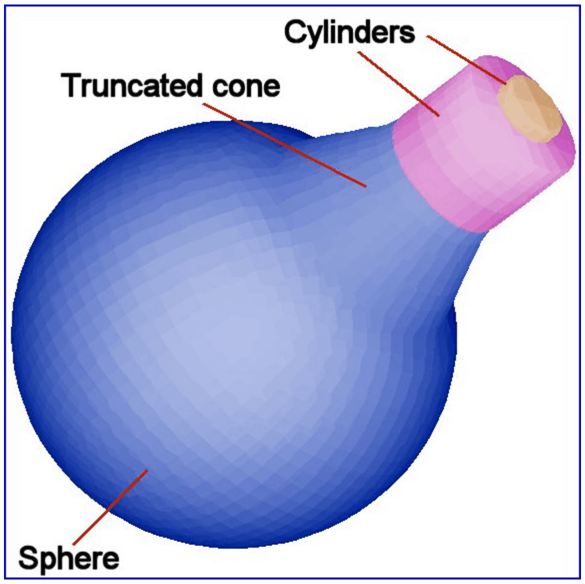


Figure 5.1. Complex shape (lightbulb) constructed from a geometric sum of simple shapes.

Dimensions
One or more real numbers that give the size and shape of the part. The dimensional quantities vary according to the part type. For example, we need the lengths along three sides to define a box.

Rotation
Three rotation angles (real numbers) that tell how to move the part so it will be correctly oriented in the solution space.

Position
Three displacements (real numbers) along the axes that tell where to put the part after it is fabricated and oriented.

5.2 Structure and order of part sections

A *PART* section has the following structure:

```
PART
 (Commands)
END
```

A number *PartNo* is assigned to parts in the order in which they appear in the script. **MetaMesh** reads the full set of parts before starting processing operations. You can change the assembly order by changing the location of the part section in the script. Note that the part numbers of elements and nodes are used within **MetaMesh**. Only region numbers are written to the `MDF` file and passed to solution programs.

If you maintain a library of standard assemblies for your applications, you can paste them in anywhere in the script. The capability is similar to blocks in a CAD program. If you define symbolic part numbers using the `NAME` command (Sect. 2.5), it is not necessary to change part number references in `SURFACE` commands that appear in previous or subsequent part sections.

Part processing in **MetaMesh** is governed by two simple rules:

- Parts are handled in the order in which they appear in the script. A part will over-write any elements in shared spaces that have lower values of *PartNo*.

- When values for *PartNo* and *RegNo* are assigned to an element, the same numbers are assigned to the eight nodes of the element.

The following example illustrates implications of the rules. The two *Part* sections describe the ceramic insulator and copper bus bar of a high-current vacuum feed-through. In a subsequent **HiPhi** solution, the copper rod is assigned a fixed potential. Because potential values are defined at nodes, we must ensure that connected nodes have the same region number as the elements of the rod. This condition occurs automatically if the ceramic appears in the file before the rod.

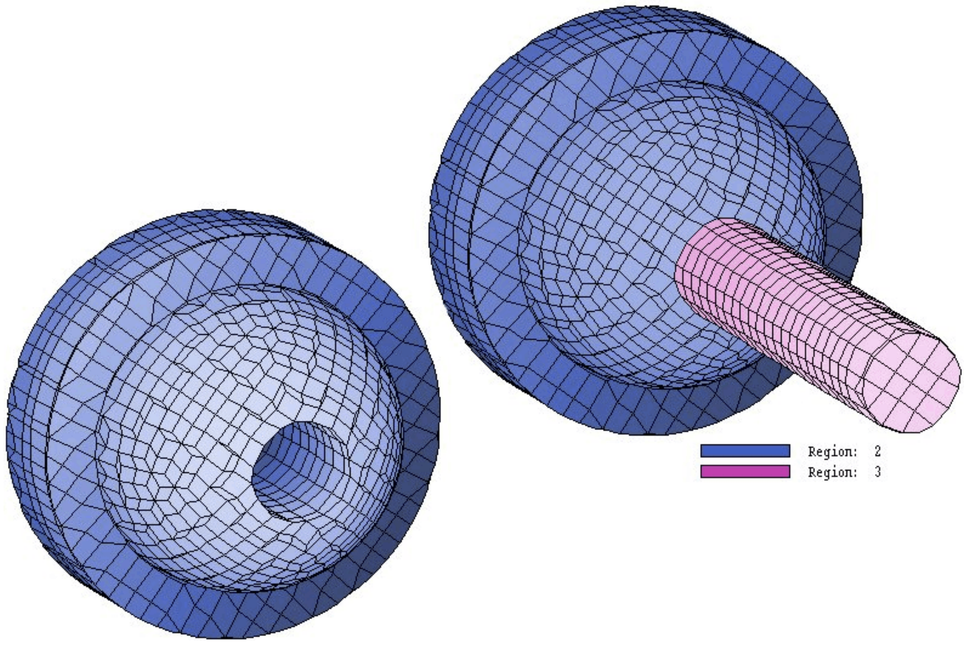


Figure 5.2. Example of parts processing order. The rod (violet) has a higher value of *PartNo* than the insulator (blue). Processing of the rod creates a hole through the insulator.

```
Part 2
* Ceramic bushing  Type Turning
    L  0.000  0.000  0.625  0.000
    A  0.625  0.000  0.250  0.375  0.250  0.00 SE
    L  0.250  0.375  0.250  0.500 SE
    L  0.250  0.500  0.000  0.500 SE
    L  0.000  0.500  0.000  0.000 SE
  End
  Region 2
  Fab 0.0  360.0
  Shift 0.00 0.00 0.00
  Surface Part 1 Edge 1.00
End

Part 3
* Copper bus bar
  Type Cylinder
  Region 3
  Fab 0.125  2.00
  Shift 0.00 0.00 0.75
  Surface Part 1 Edge 1.00
  Surface Part 2 1.00
End
```

Note the numbers 2 and 3 in the PART commands. You can include numbers or any descriptive text following the keyword PART. The information is for reference only – it is ignored by the program. Figure 5.2

shows the resulting mesh. The assembly of Parts 2 and 3 appears on the upper-right side. The lower-left side shows a plot of the surface facets of only the insulator. Note that the addition of the rod creates a hole through the bushing. Because the rod is processed last, all nodes that border the hole are set as *RegNo* = 3 and will automatically assume the fixed-potential condition of the rod.

5.3 Relationship between assembly and solution space

MetaMesh operations are easier to understand if we envision that parts are initially created in an *assembly space*. The assembly space is a coordinate system with the same origin and axes as the coordinates of the solution space. Depending on its type, a part has a specific orientation and position when created in the assembly space. For example, spheres have the centers at the origin of assembly space $[x_a,y_a,z_a] = [0,0,0]$. A cylinder of height h has its axis coincident with the z_a axis and has end faces at $z_a = -h/2$ and $z_a = h/2$. Chapters 6, 7 and 8 list the conventions for different part types. Once a part has been fabricated, we rotate it to the correct orientation in assembly space and then shift it to its location in solution space.

While the shift operation is simple, three-dimensional rotations are more difficult to envision. Section 5.7 discusses the syntax of the *Rotate* command. It involves three angles of rotation about the three Cartesian axes. The sign of the angle follows the right-hand rule. (If you place your thumb parallel to the rotation axis, the fingers of the right hand point in the direction of positive rotation.) Figure 5.3 shows two examples: $\theta_y = +30°$ and $\theta_z = -45°$. Rotation operations are not commutative so the order in which they are performed is important. For example, consider a unit vector initially pointing along z, $\mathbf{u} = (0,0,1)$. A rotation $\theta_x = 90°$ followed by $\theta_z = 90°$ gives a vector pointing along the x axis, $\mathbf{u} = (1,0,0)$. On the other hand, a rotation of $\theta_z = 90°$ followed by $\theta_x = 90°$ gives a vector pointing in the $-y$ direction $\mathbf{u} = (0,-1,0)$. The default order of rotations in **MetaMesh** is θ_x, θ_y then θ_z. It is possible to change the order to achieve any rotation in three-dimensional space with the minimum number of operations.

We will consider a fairly difficult example to illustrate applications of shifts and rotations: a Christmas tree ornament in the shape of a tetrahedron. The base of the tetrahedron lies in the x-y plane with

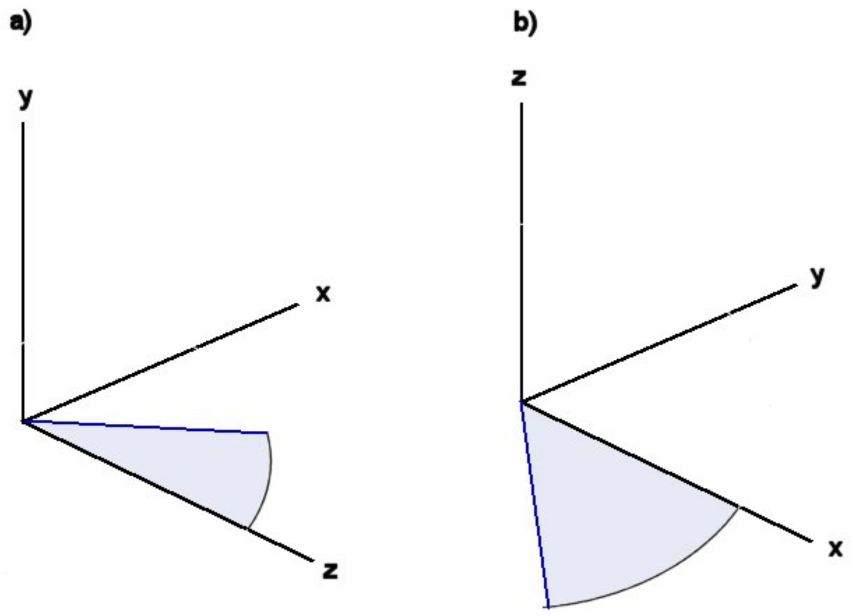


Figure 5.3. Three-dimensional rotations. (*a*) θ_y = +30°. (*b*) θ_z = -45°.

center at (0.0,0.0). The object requires ten parts. We use six cylinders of unit length to form the sides and place four spheres at the intersection points for an aesthetically-pleasing joint. Spheres have the same appearance in any orientation so rotations are not necessary. As constructed, a cylinder extends from [0.0,0.0,-0.5] to [0.0,0.0,0.5] in assembly space; therefore, we need both rotations and translations. Figure 5.4 shows the completed ornament with numbered cylinders and spheres. Table 5.1 lists the rotation and shift operations that were applied to the parts. The parameters for the cylinders are fairly involved. To derive them, we used the *Line converter* in the *Help* menu of **MetaMesh**. This useful utility takes starting and ending points for an arbitrary line in three-dimensional space and determines values of displacement and rotation for an equal-length line created in assembly space. The equivalent line is parallel to *z* with midpoint at the origin. We leave it as an exercise for the reader to construct the tree, additional ornaments and lights.

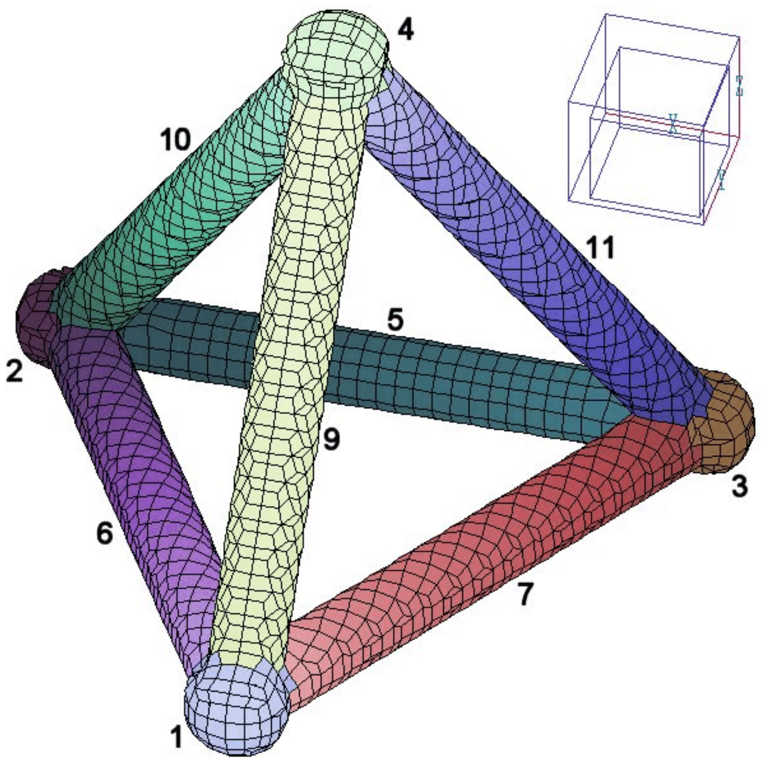


Figure 5.4. Illustration of shift and rotation operations

Table 5.1. Rotation and shift parameters for Fig. 5.4

Part	θx	θy	θz	Sx	Sy	Sz
1 (Sphere)	0.0	0.0	0.0	0.000	0.577	0.000
2 (Sphere)	0.0	0.0	0.0	0.500	-0.289	0.000
3 (Sphere)	0.0	0.0	0.0	-0.500	-0.289	0.000
4 (Sphere)	0.0	0.0	0.0	0.000	0.000	0.816
5 (Cyl)	0.0	90.0	0.0	0.000	-0.289	0.000
6 (Cyl)	0.0	90.0	120.0	0.250	0.145	0.000
7 (Cyl)	0.0	90.0	60.0	-0.250	0.145	0.000
8 (Cyl)	35.26	0.00	0.00	0.000	0.289	0.408
9 (Cyl)	35.26	0.00	-120.0	0.250	-0.145	0.408
10 (Cyl)	35.26	0.00	120.0	-0.250	-0.145	0.408

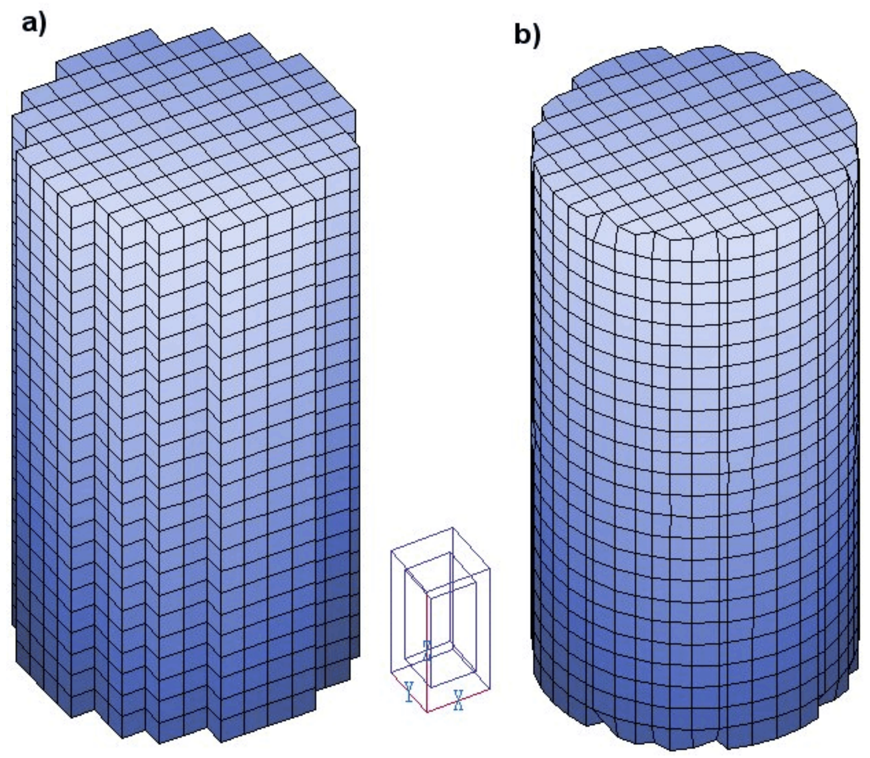


Figure 5.5. Representations of a cylindrical part, (r = 3.0, h = 12.0, volume = 339.2, surface area = 282.7). (a) State after volume fitting (volume = 336.0, surface area = 344.0). (b) State after volume and surface fitting (volume = 337.5, surface area = 280.8).

5.4 Volume fitting procedure

Part types in **MetaMesh** can be divided into two classes: *filled* and *open*. Filled parts have a non-zero volume and include at least one element. Processing of a filled part involves assigning region numbers to both elements and nodes. Open parts have zero volume. Processing an open part involves setting region numbers only for nodes. Open parts are used to represent points, lines and sheets. For example, in an electrostatic solution we could construct a grid of filamentary wires by combining several lines. **MetaMesh** uses different methods to handle open and filled parts. This section and the next two cover techniques for filled parts.

MetaMesh processes filled parts in incremental stages. The first stage is *volume fitting*. The function of volume fitting is to provide a good estimate of the division of the available space to represent the set of parts in the script. The program makes an initial division of the foundation mesh, associating elements with the listed parts. It analyzes the parts in sequence according to their values of *PartNo*. For each part, the program searches all elements of the foundation mesh over the range of volume the part could occupy. If the center-of-mass of an element lies within the part boundaries, the values of *PartNo* and *RegNo* associated with the part are assigned to the element and its eight connected nodes. The volume-fitting process generates a *regular* mesh (brick elements). Figure 5.5*a* illustrates the appearance of a regular mesh for the following part:

```
Part 3
 Type Cylinder
  Region 3
  Fab 3.0 12.0
  Shift 0.0 0.0 0.0
*  Surface Region 1 1.00
*  Surface Region 2 1.00
End
```

Note that the two *Surface* commands were commented out; therefore, the program stopped after volume fitting. At this stage the element surfaces exhibit the stair-step appearance of a regular mesh.

5.5 Fitting surfaces and edges

MetaMesh applies surface and edge fitting operations to create conformal meshes. After the procedures the facets of elements closely follow the surfaces of regions in the solution volume. Conformal meshes greatly improve the accuracy of field interpolations near material surfaces. The capability is important in applications such as electron field-emission and calculation of surface charge density to derive capacitance.

Surface fitting occurs after volume fitting of all parts. The surface of a specific part will be fitted if its part section contains one or more statements of the forms:

```
Surface Region RegNo
Surface Part PartNo
```

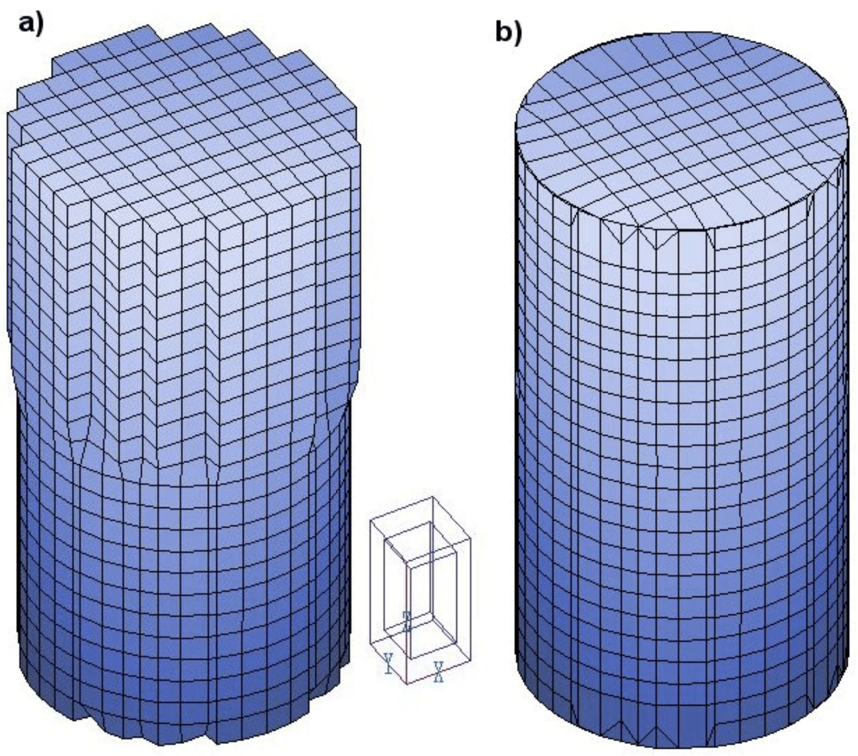


Figure 5.6. Fitting operations on a cylinder with dimensions *r* = 3.0 and *z* = 12.0. (*a*) The bottom half of the cylinder is surrounded by elements with *RegNo* = 1 and the top half by elements with *RegNo* = 2. Only the surface bounding *RegNo* = 1 has been fitted. (*b*) Appearance of the cylinder with both surface and edge fitting.

In response, **MetaMesh** collects all nodes connected to facets on the border between elements of the part and elements that have the region number *RegNo* or part number *PartNo*. For each node, the program finds the closest position, $\mathbf{x}_s$, on the ideal part surface. The closest point depends on the part's geometric type. For example, spheres are easy to fit because they have a single surface, while a box has six surfaces. If the vector **x** represents the initial position of the node, then the new position **x**'is given by

$$\mathbf{x}' = \mathbf{x} + \epsilon_s (\mathbf{x}_s - \mathbf{x}).$$

The quantity ϵ_s is the surface-fitting tolerance — the default value is ϵ_s = 0.90. A value of ϵ_s = 1.00 corresponds to a perfect fit. You can set individual values for the surface fitting tolerance with the following command form:

```
Surface Region RegNo EpsiS
```

As an example, suppose the statement

```
   Surface Region 5 1.00
```

appears in the definition of part number 8. In this case, **MetaMesh** uses ϵ_s = 1.00 to fit nodes connected to facets on the border between elements with *PartNo* = 8 and *RegNo* = 5. The movement of nodes to the surfaces is performed in multiple steps with local relaxation of neighboring nodes that have not already been fitted to a surface. Smoothing operations are necessary to prevent distortion or inversion of elements.

Figure 5.5*b* shows the effect of surface fitting. In the demonstration example, the bottom half of the cylinder is surrounded by elements with *RegNo* = 1 and the top half by elements with *RegNo* = 2. Un-commenting the two *Surface* commands gives fitting over the entire part. If we include only the statement

```
   Surface Region 1 1.00
```

then **MetaMesh** produces the mesh shown in Fig. 5.6*a*. Only the boundary nodes in the bottom half of the solution space have been displaced. The ability to isolate portions of part boundaries that abut specific regions is a powerful feature in **MetaMesh**. We shall discuss some of the implications of this capability when we discuss coatings in next section.

Because surface fitting moves nodes to the closest surface on the part, the procedure may leave bevels on parts with sharp edges (Figs. 5.5*a* and 5.6*b*). This effect is not of concern in electromagnetic field solutions because field values are undefined at surface discontinuities. With regard to the quality of the physical solution, there nothing to be gained by adding sharp corners to objects in the mesh. On the other hand, it is psychologically comforting to make a mesh that looks exactly like the physical system when preparing PowerPoint presentations. Therefore, we have including an edge-fitting capability in **MetaMesh**.

Edge fitting is invoked when you include the string *Edge* after the region number in the *Surface* command. For example, we created the mesh of Fig. 5.6b by changing the two fitting statements in the example to

```
   Surface Region 1 Edge 1.00
   Surface Region 2 Edge 1.00
```

Table 5.2 shows a comparison of volumes and surface areas for the cylinder at the three levels of fitting. Note that the volume computed with both surface and edge fitting is about 0.32% lower than the theoretical value. This small discrepancy results from the approximation of the circular surface of the cylinder with a finite number of facets. The surface area differs from the theoretical value by only 0.07%.

Table 5.2. Surface and edge fitting accuracy (Cylinder with radius 3.0 and height 12.0)

	Theoretical	**Volume fit**	**Surface**	**Surface/edge**
Volume	339.2	336.0	337.5	338.1
Surface area	282.7	334.0	280.0	282.5

With careful pre-analysis of the simulation geometry and good choices of element sizes in the foundation mesh, **MetaMesh** runs automatically and effortlessly. Nonetheless you should be aware that it is possible to create bad meshes. This is a consequence of the program’s versatility and transparency to the user. Most problems can be avoiding by following two guidelines:

- Analyze your solution geometry and simplify it if possible. Eliminate small details that would have a negligible effect on the field solution. Avoid simulating a large assembly if only a small region is of interest. The solutions of the **AMaze** programs will be much more effective if you carefully analyze the nature and goals of your calculation,

- Avoid the temptation to fit every surface in the solution volume to the highest level. Consider whether a precisely-shaped surface is necessary for the accuracy of the solution or whether a sharp edge will enhance or degrade the solution accuracy.

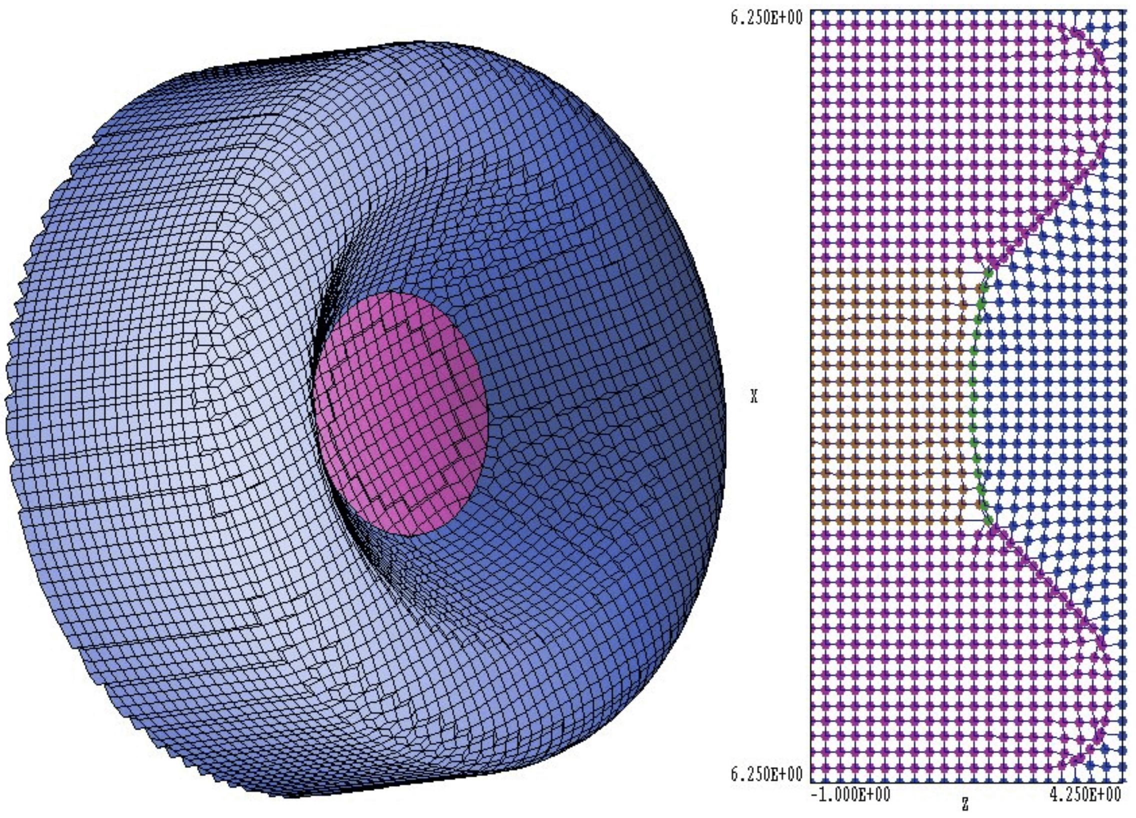


Figure 5.7. Example of coating operation — high-current cathode assembly. Three-dimensional view and slice through the mid-plane showing node identities. Note the coated region of nodes marked green.

5.6 Coating part surfaces

The coating operation paints nodes on selected surfaces of parts. We have already seen one use of coatings to correct node region numbers when drilling a hole through a fixed-potential region in an electrostatic solution (Section 2.4). The *Coat* command has the format

```
Coat RegNo RegNew
```

where *RegNo* and *RegNew* are integers. **MetaMesh** takes the following actions in response to the command. The program collects all nodes connected to facets that lie on the border between the currently-processed part and elements that have region number *RegNo.* The collection procedure is the same one used in surface and edge fitting. The program then changes the region number of the nodes from their current value to *RegNew*.

The following example illustrates an application for coatings. We want to represent a high-current dispenser cathode and focusing electrode for an **OmniTrak** simulation. To mark the electron emission surface, we need to define a set of special nodes that cover a surface in the shape of a spherical section. Table 5.3 shows the script definitions for the focus electrode (*RegNo* = 2) and cathode (*RegNo* = 3) immersed in a vacuum volume with *RegNo* = 1. The focus electrode appears first in the file and involves only surface fitting while the subsequent cathode support section includes both surface and edge fitting. The order ensures that the cathode surface has a circular edge with a precise radius of 2.0. The statement `Coat 1 4` in the cathode section instructs the program to identify nodes on the vacuum surface of the cathode and to change their number to *RegNo* = 4. Figure 5.7 shows the results. The focus electrode is blue and the cathode is violet in the three-dimensional plot on the left-hand side. The plot on the right-hand side is a two-dimensional slice through the midplane with color-coded markers to show node identities. Note the layer of nodes with *RegNo* = 4 (green) on the cathode surface.

Table 5.3. Script sections to define a high-current cathode and focus electrode

```
Part 2
  Type Turning
* Focus electrode
     L    2.00000E+00  2.00000E+00  3.70711E+00  3.70711E+00  S
     A    3.70711E+00  3.70711E+00  4.00000E+00  4.41421E+00  3.00000E+00  4.41421E+00 S
     L    4.00000E+00  4.41421E+00  4.00000E+00  5.00000E+00 S
     A    4.00000E+00  5.00000E+00  3.00000E+00  6.00000E+00  3.00000E+00  5.00000E+00 S
     L    3.00000E+00  6.00000E+00 -1.00000E+00  6.00000E+00 S
     L   -1.00000E+00  6.00000E+00 -1.00000E+00  2.00000E+00
     L   -1.00000E+00  2.00000E+00  2.00000E+00  2.00000E+00
  End
  Region 2
  Surface Region 1 1.00
End

Part 3
  Type Turning
* Cathode support
     L   -1.00000E+00  0.00000E+00  1.71989E+00  0.00000E+00
     A    1.71989E+00  1.77168E-07  2.00000E+00  2.00000E+00  9.00000E+00  0.00000E+00 SE
     L    2.00000E+00  2.00000E+00 -1.00000E+00  2.00000E+00 SE
     L   -1.00000E+00  2.00000E+00 -1.00000E+00  0.00000E+00 SE
  End
  Region 3
  Surface Region 1 Edge 1.00
  Surface Region 2 Edge 1.00
  Coat 1 4
End
```

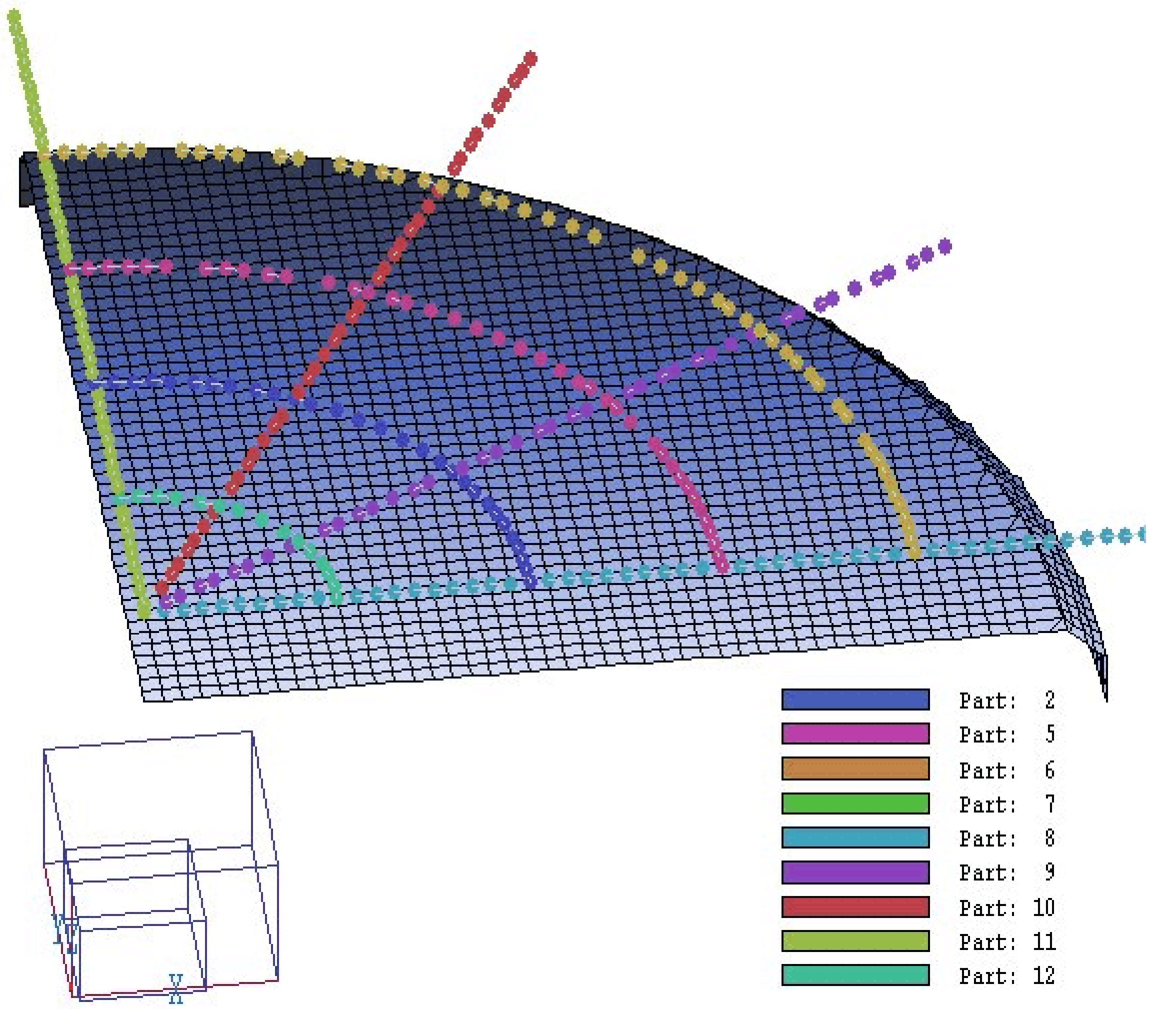


Figure 5.8. Cathode control grid created with the open part types *Line* and *Circle*.

5.7 Using open parts for node fitting

In the previous section we saw that the coating operation can be used to modify node identities on selected surfaces of filled parts. This section discusses an alternate approach to node editing. **MetaMesh** supports a set of *open* part types shown in Table 5.4. The *Point* type sets the position and identity of a single node. The next group (*BoundXUp*, *BoundXdn*,...) sets all nodes on a specified boundary of the solution volume. These types are useful to define ground planes or to implement symmetry conditions. The *Line*, *Arc* and *Circle* types create one-dimensional objects consisting of a chain of nodes that follows a desired shape. The final group sets nodes over two-dimensional surfaces. A *Disk* is a flat surface with a round edge, while a *Plate* has a rectangular edge. The *Bubble* is a spherical surface. Chapter 8 describes open part types in detail. Here we shall summarize the general rules for processing.

Table 5.4. Open part types

Class	Type	Fab01	Fab02	Fab03
Zero-dimensional	*Point*			
Boundaries	*BoundXUp*			
	BoundXDn			
	BoundYUp			
	BoundYDn			
	BoundZUp			
	BoundZDn			
One-dimensional	*Line*	L		
	Arc	R	θ	
	Circle	R		
	Rectangle	W_x	W_y	
Two-dimensional	*Disk*	R		
	Plate	W_x	W_y	
Three-dimensional	*Bubble*	R		

The fitting of open parts begins after all filled parts have been processed. The identities and positions of open part nodes overwrite previous definitions for filled parts. Furthermore, open parts with higher values of *PartNo* overwrite previously-processed open parts. The procedure used in **MetaMesh** to set a *Point* is simple. The code searches for the node closest to the desired location. It then shifts the node position and reassigns the values of *PartNo* and *RegNo.* The processing of boundary commands is also straightforward. For example, in response to the *BoundYUp* command, **MetaMesh** reassigns values of *PartNo* and *RegNo* for all nodes with index $J = J_{max}$. Because the sides of the solution volume are fixed, there is no need for position shifts.

The processing of other part types involves complex operations. The code checks all nodes in the vicinity of the part, comparing the minimum distance between the node and the part to the distance to neighboring nodes projected along the direction of the shift. If the ratio of the distances is relatively small, **MetaMesh** moves the node and reassigns values of *PartNo* and *MeshNo*. Shifting operations are performed in incremental stages with intermediate local smoothing to avoid mesh distortion. For the two-dimensional *Disk* and *Plate* types, the code first sets the edges and then fills in the enclosed area. The procedure must handle variations in mesh resolution and arbitrary rotations of the part. In the end, all shifted nodes will lie exactly on the entity but may not be evenly spaced, particularly if the part crosses planes of the foundation mesh. Figure 5.8 shows an example, a control grid above the surface of a beveled cathode (*RegNo* = 2). A focusing electrode is not shown. The grid was created with the commands listed in Table 5.5.

5.8 Summary of part section commands

This section documents the commands that may appear in *Part* sections.

Symbolic command: `Type PartType`
Example: `Type Cylinder`
Function: Specify the geometric class of the part
Comments: The string parameter gives the part type, one of the choices listed Chaps. 6, 7 and 8. The special form of the *Type* command for extrusions and turnings is described in Chap. 7.

Symbolic command: `Region RegNo`
Example: `Region 4`
Function: Associate the part with a region of the solution volume.
Comments: The quantity *RegNo* is an integer from 1 to 127 or a symbolic name defined by the *REGNAME* command. Several parts may be used to construct a single region.

Symbolic command: `Name NameString`
Example: `Name EpoxyPot`
Function: Set up a symbolic reference to the assigned part number.
Comments: The string may have length from 1 to 20 characters. Standard delimiters may not be included. The underscore symbol is allowed.

Table 5.5. Commands to create the grid of Fig. 5.8

```
Part 5
  Type Line
  Region 3
  Fab 2.6
  Rotate 0.00 90.00 0.00 XYZ
  Shift 1.3001 0.00 0.30
End

 Part 6
  Type Line
  Region 3
  Fab 2.6
  Rotate 0.00 90.00 30.00 XYZ
  Shift 1.1259 0.6501 0.30
 End

 Part 7
  Type Line
  Region 3
  Fab 2.6
  Rotate 0.00 90.00 60.00 XYZ
  Shift 0.6501 1.1259 0.30
 End

 Part 8
  Type Line
  Region 3
  Fab 2.6
  Rotate 0.00 90.00 90.00 XYZ
  Shift 0.0000 1.300 0.30
 End
```

```
Part 9
 Type Circle
 Region 3
 Fab 0.5
 Shift 0.00 0.00 0.30
End

Part 10
 Type Circle
 Region 3
 Fab 1.0
 Shift 0.00 0.00 0.30
End

Part 11
 Type Circle
 Region 3
 Fab 1.5
 Shift 0.00 0.00 0.30
End

Part 12
 Type Circle
 Region 3
 Fab 2.0
 Shift 0.00 0.00 0.30
End
```

Symbolic command: `FAB Param1 Param2 ...`
Example: `FAB  6.00 8.00`
Function: Dimensions of the part fabricated in the assembly space.
Comments: The quantities *Param1*, *Param2*, ... are real numbers that give the dimensions of the part. The number of parameters required will depend on the part type. For example, a sphere requires one parameter while a box has three dimensions.

Symbolic command: `ROTATE θX θY θZ [ROrder]`

Examples:
```
ROTATE 0.0 90.0 45.0
ROTATE 0.0 90.0 45.0 ZY
```
Function: Orient the part in assembly space before shifting to the solution space..

Comments: The three real number parameters are the angles (in degrees) for rotations θ_x, θ_y and θ_z about the *x*, *y* and *z* axes respectively. By default, the rotations are performed in the order θ_x:θ_y:θ_z. If you want to change the order, enter a string as Parameter 4. The string may contain from 1 to 3 characters (*X*, Y and/or *Z*) giving the order of rotations. For illustration, the second example above calls for rotation by 45° about the *z* axis and then by 90° about the ***new*** *y* axis. The directions of positive rotation angles follow the right hand rule.

Symbolic command: `SHIFT XS YS ZS`

Example: `SHIFT 5 0.00 0.00 3.56`

Function: Position the part in the solution space.

Comments: Shifts are performed after rotations. The three real number parameters are displacements along the *x*, *y* and *z* axes respectively.

Symbolic command: `SURFACE REGION RegNo [Edge] [EpsiS]`

Examples:
```
SURFACE REGION 5 1.00
SURFACE REGION 7 Edge 0.80
SURFACE REGION AirVolume
```
Function: Fit specified surfaces of the part to ideal values to construct a conformal mesh.

Comments: A part may abut elements of several different regions. The command instructs the program to shift nodes along facets that border on a specific region. The integer parameter is the region number (1-127). You may also use a string synonym defined with by `REGNAME` command. The optional string *Edge* initiates edge fitting for the surface. The optional real-number parameter *EpsiS* is the fitting tolerance. A value of 0.0 gives no fitting, a value of 1.00 gives perfect fitting. The default is *EpsiS* = 0.90. Several *Surface* commands may appear in a *Part* section. The command has no effect if the part does not share a boundary with the specified region.

Symbolic command: `SURFACE PART PartNo [Edge] [EpsiS]`

Examples:
```
SURFACE PART 6 0.85
SURFACE PART 7 Edge 0.95
SURFACE PART EndPlate
```

Function: Fit specified surfaces of the part to ideal values to construct a conformal mesh.

Comments: A part may abut elements of several different parts. The command instructs the program to shift nodes along facets that border on a specific part. The integer parameter is the part number. You can also use a string synonym defined with a `NAME` command in a different `PART` section. (**MetaMesh** reads all part names before processing, so synonyms are valid even if the part appears later in the script.) The optional string *Edge* initiates edge fitting for the surface. The optional real-number parameter *EpsiS* is the fitting tolerance. A value of 0.0 gives no fitting, a value of 1.00 gives perfect fitting. The default is *EpsiS* = 0.90. Several *Surface* commands may appear in a *Part* section. The command has no effect if the part does not share a boundary with the specified part.

Symbolic command: `COAT RegNo RegNew`

Examples:
```
COAT 2 6
COAT  AirVolume  LowerElectrode
```

Function: Change the region number of nodes connected to facets that border on region number *RegNo* to *RegNew*.

Comments: The parameters *RegNo* and *RegNew* are integers in the range 1-127. You can also use symbolic strings defined by the *REGNAME* command. The *COAT* command has no effect if the part does not share a boundary with elements of region number *RegNo*.

5.9 Script directives and block operations

Script directives are special commands that can be included between *PART* sections to modify the operation of **MetaMesh**. Their main use is to shift or rotate a group of parts as a block. With this feature, you can build libraries of standard assemblies and place them at different positions within a mesh.

Script directives start with the pound sign (#) and must appear ***between*** *PART* sections. Each command must be balanced by a corresponding *#END* type command. You can combine different directives but only one instance of a each directive may be active at time.

The following directives control the position and orientation of a group of parts.

```
#SHIFT  XShift YShift ZShift
#SHIFT 0.25 0.00 0.00
#ENDSHIFT
```

Shift all parts contained between the lines *#SHIFT* and *#ENDSHIFT* by the given displacements along the *x*, *y* and *z* directions. **MetaMesh** adds the global shift to any local displacements defined by the *SHIFT* command in *PART* sections. Global shifts are performed *after* global rotations. The program issues an error message if the *#SHIFT* command does not include three real parameters or have a corresponding *#ENDSHIFT* command..

```
#ROTATE  XRot YRot ZRot [RotOrder]
#ROTATE 22.5 0.00 90.00 ZX
#ENDROTATE
```

Rotate all parts between the lines by the angles *XRot*, *YRot* and *ZRot* (specified in degrees). Rotations are performed about the axes of the assembly coordinate system after any local rotations and shifts. Global translations are performed *after* global rotations. By default, rotations are performed in the order *XRot* ➜ *YRot* ➜ *ZRot*. You can change the order with the optional string parameter *RotOrder*. Enter from one to three characters. For example, *RotOrder* = ‘X’ specifies that there is a rotation only about the x axis, while ‘ZY’directs the program to perform a rotation about *z* and then *y*. **MetaMesh** issues an error message if the *#ROTATE* command does not include three real parameters and (optionally) a string parameter.

To summarize, the script directives *#SHIFT* and *#ROTATE* define a *global* displacement and rotation that is applied to all included parts. The commands *SHIFT* and *ROTATE* within each *PART* section define *local* shifts and displacements. Operations are performed in the following order: local rotation ➜ local displacement ➜ global rotation ➜ global displacement.

```
#SKIP
#ENDSKIP
```

Ignore all parts between the commands. Use this construction to remove parts temporarily from an assembly.

#NOFIT
#ENDNOFIT

Deactivate *SURFACE* commands for all included parts.

The following example shows how to move an assembly consisting of several parts and to remove one of the parts:

```
#SHIFT 1.50 0.0 0.0
Part
  Type Cylinder
  Name CrossPiece
  Region Assembly
  Fab 0.75 5.0
  Surface Region SolnVolume
End
Part
  Type Sphere
  Name UpperSphere
  Region Assembly
  Fab 2.5
  Shift 0.00 0.00 4.50
  Surface Region SolnVolume
End
#SKIP
Part
  Type Sphere
  Region Assembly
  Name LowerSphere
  Fab 2.5
  Shift 0.00 0.00 -4.50
  Surface Region SolnVolume
End
#ENDSKIP
#ENDSHIFT
```

The next example shows the rotation and displacement of an assembly of three spheres:

```
#ROTATE 0.0 0.0 45.0
#SHIFT 0.5 0.5 0.0

PART
  Type Sphere
  Region PVolume
  Name PVolume
  Fab 1.0
  Surface Region Vacuum
END

PART
  Type Sphere
  Region SphereXUp
  Name SphereXUp
  Fab 1.0
  Shift  5.0  0.0  0.0
  Surface Region Vacuum
END

PART
  Type Sphere
  Region SphereXDn
  Name SphereXDn
  Fab 1.0
  Shift -5.0  0.0  0.0
  Surface Region Vacuum
END


#ENDROTATE
#ENDSHIFT
```

Chapter 6. Basic filled parts

There are fives types of filled parts available to construct regions: 1) basic shapes, 2) extrusions, 3) turnings, 4) transitions and 5) neon. Basic shapes are simple solids for which you can choose sizes and aspect ratios. Extrusions are shapes with arbitrary cross-sections that extend over a given height. Turnings are figures of revolution with arbitrary cross-sections. Transitions are objects that smoothly change from one cross-section to another between two planes. Neon objects are tubes that follow an arbitrary path in three-dimensional space. We shall cover basic shapes in this chapter. The next chapter describes methods to define extrusions, turnings and transitions. Chapter 8 describes the neon model. The listed basic shapes are available in the current version of **MetaMesh**. We anticipate adding parts to the library in the future. See the file `LIBRARY.DAT` for an updated list. If you find that you are unable to model your application with the standard parts library, please contact us to propose a new model. We may be able to implement the part and send you an updated program.

In the following list, the **Type** is the string entry that appears in the `TYPE` command of the `PARTS` section. The **Dimensions** are the real-number quantities that appear in the `FAB` command. The **Comments** describe the position and orientation of the part when it is constructed in assembly space.

Type: SPHERE
Dimensions: 1) R (radius)
Comments: The center of the sphere is at (0.0,0.0,0.0); the sphere extends from $-R$ to R along each axis.

Type: BOX
Dimensions: 1) L_x (full length of side along x), 2) L_y (full length of side along y) and 3) L_z (full length of side along z).
Comments: The center-of-mass of the box is at (0.0,0.0,0.0); the box extends from $-L_x/2$ to $L_x/2$ along x, $-L_y/2$ to $L_y/2$ along y, and $-L_z/2$ to $L_z/2$ along z.

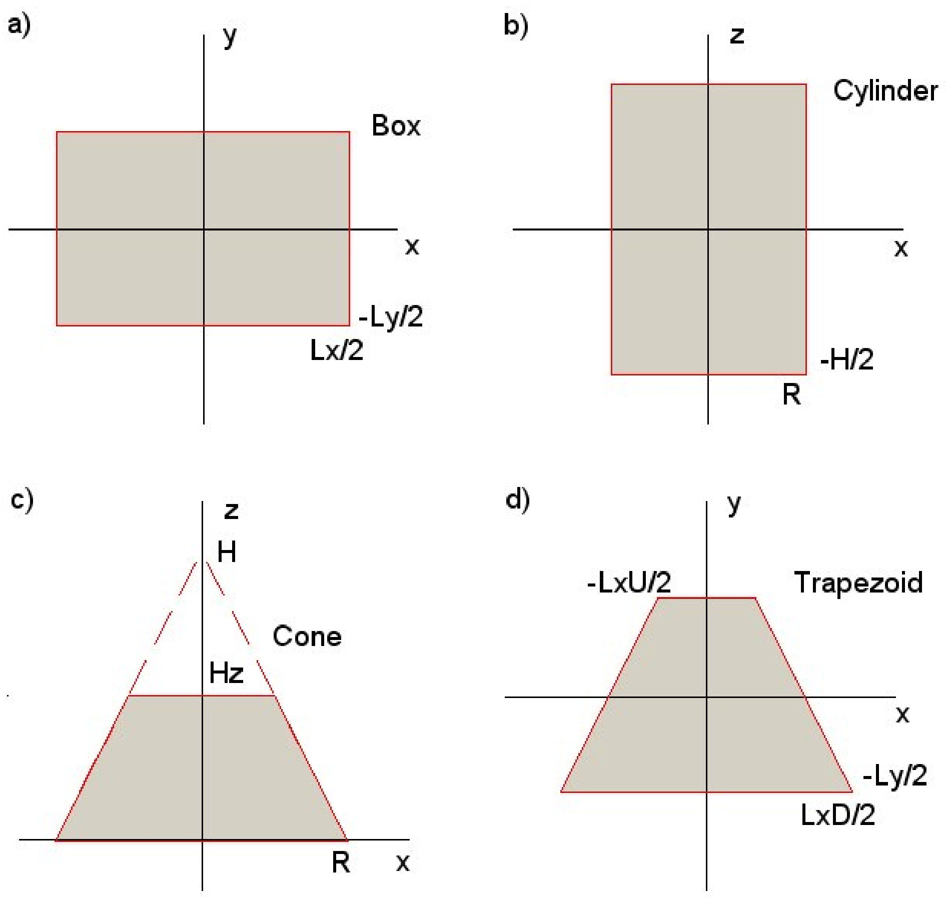


Figure 6.1. Geometric definitions of basic parts

Type: CYLINDER
Dimensions: 1) *R* (radius) and 2) *H* (height)
Comments: The center-of-mass of the cylinder is at (0,0,0) with the height aligned along z. The cylinder has a circular cross-section that extends from -*R* to *R* in *x* and *y* and faces at -*H*/2 and *H*/2 in *z*.

Type: CONE
Dimensions: Truncated cone with 1) *R* (base radius), 2) *H* (height of a the full cone) and 3) H_z (height of the truncated cone)
Comments: The cone axis is aligned along *z* with the base in the *x*-*y* plane at *z* = 0. The circular base of the cone extends from -*R* to *R* in *x* and *y*. The dimension *H* defines the cone apex and H_z defines the height along the *z* axis of the truncated figure.

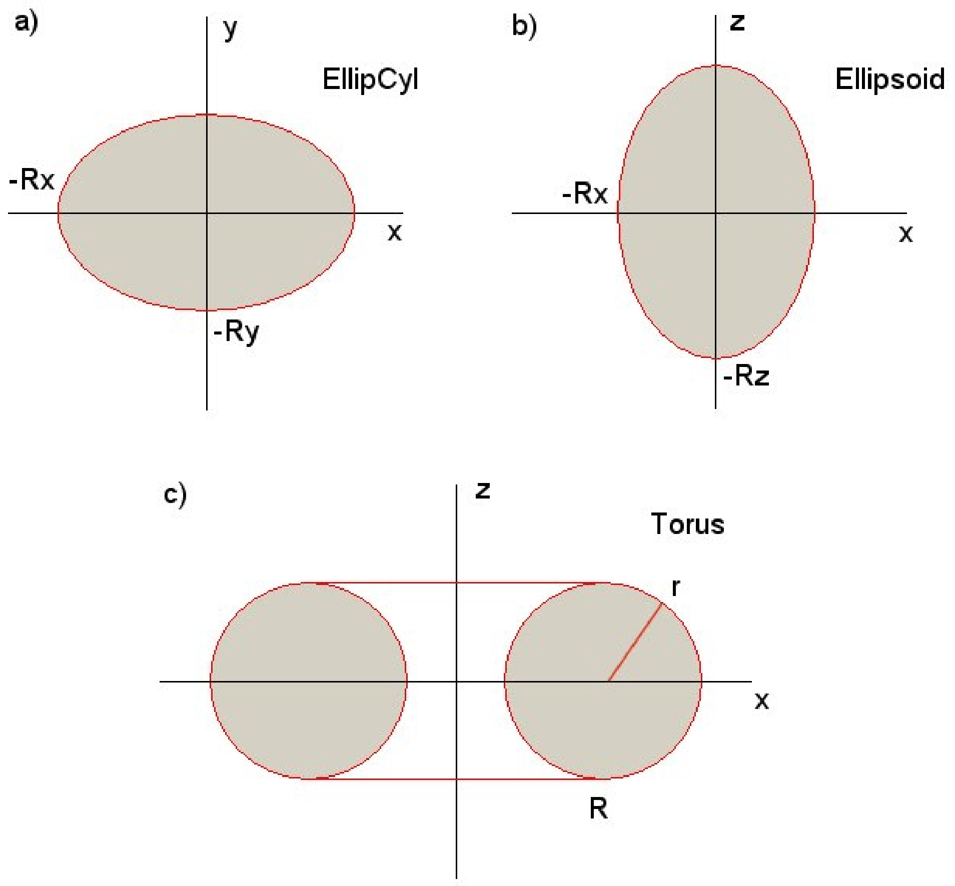


Figure 6.2. Geometric definitions of basic parts

Type: ELLIPCYL
Dimensions: 1) R_x, 2) R_y and 3) H
Comments: A cylinder with an elliptical cross-section in the *x-y* plane. The center-of-mass of the elliptical cylinder is at (0,0,0) with the height aligned along *z*. It has faces at -*H*/2 and *H*/2 in *z*. The boundary in the **x-y** plane is defined by the equation

$$(x/R_x)^2 + (y/R_y)^2 = 1$$

Type: ELLIPSOID
Dimensions: 1) R_x, 2) R_y and 3) R_z
Comments: The surface of the ellipsoid is defined by the equation

$$(x/R_x)^2 + (y/R_y)^2 + (z/R_z)^2 = 1.$$

Type: TORUS
Dimensions: 1) *R* (major radius) and 2) *r* (minor radius,)
Comments: The torus is centered at (0,0,0). A plane of the hole is normal to *z*.

Type: HELIX
Dimensions: 1) *R* (radius of the bounding cylinder), 2) *r* (radius of the winding), 3) *H* (total height of the bounding surface), 4) H_w (height of one revolution).
Comments: The helix winding has a circular cross-section with a radius r. The centerline of the winding lies on a bounding cylinder of radius *R* and height *H* aligned with the *z* axis. The center of mass of the helix lies at (0.0,0.0,0.0). The projection of the helix winding in the *x-y* plane extends from a radius of $R - r$ to $R + r$. The shape extends from $-H/2 - r$ to $H/2 + r$ in *z*.

Type: TRAPEZOID
Dimensions: 1) *LxU* (full width in *x*, top side), 2) *LxD* (full width in *x*, bottom side, 3) L_y (full width in *y*) and 4) L_z (full width in *z*)
Comments: The shape is an extrusion along z with a trapezoidal cross section. It extends from -Ly/2 to Ly/2 in y and -Lz/2 to Lz/2 in z. The full width in x is LxD at -Ly/2 and LxU at Ly/2.

Chapter 7. Extrusions, turnings and transitions

Extrusions, turnings and transitions greatly enhance the ability of **MetaMesh** to model complex shapes. An extrusion of height *H* has an arbitrary cross-section in the *x-y* plane of assembly space that extends uniformly along the *z* axis over the range $-H/2 \le z \le H/2$. The outline in the *x-y* plane is defined by a modified form of the *TYPE* statement:

```
TYPE EXTRUSION
  Vector 1
  Vector 2
  Vector 3
    ...
  Vector N
END
```

The vectors comprise a set of up to 50 arcs and/or lines that define a closed outline in the *x-y* plane. A line vector has the format:

```
L  XStart  YStart  XEnd  YEnd
```

For example, the vector

```
L  0.0  0.0   1.0  2.0
```

extends from the point (0.0,0.0) to (1.0,2.0). An arc vector has the format

```
A  XStart  YStart  XEnd  YEnd  XCenter  YCenter
```

An arc must must not cover more than 90°. Break larger arcs into two or more parts. For example, the following data set defines a half-circular cross-section with radius 1.0:

```
L  -0.5  0.0  0.5  0.0
A   0.5  0.0  0.0  0.5  0.0  0.0
A   0.0  0.5 -0.5  0.0  0.0  0.0
```

The `FAB` command for an extrusion has a single parameter, the height *H* along *z* of the reference space. As an example, the following commands define a keyhole-shaped object with a height *H* = 4.0. The additional characters `SE` in the data lines control surface and edge fitting. Figure 7.1 shows the resulting part.

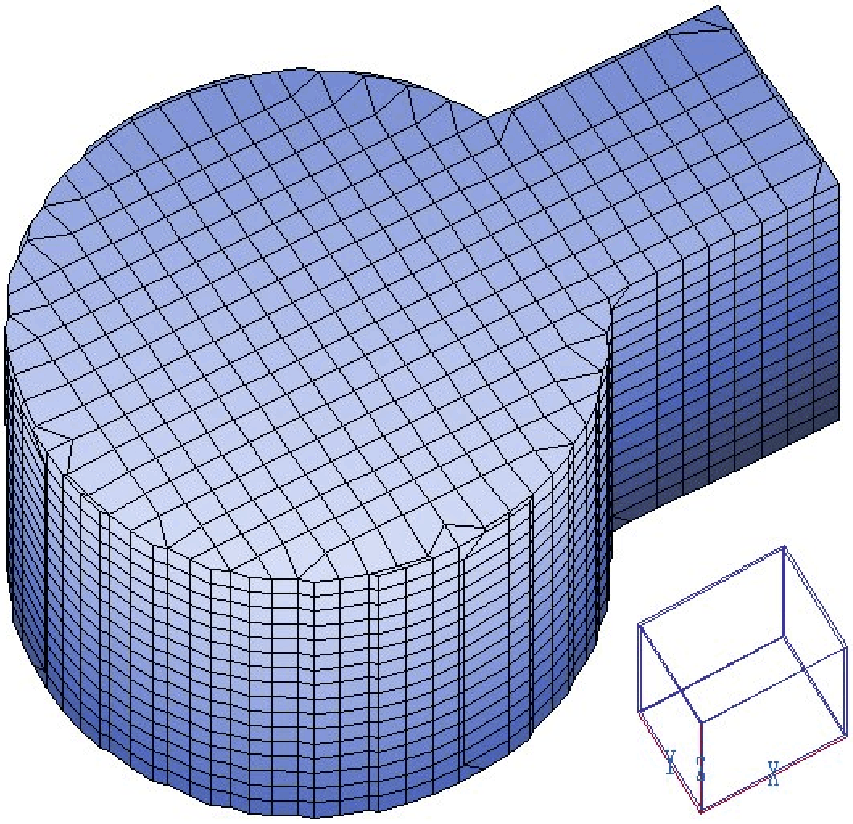


Figure 7.1. Example of an extrusion with surface and edge fitting

```
Part 2
  Type Extrusion
     L    5.29129E+00 -1.00000E+00  7.50000E+00 -1.00000E+00 SE
     L    7.50000E+00 -1.00000E+00  7.50000E+00  1.00000E+00 SE
     L    7.50000E+00  1.00000E+00  5.29129E+00  1.00000E+00 SE
     A    5.29129E+00  1.00000E+00  3.00000E+00  2.50000E+00  3.0E+00  0.0E+00 S
     A    3.00000E+00  2.50000E+00  5.00000E-01  6.08398E-08  3.0E+00  0.0E+00 S
     A    5.00000E-01  6.08398E-08  3.00000E+00 -2.50000E+00  3.0E+00  0.0E+00 S
     A    3.00000E+00 -2.50000E+00  5.29129E+00 -1.00000E+00  3.0E+00  0.0E+00 SE
  End
  Region 2
  Fab 4.0
  Shift 0.0 0.0 0.0
  Surface Region 1 Edge
End
```

A turning is the type of part you would make on a lathe. It is a figure of revolution about the *z*-axis of assembly space. It has an arbitrary cross section in the *r*-*z* plane of assembly space defined by line and arc vectors following the same convention as the extrusion. The `TYPE` command has the format.

```
TYPE TURNING
     Vector 1
     Vector 2
     Vector 3
       ...
     Vector N
  END
```

In this case the vector lines have components:

```
  L  ZStart  RStart  ZEnd  REnd
```

or

```
  A  ZStart  RStart  ZEnd  REnd  ZCenter  RCenter
```

The `FAB` command for a turning has two parameters:

```
  FAB AngMin AngMax
```

The quantities *AngMin* and *AngMax* give the minimum and minimum azimuthal extent (in degrees). A value of 0.0° corresponds to the *x* axis of assembly space. The default values are *AngMin* = 0.0 and *AngMax* = 360.0 (a full turning).

A *transition* is a powerful model that makes a smooth three-dimensional connection between arbitrary shapes in two parallel planes. The transition requires some work to set up but yields dramatic results. Figure 7.2 shows an example where a circle changes to a elongated rectangle. The shapes in the two planes are defined by a series of up to 50 connected line vectors. A transition section has the following syntax:

```
TYPE Transition
   XsDn1 YsDn1 XeDn1 YeDn1   XsUp1 YsUp1 XeUp1 YeUp1
   XsDn2 YsDn2 XeDn2 YeDn2   XsUp2 YsUp2 XeUp1 YeUp2
   XsDn3 YsDn3 XeDn3 YeDn3   XsUp3 YsUp3 XeUp1 YeUp3
   ...
   XsDnN YsDnN XeDnN YeDnN   XsUpN YsUpN XeUpN YeUpN
END
```

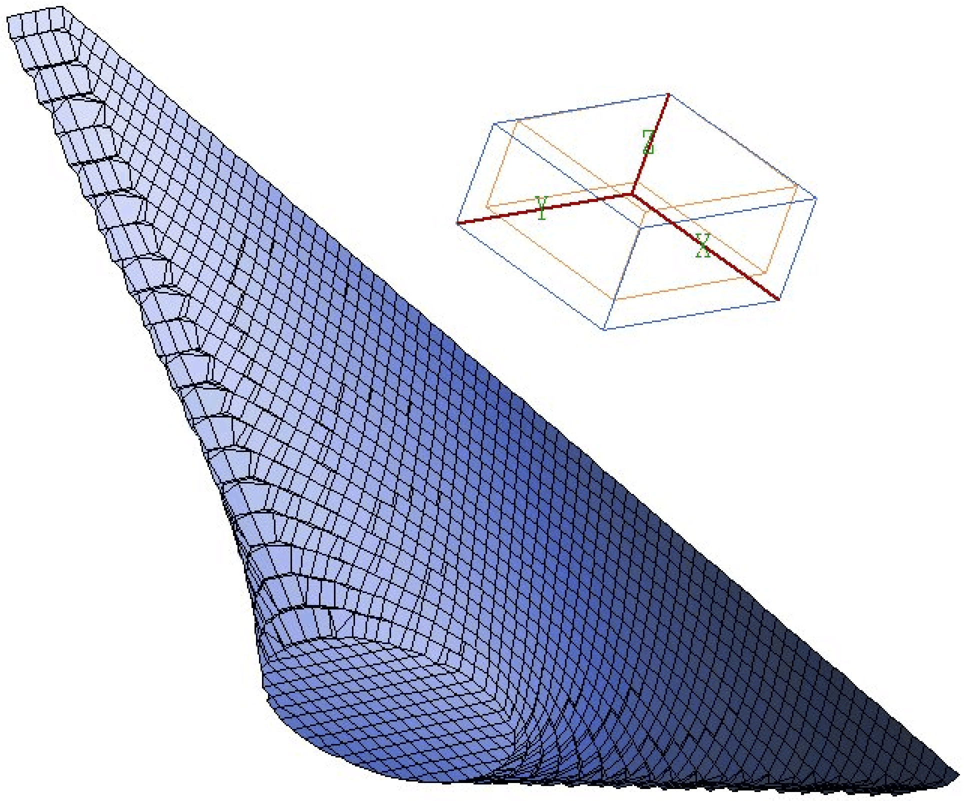


Figure 7.2. Use of the transition model, circle to elongated rectangle.

Note that each data line contains the start and end points of two line vectors. The first vector lies in the lower plane (normal to *z* axis of assembly space) and the second vector is the mapping to the upper plane. The *FAB* command gives the distance between the two planes:

```
FAB DPlane
```

The planes are located at positions *z* = -*DPlane*/2 and z = +*DPlane*/2. Both sets of vectors must define closed shapes in their respective planes. The syntax of the transition data lines ensures that 1) there are the same number of vectors in both planes, and 2) both sets of vectors have the same logical relationship to each other.

As an example, the following transition section defines the shape of Fig. 7.2:

```
Region Compression
Type Transition
   1.7658   0.2539   1.6228   0.7410   10.00   0.50   8.00   0.50  S
   1.6228   0.7410   1.3483   1.1682    8.00   0.50   6.00   0.50  S
   1.3483   1.1682   0.9647   1.5007    6.00   0.50   4.00   0.50  S
   0.9647   1.5007   0.5029   1.7117    4.00   0.50   2.00   0.50  S
   0.5029   1.7117   0.0004   1.7840    2.00   0.50   0.00   0.50  S
   0.0004   1.7840  -0.5022   1.7119    0.00   0.50  -2.00   0.50  S
  -0.5022   1.7119  -0.9641   1.5011   -2.00   0.50  -4.00   0.50  S
  -0.9641   1.5011  -1.3479   1.1687   -4.00   0.50  -6.00   0.50  S
  -1.3479   1.1687  -1.6225   0.7417   -6.00   0.50  -8.00   0.50  S
  -1.6225   0.7417  -1.7657   0.2546   -8.00   0.50 -10.00   0.50  S
  -1.7657   0.2546  -1.7660  -0.2531  -10.00   0.50 -10.00  -0.50  S
  -1.7660  -0.2531  -1.6231  -0.7403  -10.00  -0.50  -8.00  -0.50  S
  -1.6231  -0.7403  -1.3489  -1.1676   -8.00  -0.50  -6.00  -0.50  S
  -1.3489  -1.1676  -0.9653  -1.5003   -6.00  -0.50  -4.00  -0.50  S
  -0.9653  -1.5003  -0.5036  -1.7114   -4.00  -0.50  -2.00  -0.50  S
  -0.5036  -1.7114  -0.0011  -1.7840   -2.00  -0.50   0.00  -0.50  S
  -0.0011  -1.7840   0.5015  -1.7121    0.00  -0.50   2.00  -0.50  S
   0.5015  -1.7121   0.9634  -1.5015    2.00  -0.50   4.00  -0.50  S
   0.9634  -1.5015   1.3474  -1.1693    4.00  -0.50   6.00  -0.50  S
   1.3474  -1.1693   1.6222  -0.7424    6.00  -0.50   8.00  -0.50  S
   1.6222  -0.7424   1.7656  -0.2553    8.00  -0.50  10.00  -0.50  S
   1.7656  -0.2553   1.7658   0.2539   10.00  -0.50  10.00   0.50  S
End
Fab 5.0
```

Here, the circle is approximated by 22 line segments. A spreadsheet template was used to generated the coordinates. The additional string parameter *S* indicates that surface fitting will occur on the associated facets.

The `ROTATE` and `SHIFT` commands have the same effect on extrusions, turnings and transitions as they do on basic parts.

MetaMesh can perform both surface and edge fitting on extrusions, turnings and transitions. Fitting occurs when the *Part* section contains commands of the form:

```
SURFACE REGION NReg [Edge] [EpsiS]
SURFACE PART NPart [Edge] [EpsiS]
```

The main difference from basic parts is that you can control fitting on individual surfaces by adding entries to the outline vector list. Surface fitting on a particular facets occurs if you place the character `S` at the end of the vector line. Both surface and edge fitting applies if you place the string `SE` at the end of the line. The following are important points to note about fitting extrusions, turnings and transitions:

- Fitting is not performed on cross-section surfaces if the strings `S` or `SE` do not appear at the end of vector lines, even if the *Part* section contains a `SURFACE` command.

- The end faces of extrusions are fitted when there is a `SURFACE` command in the *Part* section, independent of the `S` or `SE` markings on the cross-section vectors.

- Only surface fitting may be applied to a transition.

- When `SE` appears in a vector line, the fitted edge is the intersection between the facet defined by that vector and the following one in the list. Both facets must have surface fitting. If `SE` appears in the final vector, fitting is applied between the facet for that vector and the facet corresponding to the first vector in the list.

Regarding the last point,. the keyhole extrusion listed earlier in this chapter is a good example of selective fitting. Although surface fitting is applied to all facets, edge fitting applies only to the four intersections with flat surfaces. There is no reason to fit the edges between arc segments.

By default **MetaMesh** assumes that *EXTRUSION*s, TURNINGs and TRANSITIONS exist in free space as in Fig. 7.3. To achieve the best fit the code attempts to move nodes to nearby surfaces on the ends as well as the sides. In many applications we construct objects by butting *EXTRUSION*s, TURNINGs and/or *TRANSITION*s together to form a continuous structure where the ends are not exposed. In this case we can assume that all identified surface facets are on extrusion sides. Therefore, we should move nodes only to the sides and disregard the ends. You can signal this intention to **MetaMesh** by adding the keyword *SIDEFIT* to the definitions of *EXTRUSION*s, TURNINGs and *TRANSITION*s:

```
Type Extrusion SideFit
```

Important note: mesh distortions may occur if the *SIDEFIT* option is used when end facets are exposed.

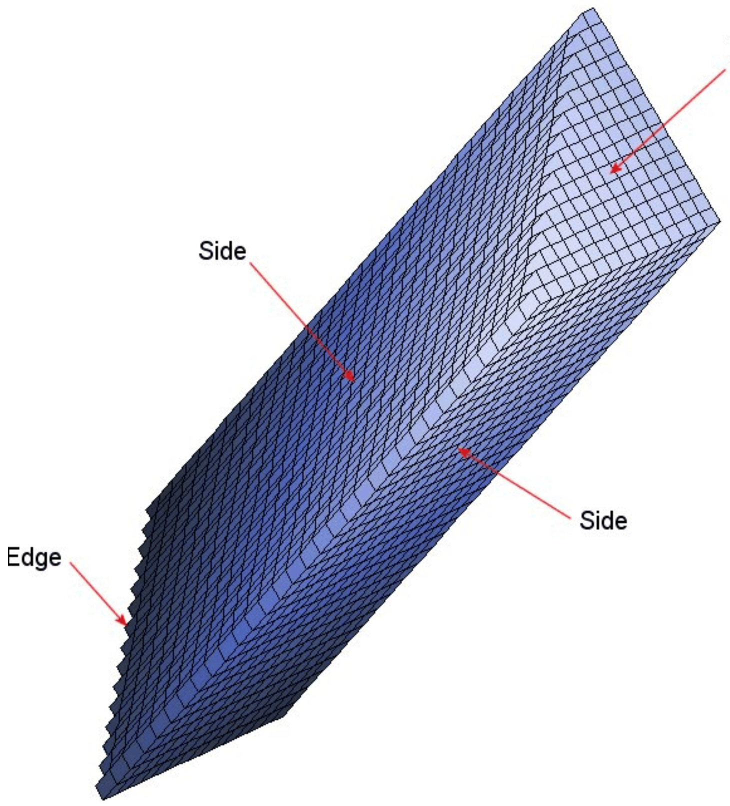


Figure 7.3. Parts of an extrusion. A ragged edge may occur if **MetaMesh** has to apportion fitted nodes to both the sides and the ends.

The final topic in this chapter is when to use extrusions and turnings rather than basic parts. In some cases, either approach is possible. For example, we could create a torus by using the *Torus* part or by using a 360° turning with a circular cross section. Similarly, we could create a *Box* or *Trapezoid* with an extrusions. Use the following two guidelines:

- ■ Use basic parts if you are modeling a standalone part or a simple combination. **MetaMesh** can process basic parts faster and more reliably.

- ■ If a region has a complex cross-section, it is better to use extrusions and turnings than to stack several basic parts together. Extrusions and turnings usually give better results and offer more control over surfaces and edges.

Note that a maximum number of 40 extrusions and/or turnings and 10 transitions may appear in a **MetaMesh** script.

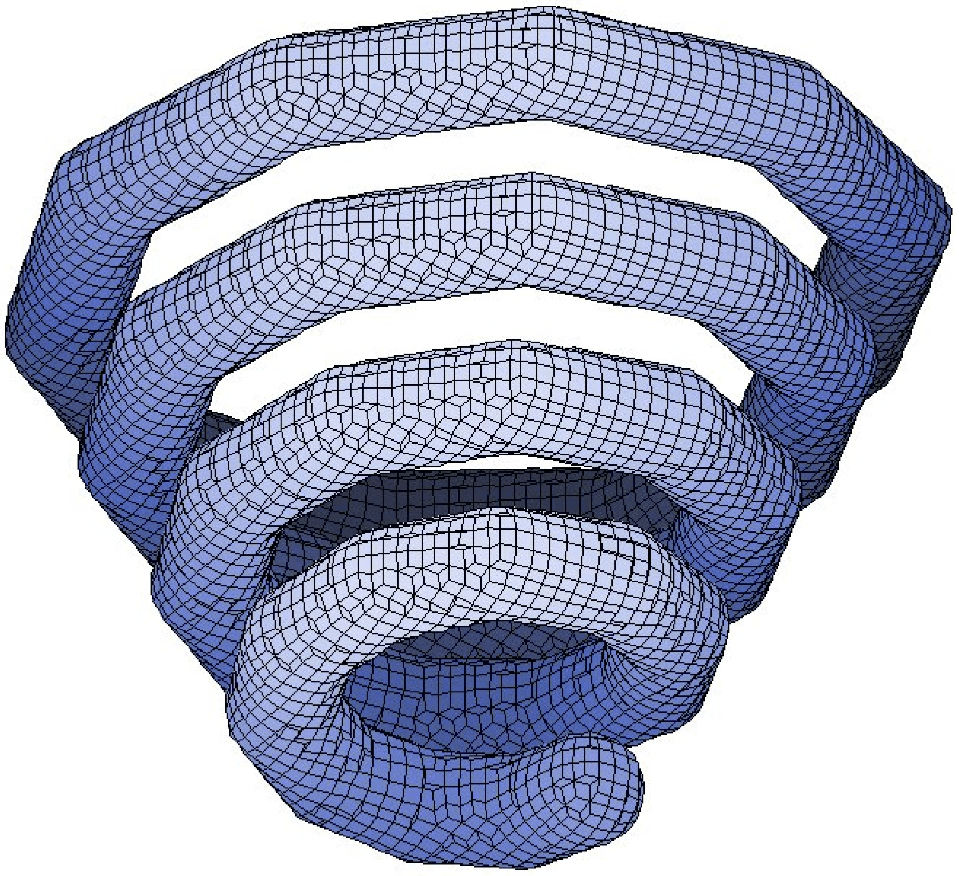


Figure 8.1. Shape created by the file NEONDEMO.MIN

Chapter 8. Neon model for generalized tubing

The neon model creates a tube of given radius that follows an arbitrary path in three-dimensional space. The resulting object looks like the tube in a neon light. The model can be used to create complex wiring, heater filaments, spiral coils and a variety of other shapes. The path of the tube is defined by the following form of the TYPE statement:

```
TYPE NEON
  X0 Y0 Z0
  X1 Y1 Z1
     ...
  XN YN ZN
END
```

The three numbers in each line specify a point in the three-dimensional assembly space. The set of points defines a set of (*N*-1) line segments. Use more points for smoother curves. The points must appear in order. The spacing between the points may be varied. For example, a pipe with a bend could be defined by a long straight section, several closely-spaced points to defined the elbow and another straight section. The maximum number of points in the list is 256. You can define up to 12 neon parts in a script. The radius of the tube is set by the first parameter in the `FAB` command.

As an example, the following section defines the shape of part in Fig. 8.1:

```
Part 2
  Type Neon
     2.0000    0.0000   0.0000
     1.6989    1.2343   0.2000
     0.6798    2.0923   0.4000
    -0.7107    2.1874   0.6000
    -1.9416    1.4107   0.8000
    -2.5000    0.0000   1.0000
    -2.1034   -1.5282   1.2000
    -0.8343   -2.5679   1.4000
     0.8652   -2.6630   1.6000
     2.3461   -1.7046   1.8000
     3.0000   -0.0000   2.0000
     2.5080    1.8221   2.2000
     0.9889    3.0434   2.4000
    -1.0198    3.1385   2.6000
    -2.7507    1.9985   2.8000
    -3.5000   -0.0000   3.0000
    -2.9125   -2.1160   3.2000
    -1.1434   -3.5189   3.4000
     1.1743   -3.6140   3.6000
     3.1552   -2.2924   3.8000
     4.0000    0.0000   4.0000
     3.3170    2.4099   4.2000
     1.2979    3.9944   4.4000
    -1.3288    4.0895   4.6000
    -3.5597    2.5863   4.8000
    -4.5000   -0.0000   5.0000
    -3.7215   -2.7038   5.2000
    -1.4524   -4.4700   5.4000
     1.4833   -4.5651   5.6000
     3.9642   -2.8801   5.8000
     5.0000    0.0000   6.0000
     4.1260    2.9977   6.2000
     1.6069    4.9455   6.4000
    -1.6378    5.0406   6.6000
    -4.3687    3.1740   6.8000
    -5.5000   -0.0000   7.0000
    -4.5305   -3.2916   7.2000
    -1.7614   -5.4210   7.4000
     1.7923   -5.5161   7.6000
```

```
       4.7732   -3.4679   7.8000
       6.0000    0.0000   8.0000
    End
    Region NeonSpiral
    Fab   0.75
    Surface Region SolutionVolume
  End
```

Note that surface fitting may become unpredictable for a coiled object if the surfaces associated with parallel line segments are very close together. Surface fitting usually works if a neon object crosses itself (Fig. 8.2). Because the objects have no edges, the `EDGE` fitting option is ignored. You can position neon objects with the `ROTATE` and `SHIFT` commands.

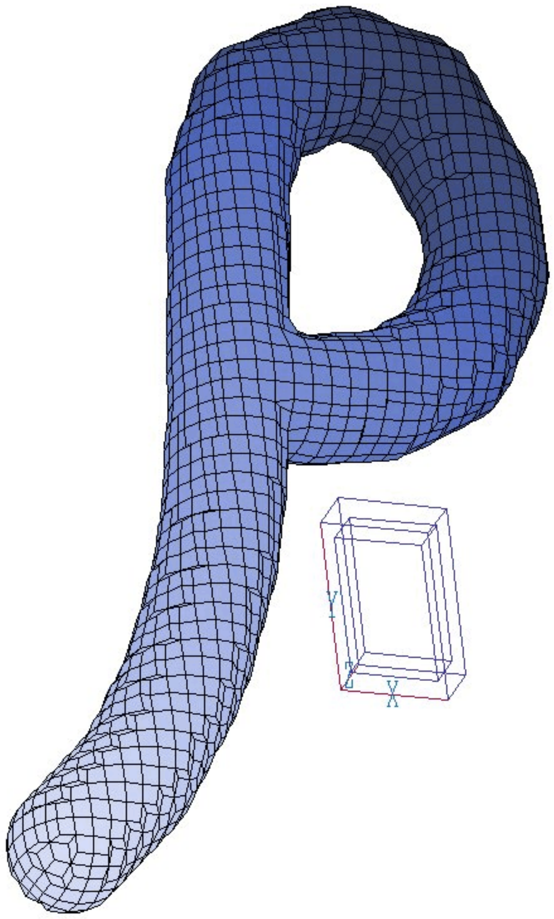


Figure 8.2. Creating a three-dimensional font.

Chapter 9. Open parts for node fitting

Open parts have no volume — they consist only of nodes. Open part fitting occurs after all filled parts have been processed. Node assignments overwrite any previous definitions. The fitting procedure consists of moving nodes to follow the desired shape and assigning *Part* and *Region* numbers. The dimensional parameters of all open parts are referenced to assembly space. In other words, a *Line* of length *L* points along the *z* axis of assembly space between the points (0, 0, -*L*/2) and (0, 0, *L*/2). You must apply `ROTATE` and `SHIFT` operations to create general lines in solution space. We have chosen this convention rather than specifying line endpoints as absolute coordinates in solution space. The reason is that you can include open parts in generalized blocks whose geometry will be preserved when positioned in assemblies by the `ROTATE` and `SHIFT` operations.

This section covers the available open parts and discusses the meaning of assembly space parameters in the `FAB` command. Note that the `SURFACE` and `COAT` commands have no effect on open parts.

Type: `POINT`
Dimensions: None
Comments: A *Point* object is created at the location (0, 0, 0) in assembly space. The *Shift* command determines the position in solution space. The *Rotate* command has no effect.

Type: `BOUNDXUP`
Dimensions: None
Comments: This command sets the region number of all nodes on the upper *x* boundary ($I = I_{max}$) of the solution space to the value specified in the *REGION* command for the part. The *ROTATE* and *SHIFT* commands have no effect. The following commands are used for the other five boundaries of the solution space: `BOUNDXDN, BOUNDYUP, BOUNDYDN, BOUNDZUP`, and `BOUNDZDN`,

Type: `LINE`
Dimensions: 1) L
Comments: The single parameter is the line length. The *Line* is created parallel to the *z* axis of assembly space between the points (0, 0, -L/2) and (0, 0, L/2). Use the *Rotate* and *Shift* operations to create general lines in solution space. If you want to connect two particular points in solution space, use the *Line converter* in the **MetaMesh** *Help* menu to find L and appropriate rotation angles and displacements. You can also use the *Line converter* to determine the endpoints of a line in solution space given its length and *Rotate* and *Shift* parameters.

Type: `ARC`
Dimensions: 1) R, 2) θ_{min} and 3) θ_{max}
Comments: The *Arc* has a radius R and lies in the *x*-*y* plane of assembly space. The arc center is at the origin of assembly space, [0, 0, 0]. The arc starts at θ_{min} and extends to θ_{max}. Enter the angles in degrees. Note that the angle $\theta = 0.0°$ corresponds to the *x* axis. You can use the *Arc Converter* in the **MetaMesh Help** menu to switch between solution space and assembly space coordinates.

Type: `CIRCLE`
Dimensions: 1) R
Comments: The *Circle* has radius R and lies in the *x*-*y* plane of assembly space. The center of the circle is at the origin of assembly space, [0, 0, 0].

Type: `RECTANGLE`
Dimensions: 1) L_x, 2) L_y
Comments: The *Rectangle* lies in the *x*-*y* plane of assembly space with sides parallel to the *x* and *y* axes. It is centered at the origin of assembly space. The *Rectangle* has length L_x along *x* ($-L_x/2 \leq x \leq L_x/2$) and length L_y along *y* ($-L_y/2 \leq y \leq L_y/2$).

Type: `DISK`
Dimensions: 1) *R*
Comments: A *Disk* is a *Circle* with the nodes filled in over the enclosed plane The object has radius *R* and lies in the *x*-*y* plane of assembly space. The center of the disk lies at the origin of assembly space, [0, 0, 0]. .

Type: `PLATE`
Dimensions: 1) L_x, 2) L_y
Comments: A *Plate* is a *Rectangle* with nodes filled in over the enclosed plane. The object lies in the *x*-*y* plane of assembly space with sides parallel to the *x* and *y* axes. It is centered at the origin of assembly space. The *Plate* has length L_x along *x* ($-L_x/2 \leq x \leq L_x/2$) and length L_y along *y* ($-L_y/2 \leq y \leq L_y/2$).

Type: `BUBBLE`
Dimensions: 1) R
Comments: A *Bubble* is a spherical surface of radius *R* with its center at the origin of assembly space.

Chapter 10. Loading, editing and processing scripts

10.1. Running under batch file control

There are two ways to run the program `METAMESH.EXE`: 1) as a non-graphical program under control of the **GCon** utility or from the command prompt or 2) as in interactive graphics program. The first mode enables runs of **MetaMeshesh** and **AMaze** solution programs under batch file control. In this case you can set up your computer to perform an extended series of operations autonomously. **MetaMesh** runs in the non-graphical mode if a file prefix is supplied when the program is called. For example, suppose you type

```
\AMAZE\METAMESH \DATA\IFACE <Enter>
```

from the Command Prompt. **MetaMesh** starts and searches for the file `IFACE.MIN` in the current or the specified directory. If successful, the program runs in the background. A batch file to create meshes and to find electrostatic solutions for a series of geometries may look like this:

```
REM Variation of focus electrode
REM Processing FELEC01
START METAMESH \GUNDESIGN\FELEC01\FELEC01
START HIPHI \GUNDESIGN\FELEC01\FELEC01
REM Processing FELEC02
START METAMESH \GUNDESIGN\FELEC02\FELEC02
START HIPHI \GUNDESIGN\FELEC02\FELEC02
REM Processing FELEC03
START METAMESH  \GUNDESIGN\FELEC03\FELEC03
START HIPHI \GUNDESIGN\FELEC03\FELEC03
REM Job completed
```

Microsoft has released over thirty versions of it’s 32-bit operating system since Windows 95. There is considerable inconsistency in DOS emulation between versions, particularly in the implementation of the *START* command. To ensure consistent batch file operation we supply the utility **GCon** with all our software. The program emulates many DOS commands and has advanced features to organize and to analyze large data runs. To avoid problems, we advise running batch scripts from **GCon** rather than from the Command Prompt.

10.2. File menu

In the remainder of this chapter we shall discuss operation of **MetaMesh** in the interactive graphical mode. This mode is invoked when the program is called with no command line parameter. When **MetaMesh** starts only the *File* and *Help* menus are active. Other menus become active when data is loaded and processing takes place. We shall begin by discussing the commands of the file menu.

Load MIN file

Pick an input script for processing. This is usually the first step in a **MetaMesh** session. The dialog box displays a list of files with names of the form `FName.MIN`. Changing directories in the dialog will also change the working directory of the program. In other words, output files will be written to the new directory.

Load MDF file

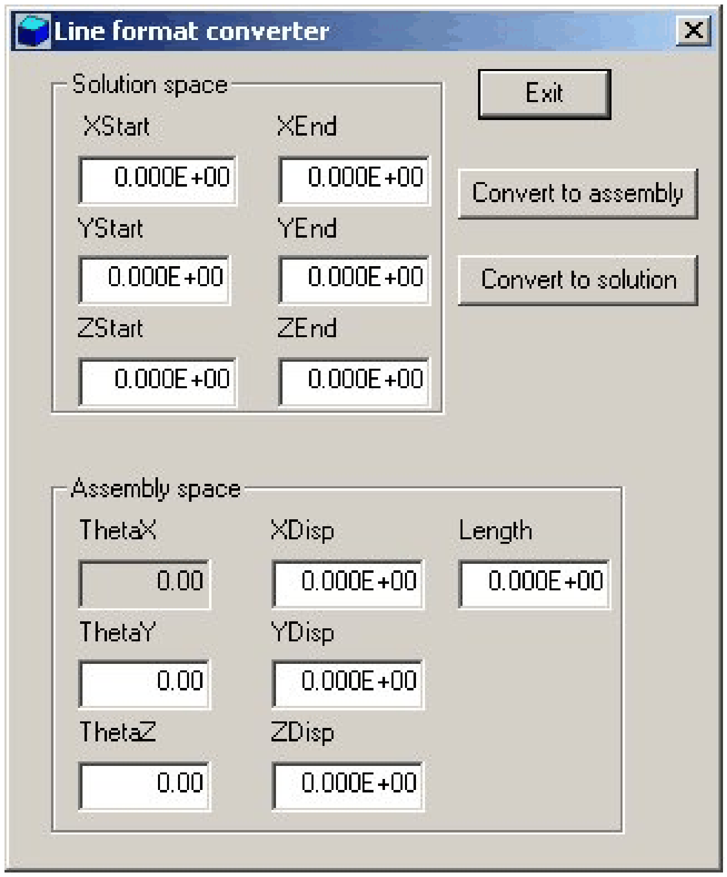
Load a previously-processed mesh file in the format described in Chapter 13. You can inspect the mesh, but it is not possible to make changes. Because of the interaction between parts in the fitting process a completed mesh cannot be modified. You must make changes in the script and then reprocess the mesh. Note that you cannot reload an `MDF` file save in ASCII format.

Save mesh

Save the currently-processed mesh in the format described in Chapter 13. The output file has a name of the form `FPrefix.MDF`, where `FPrefix` is the prefix of the currently loaded script and *MDF* standard for **M**esh **D**efinition **F**ile. The command is active only if the mesh has been processed. You will be prompted whether to save the current mesh when leaving the program or loading new files.

Edit MIN file

Open a full-featured Windows editor to view or to modify the current input script (`FPrefix.MIN`). This command is active only if a script file has been loaded. If an error occurred during script processing, the cursor will be positioned on the error line when the editor is opened.



Solution space

XStart 0.000E+00
XEnd 0.000E+00
YStart 0.000E+00
YEnd 0.000E+00
ZStart 0.000E+00
ZEnd 0.000E+00

Assembly space

ThetaX 0.00
XDisp 0.000E+00
Length 0.000E+00
ThetaY 0.00
YDisp 0.000E+00
ThetaZ 0.00
ZDisp 0.000E+00

Figure 10.1. Line format converter dialog

Edit MLS file

Open an editor to view or to modify the current listing file. This command is active only after a mesh has been processed.

Edit file

Use the Windows editor on any file. Changing directories in the dialog does not change the working directory of the program.

10.3. Help menu

The *Help* menu contains the following commands

MetaMesh manual

This command shows a cross-linked HTML instruction manual in your default browser (*i.e.*, Netscape Navigator, Internet Explorer, ...). This command will not operate unless the file `METAMESH.HTML` is in the same directory as the executable program `METAMESH.EXE`.

Line converter

The Line Converter is a useful utility for the preparation of scripts. In response to the command, the program displays the dialog of Figure 10.1. The quantities in the top group are the absolute coordinates in solution space of the starting and ending points of a line. When you click the button *Convert to assembly*, the program fills in values for the bottom group. The quantity *Length* is the line length, the single parameter in the `FAB` command for a *Line*. The quantities *XDisp, YDisp* and *ZDisp* are the three displacements in the `SHIFT` command. Finally, *ThetaX. ThetaY* and *ThetaZ* are the angles in the `ROTATE` command. Note that for a simple line only two angles are necessary for the transformation. By convention, *ThetaX* is set equal to zero and the rotations are performed in the order *ThetaY* then *ThetaZ*. If you supply values in the bottom group and click the *Convert to solution* button, **MetaMesh** calculates the absolute starting and ending points in solution space. This feature is useful to check the validity of parameters in the `ROTATE` and `SHIFT` commands.

Arc converter

The Arc Converter uses a dialog similar to the Line Converter. It is used to transform absolute coordinates for an arc in solution space (start point, end point, center point) to the `FAB`, `ROTATE` and `SHIFT` parameters for use in the MetaMesh script file. The dialog can also perform the inverse transformation.

10.4. Process mesh command

When you are satisfied that the currently-loaded script is correct, click the *Process mesh* button. The program proceeds through the volume, surface and edge fitting stages described in previous chapters. Processing aborts if **MetaMesh** encounters a syntax error in the script. In this case, when you click on *Edit MIN file*, the cursor moves to the line with the error. The complete processing history is recording in the ASCII listing file `FPrefix.MLS` (where *FPrefix* is the prefix of the script).

MetaMesh performs several checks to ensure that the completed mesh is valid and records the results in the listing file. An example is listed below:

```
Volume calculation and element integrity check
 Gaussian integration parameter, NGauss:  4
 Global volume (theoretical):   1.600000E+01
 Global volume (calculated):   1.600047E+01
 Relative error:   2.956390E-05

 NReg     Volume
 ===================
     1  1.435951E+01
     2  1.640944E+00

Element integrity checks
   Atomatic correction of distorted elements
  Pass number: 1   distorted elements:      300
  Pass number: 2   distorted elements:        4
  Pass number: 3   distorted elements:        2
  Pass number: 4   distorted elements:        0

Surface area calculation
 Gaussian integration parameter, NGauss:  4

 NReg     Surface area
 ===================
     1  8.310444E+00
     2  8.310443E+00

     2  5.163532E+02
```

Volume integrals are taken over all elements and organized by region number. Three-dimensional integrations by Gaussian quadrature are performed over each hexahedron using the normal-coordinate method. The volume integrals are organized by region number. The quantity *Global volume (theoretical)* is calculated from the formula:

$(x_{\max} - x_{\min}) \times (y_{\max} - y_{\min}) \times (z_{\max} - z_{\min})$

The quantity *Global volume (calculated)* is the summation of volumes taken over the conformal elements for all regions. The quantity *Relative error* is the difference between the two volume calculations divided by the *Global volume (theoretical)*. The program also lists volumes for individual regions.

MetaMesh reviews each element in the finished mesh for integrity by checking the edge intersections at each of the eight nodes. Given $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{u}_3$ (the edge vectors at a node in the correction order of rotation), **MetaMesh** checks that the vector product $\mathbf{u}_3 \cdot (\mathbf{u}_1 \times \mathbf{u}_2)$ is a positive number. If the condition does not hold, then the program relaxes the element node positions. By default, the process repeats for up to seven cycles or until all nodes have been corrected. You can increase the number of cycles with the *AUTOCORRECT ON* command. Note that the test for bad elements is not absolute – meshes with a few reported distortions may still give valid solutions in **HiPhi** or **Magnum**. **MetaMesh** also reports integrals of the surface area of regions. The calculation includes region surfaces on the borders of the solution volume.

For a poor choice of foundation mesh, **MetaMesh** may not be able to correct all distorted elements and the field solution program (**HiPhi**, **Magnum**, ...) may exhibit numerical instability. There are several ways to correct the condition, listed in order of severity:

- Increase the number of cycles in the *PreSmooth*, *AxisSmooth* and/or *Smooth* commands.
- Reduce the element size in the problem area.
- Reduce the fitting tolerance (ϵ_s) on the affected surface.
- Eliminate edge fitting for the affected surface.
- Eliminate surface fitting for the affected surface.

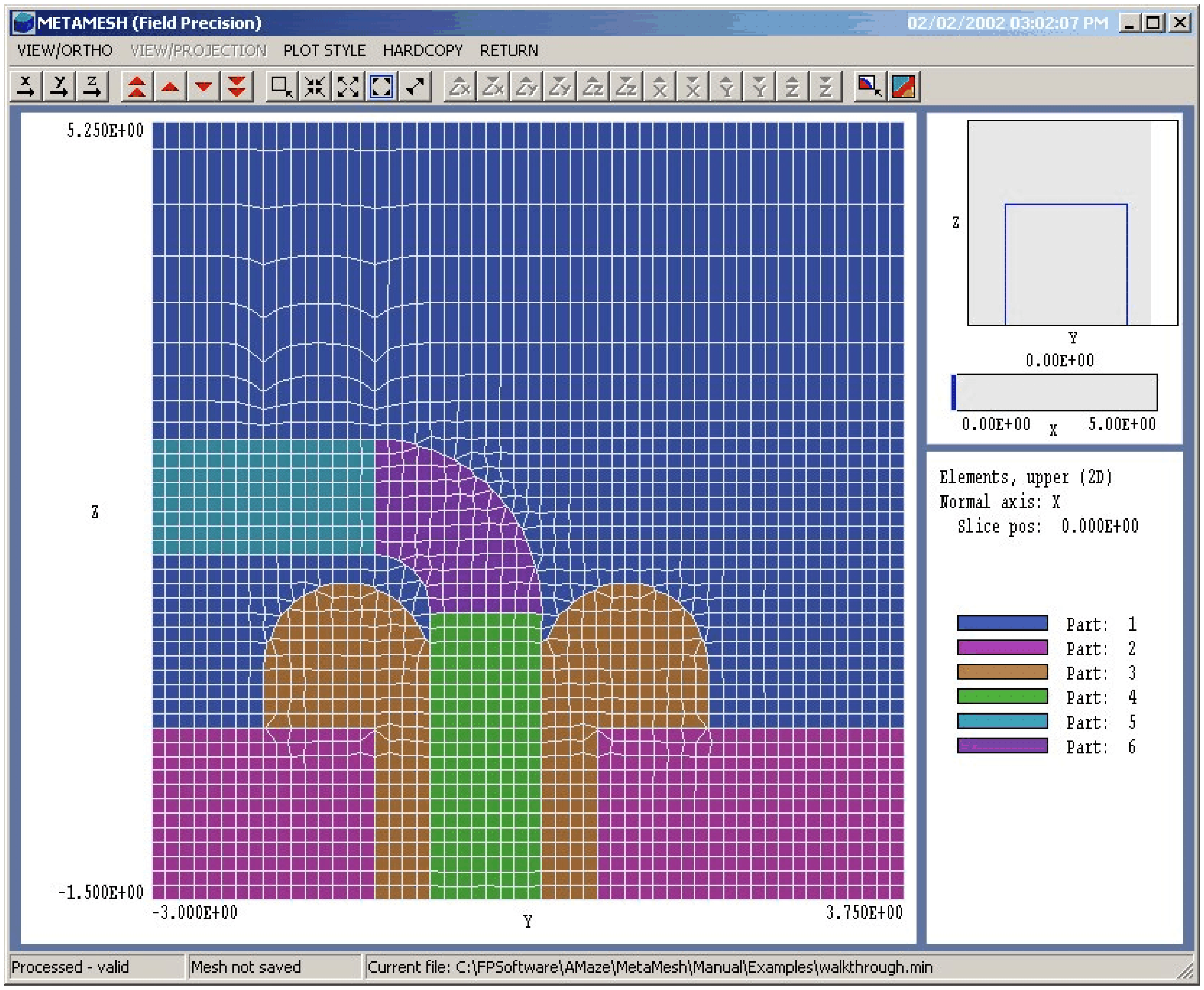


Figure 11.1. Screen display, 2D plots. Example of *Mesh/upper elements* plot.

Chapter 11. 2D plots

11.1 Logical planes and 2D plot types

After a mesh has been successfully processed, **MetaMesh** offers a variety of plotting options. This chapter covers plots of logical slices of the solution space. These plots can be generated quickly and give quantitative information on the state of the mesh in a particular plane. Click on *Plot2D* in the main menu to call up the menu and toolbars illustrated in Fig. 11.1.

The term *logical plane* implies that plotted entities are organized by their indices. For 2D plots, a logical plane is a set of nodes with one index held constant. For example, a slice normal to *x* at the index I_o contains the set of nodes (I_o, J, K), where $0 \leq J \leq J_{max}$, $0 \leq K \leq K_{max}$. In a structured conformal mesh, the nodes may not have exactly the same position along *x* but do lie close to a plane normal to *x*.

The following sections describe how to adjust plot views. In this section, we shall discuss the layout of the plot screen and the available type of 2D plots. **MetaMesh** displays a special menu and toolbar for 2D plots. The window space is divided into three areas (Fig. 11.1). The upper area to the right of the main plot is the orientation area. For orthographic projections of logical planes, the orientation area shows the location and size of the zoomed view as well as a slider indicating the position of the plane along the normal axis. For three-dimensional projection plots, the area shows reference Cartesian axes, the size of the solution volume, and the approximate limits of the current view. The lower area to the right of main plot is the information window. The content depends on the plot type. A legend is included for plots with color coding by parts or regions.

To choose the type of plot, click the *Set plot style* command. The following types are available in the resulting dialog:

Mesh

An orthographic plot of element edges in the logical plane projected to a two-dimensional normal surface. Because the plot contains three-dimensional information flattened to a plane, sometimes the view is not identical to an idea slice plot in a plane normal to the axis at a given position. In regions where the mesh is highly conformal, boundaries may have uneven edges even though the conformal mesh closely follows the boundary in an idea plane.

Mesh/nodes

This plot is similar to the mesh plot except that colored markers are added to show the part or region identity of the nodes. Figure11.2 shows the appearance of the plot.

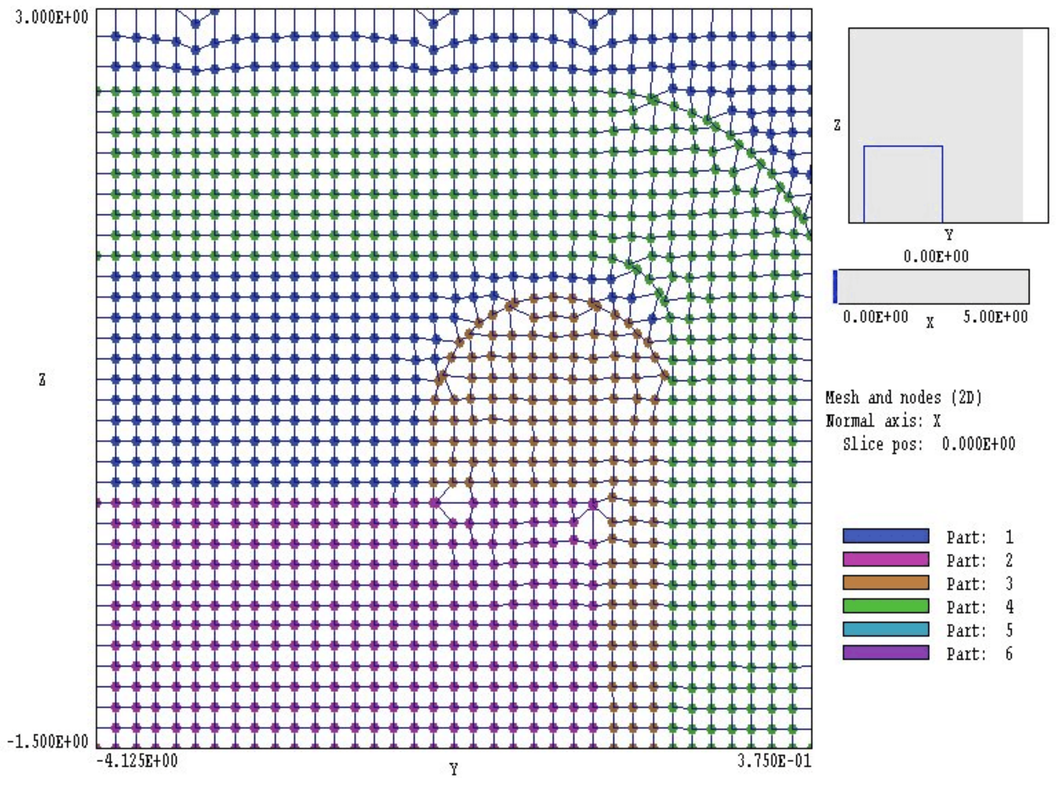


Figure 11.2. Example of *Mesh/node* plot

Mesh/upper elements

A plot of element sides in the node plane with color-coding by elements (Fig. 11.1). Note that node planes lie on boundaries between elements. Therefore, we can choose to pick colors corresponding to the part or region number of the elements on either the top or bottom of the plane. This plot shows element identities above the logical plane, where the term *above* indicates the direction of higher values of the normal axis index.

Mesh/ lower elements

This plot is the same as the *Mesh/upper elements* plot, except that color coding follows the identity of elements below the logical plane. The term *below* indicates the direction toward lower values of the normal axis index.

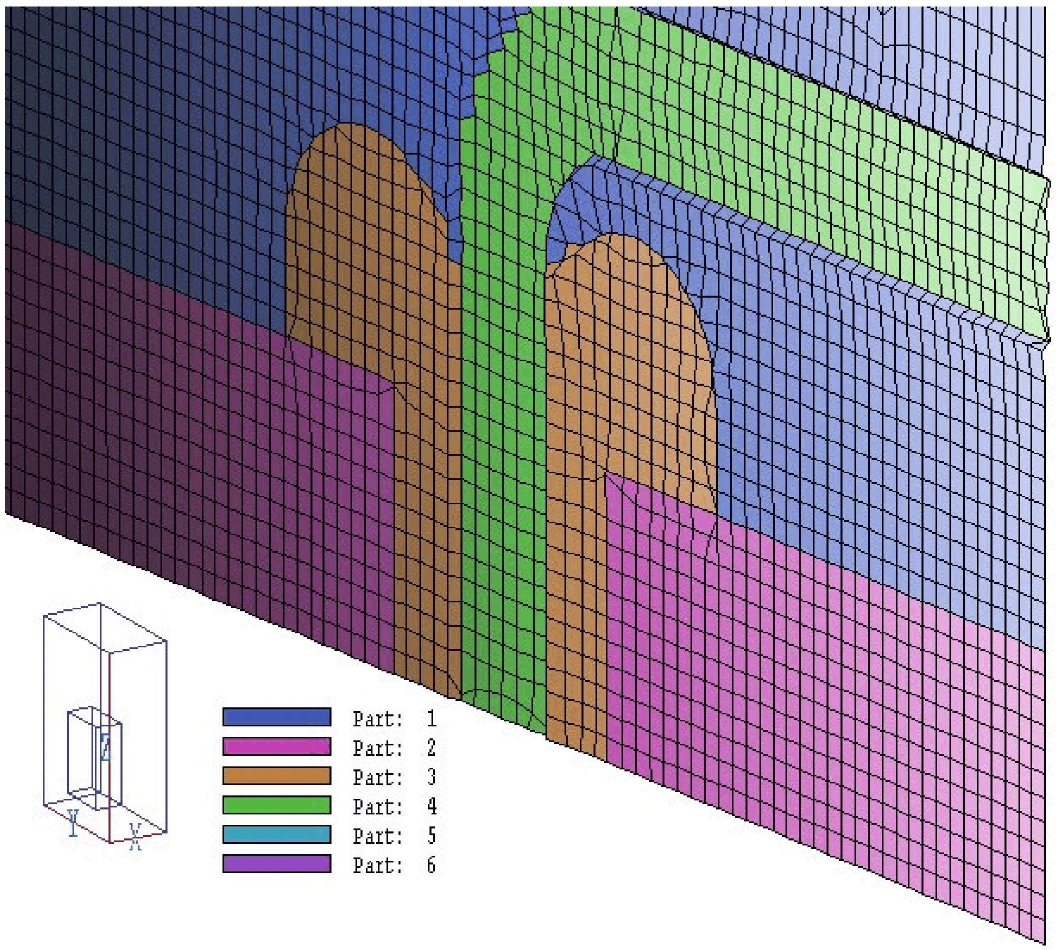


Figure 11.3. Example of *Projection/upper elements* plot

Projection/upper elements
Projection/lower elements

Projection plots show the true three-dimensional appearance of logical planes. Several commands and tools are available to adjust the view. Figure 11.3 shows an example. Again, we have the choice of whether to display color-coded information on elements above or below the logical plane.

11.2 Adjusting the view - orthographic plots

We can divide two-dimensional plot types into two groups. Orthographic plots include the types *Mesh*, *Mesh/nodes*, *Mesh/ upper elements* and *Mesh/ lower elements*. Three-dimensional projection plots include *Projection/ upper elements* and *Projection/ lower elements*. The available view adjustment comments depend on the plot group. In this section, we shall discuss commands in the *View/ortho* menu. This menu is active only when orthographic plots are displayed.. The first set of commands is used to pick the display plane.

Jump forward
Step forward
Step backward
Jump backward

These commands control motion along the normal axis. The *Jump forward* command increases the index of the plane by 5. In other words if you are looking at a plane normal to *z* with index k, then the view changes to a plane with index (*k*+5). The other commands have the following effects: *Step forward* (*k* = *k*+1), *Step backward* (*k* = *k*-1) and *Jump backward* (*k* = *k*-5).

Normal plane X
Normal plane Y
Normal plane Z

Change the normal plane axis. **MetaMesh** automatically sets the plane position to the midpoint along the new normal axis.

Set plane

This command calls up a dialog where you can change the normal axis and also set the plane position along the axis by moving a slider.

The next set of commands in used to adjust the view within a logical plane.

Zoom window

You can take a closer look at an area in the plane by specifying a view box with the mouse or keyboard. After clicking on the *Zoom window* command, move the mouse into or near the main plot area. The pointer changes to a cross-hair symbol when it is inside the plot boundaries.

Note that the status bar at the bottom of the screen changes to the coordinate mode, displaying the current mouse coordinates and the mouse snap status. Click the left button to set a corner. Move the mouse to define a box and click the left button again to define a box. Click the right button or press *Escape* to cancel the operation. If you click the left button outside the plot area, the program picks the closest point on the boundary. You can enter coordinates from the keyboard by pressing the *F1* key when the mouse is active. Note that the outline of the current view appears in the orientation area.

Zoom in

Magnify the plot about the current center-of-view.

Zoom out

Expand the plot about the current center-of-view.

Global view

Reset the view to show the full plane.

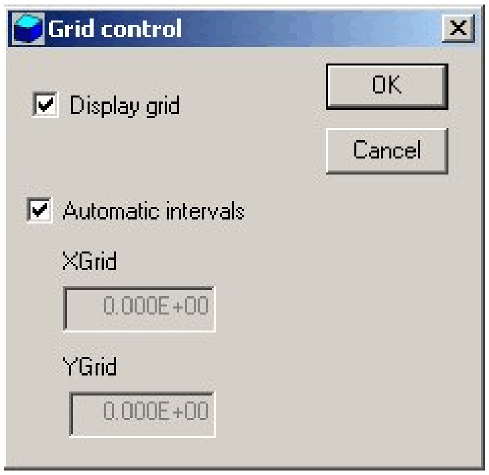
Pan

Use the mouse to shift the view point in the slice plane. Click the left button once to set a reference point. Click it a second time to define an offset of the view point. You can also use the keyboard to enter coordinates by pressing the *F1* key when the mouse is active.

The final set of commands in *View/ortho* menu are used to set plot features to request element information

Grid display

The program displays the dialog shown in Fig. 11.4 to control the display of the plot grid. Under the *Automatic* option, **MetaMesh** picks convenient intervals for the grid display. Alternatively, you can set intervals manually. The grid intervals are listed in the information window.



[✓] Display grid

[✓] Automatic intervals

XGrid: 0.000E+00

YGrid: 0.000E+00

Figure 11.4 Grid control dialog

Region/part display

This command toggles color coding by parts or regions in plots of the type *Mesh/nodes*, *Mesh/upper element* and *Mesh/lower element.* The information window shows the current status.

Toggle snap mode

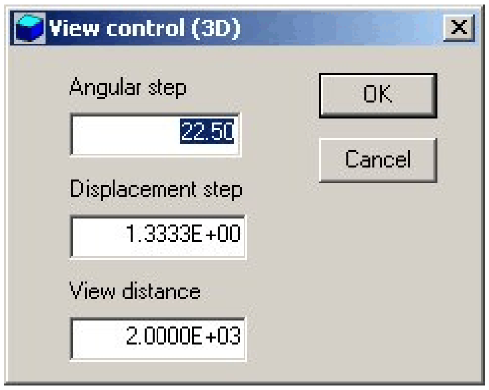
The mouse snap mode is a convenient feature for creating zoomed views. When snap mode is active, the mouse returns the coordinate values closest to an integer multiple of the quantity *DSnap*. In other words, if *DSnap* = 0.5 and the mouse position is [5.4331,-2.6253], the returned coordinates are [5.5,-2.5]. The coordinate display in the status bar shows the snapped values.

Set DSnap

Set the distance scale for the mouse snap mode.

Element info

You can get detailed information on elements above or below the logical plane (in *Mesh/upper elements* and *Mesh/lower elements* type plots) with this command. Move the mouse to a position inside the element boundary and click the left button. The program shows the indices, region and part numbers, node locations, volume and surface area of the element.


View control (3D)

Angular step

22.50

Displacement step

1.3333E+00

View distance

2.0000E+03

Figure 11.5. View control dialog

11.3 Adjusting the view - projection plots

The *View/projection* menu is active when the plot type is either *Projection/upper element* or *Projection/lower element.* The menu contains the standard set of commands to set the logical plane described in proceeding section. There is also active control pad that appears between the orientation and information windows on the right-hand side of the screen. The mouse cursor changes to a four-arrow pattern within the control pad. Click on the red arrows to control translation and rotation of the coordinate box in the orientation window. Hold the left mouse button down for continuous motion. When you are satisfied with the altered view, click the right mouse button or left-click on the DRAW command to update the main plot. Note that the same control pad is used for the 3D plots described in Chapter 12. The buttons in the control pad perform the following functions:

Zoom (Zm)
Enlarge or reduce the size of the view box.

RotateX(Rx)
Rotate about the x axis. The default angular step is 22.5°. You can change the angular step with the *View control* command.

RotateY(Ry)
Rotate about the y axis.

RotateZ(RZ)
Rotate about the z axis.

ShiftX(X)
Move the view point along x. You can change the displacement step with the *View control* command.

ShiftY(Y)

Move the view point along *y*.

ShiftZ(Z)

Move the view point along *z*.

The following command is used to set parameters for projection plots.

View control

This command calls the dialog of Fig. 11.5 to control the 3D view tools. You can enter new values for the angular displacement in for rotations and the linear displacement for displacements. The third parameter, *DView*, controls plot perspective, a useful adjustment for the 3D surface plots discussed in the next chapter. The quantity *DView* equals the distance from the view point from the center of the plot. It is normally set to large number to give an isometric plot. Lower the value for more perspective. Choosing a viewpoint inside the assembly may result in an invalid plot.

11.4 Exporting plots

The commands of the *Export plot* menu offer several options to create plot output.

Default printer

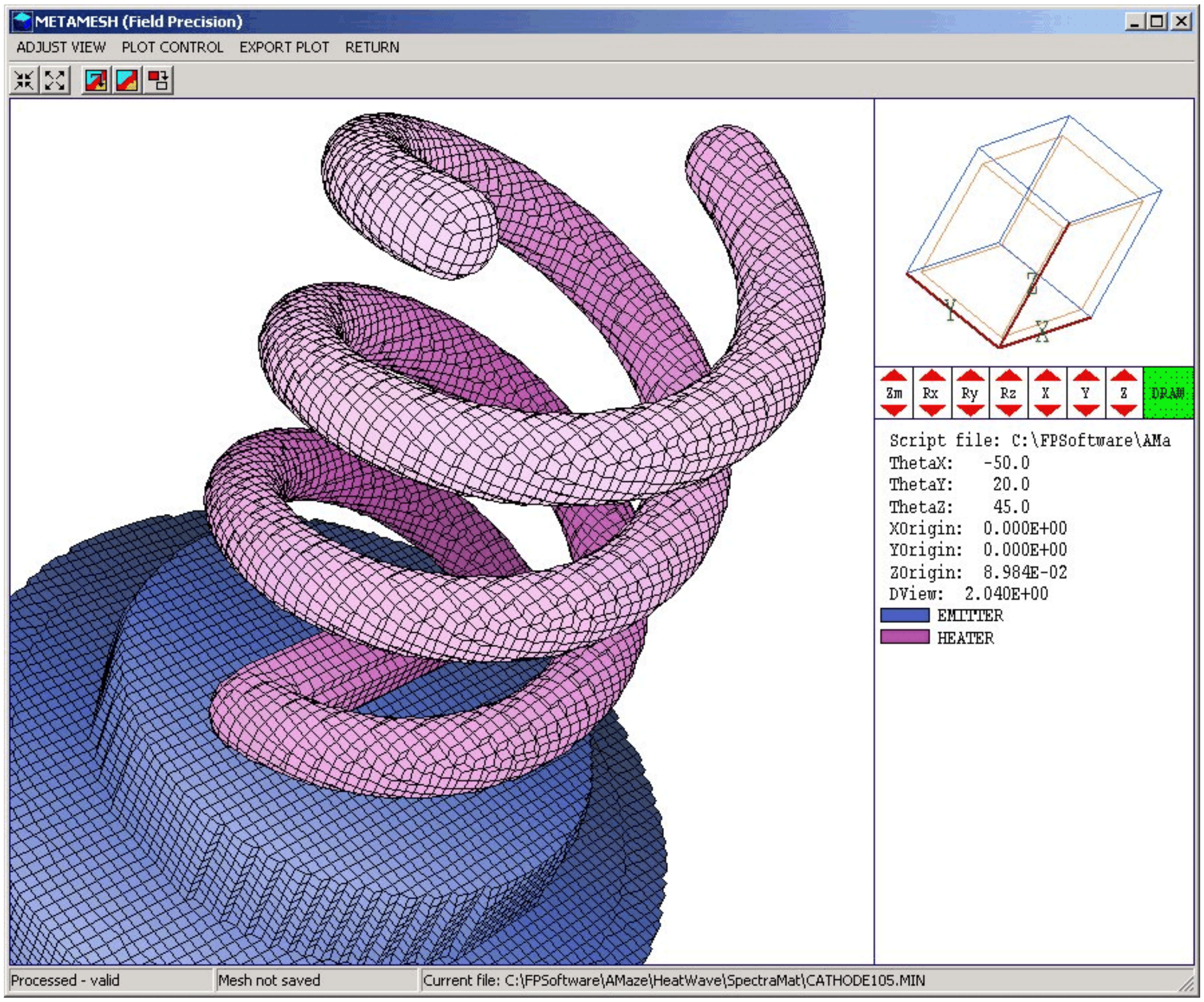
This command sends a copy of the current plot to the default Windows print driver. If you have several printers, use the *Settings/Printers* option on the *Start Menu* to make changes in the default before using this command. The plot will be in full color if you have a color printer with the correct Windows driver. With the appropriate print drivers, you can create PDF files or send the information to a network printer.

Plot file (EPS)
Plot file (BMP)
Plot file (PNG)

With this command you can create a graphics file of the current plot in either Encapsulated PostScript, Windows BitMap or Portable Network Graphics formats. The program prompts for a file prefix. The graphics files are created in the current directory with names of the form `FPrefix.EPS`, `FPrefix.BMP`. or `FPREFIX.PNG`.

Copy to clipboard

This command copies the current plot to the Windows clipboard in Windows MetaFile format. You can then paste the information into any graphics program.



```
Script file: C:\FPSoftware\AMa
ThetaX:   -50.0
ThetaY:    20.0
ThetaZ:    45.0
XOrigin:  0.000E+00
YOrigin:  0.000E+00
ZOrigin:  8.984E-02
DView:  2.040E+00
```

🟦 EMITTER
🟪 HEATER

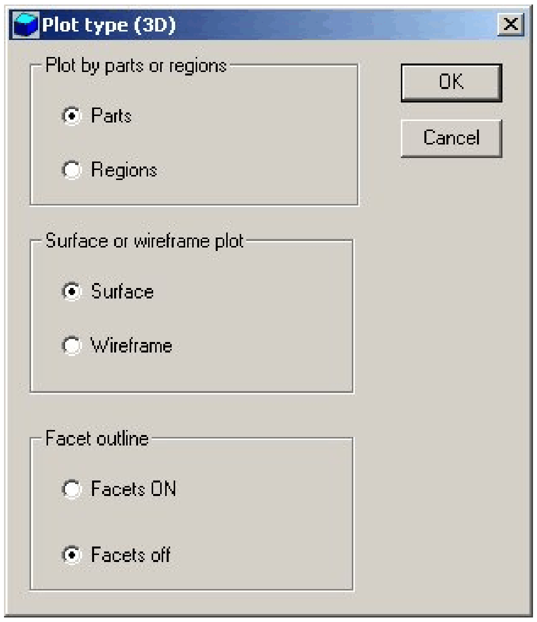
Processed - valid | Mesh not saved | Current file: C:\FPSoftware\AMaze\HeatWave\SpectraMat\CATHODE105.MIN

Figure 12.1. Screen display, 3D plots

Chapter 12. Three-dimensional plots

12.1 Surface plot methods

If you click on *Plot 3D* after mesh processing, **MetaMesh** displays the toolbar and menus illustrated in Fig. 12.1. In this mode you can create three-dimensional plots of the surfaces of filled parts or regions in the solution volume. In this section we shall briefly discuss the nature of the surface plots.


Plot type (3D)

Plot by parts or regions

[✓ Parts] Regions

Surface or wireframe plot

[✓ Surface] Wireframe

Facet outline

Facets ON [✓ Facets off]

Figure 12.2. Three-dimensional plot style dialog

Suppose we wanted to plot the shape of part number 8. The *surface* of the part is defined as the set of all facets of elements that have *PartNo* = 8 that border on elements with *PartNo* ≠ 8. Similarly, the surface of region 3 consists of the set of facets of elements that have *RegNo* = 3 that border on elements with *RegNo* ≠ 3. To create a plot, **MetaMesh** searches the mesh to identify target elements for one or more parts (or regions) and then collects facets that meet the criterion. The program sorts the facets in order of distance from a viewpoint (at a distance *DView*) and then plots the set starting with facets at the greatest distance. The procedure reliably creates hidden surface plots for complex geometries. **MetaMesh** color-codes the facets by part or region number and also by distance to add a sense of depth. Facets on the boundaries of the solution volume are not included.

The procedure has two implications for the user. First, when you adjust the view or style the plot regeneration time depends on the nature of the change. For example, if you modify the list of plotted parts **MetaMesh** must repeat facet collection, recalculate the distances, sort the set and then generate the plot. If you rotate the plot, the program can use the same set of facets but must update the distances and sort order before plotting. Finally, if you switch the plot style from surface to wireframe, the program can refresh the plot immediately with the same facet set and order. The second implication is that the procedure affects the display of facets when parts or regions share a boundary. Depending on the mesh

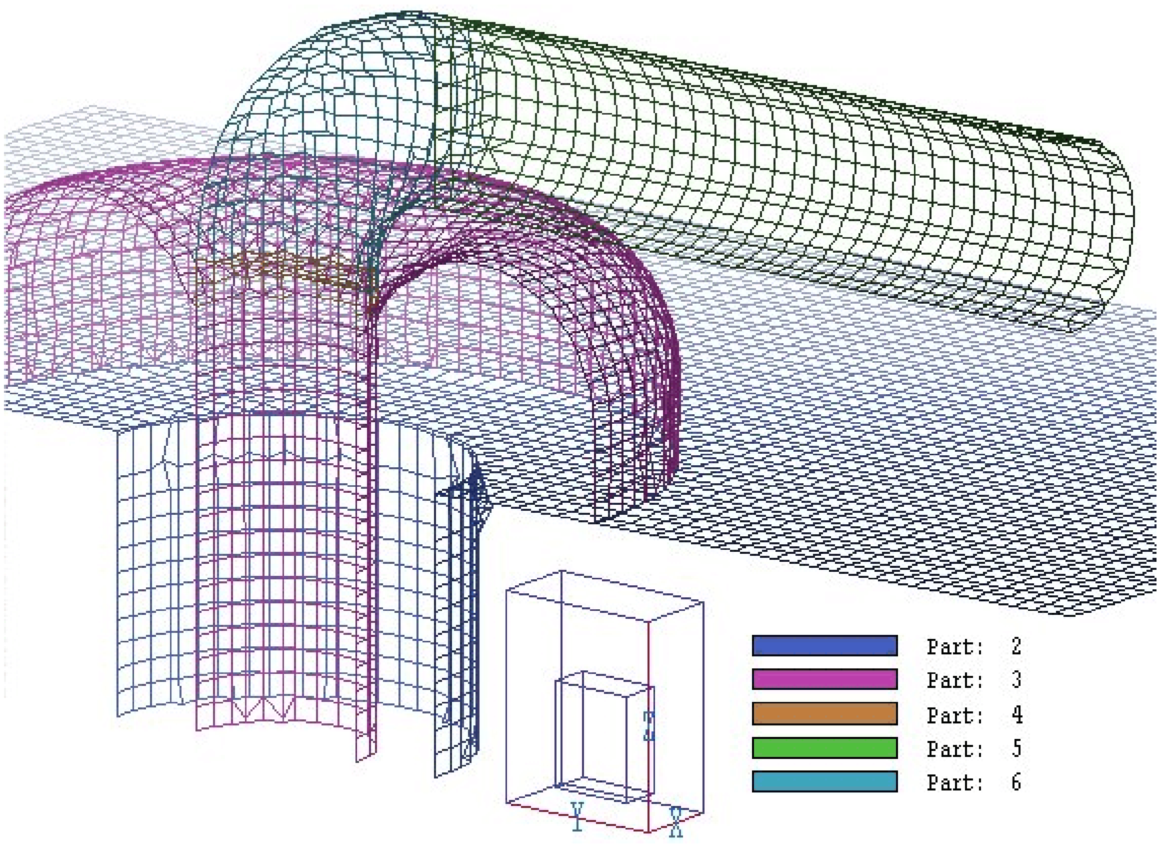
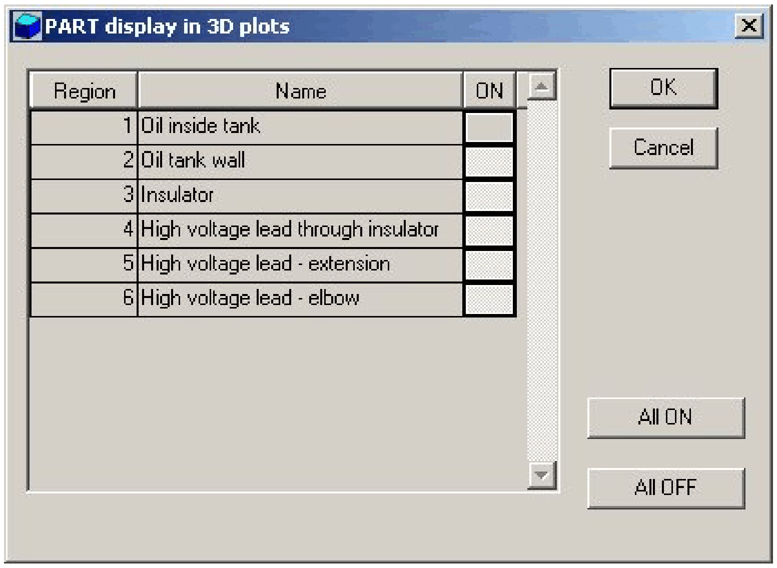


Figure 12.3. Wireframe plot

parts or regions share a boundary. Depending on the mesh scan order and calculated distances, facets on a boundary may be plotted with colors associated with either adjacent part or region.

You can include open parts in three-dimensional plots. For open parts, **MetaMesh** plots the nodes as colored marker points. If two nodes of an open part are neighbors on the logical mesh, the program plots a light gray line to show the connection. Figure 5.8 illustrates the 3D plot with open parts. Note that open parts are not included in the hidden surface algorithm, so they are visble behind filled part surfaces. (Note that open parts are displayed only when surfaces are plotted by Part Number.)

The control panel to adjust the three-dimensional view was described in Section 10.3. In this chapter we shall concentrate on 3D plot options not available in the 2D projection plots.



Region	Name	ON
1	Oil inside tank	
2	Oil tank wall	
3	Insulator	
4	High voltage lead through insulator	
5	High voltage lead - extension	
6	High voltage lead - elbow	

Figure 12.4. Region/part display dialog

12.2 Plot control menu

With the commands of the *Plot control* menu you can plot styles and control the display of part or region surfaces.

Plot style

This command calls the dialog shown in Fig. 12.2. The radio buttons labeled *Parts* and *Regions* determine whether the program will pick facets and apply color-coding according to the part number or the region number of the elements. The buttons *Surface* and *Wireframe* determine the plot style. Figure 12.1 shows the surface plot style, while Fig. 12.3 shows a wireframe plot. The wireframe plot regenerates quickly but it may be confusing for complex assemblies. A good practice is to line up a view in the wireframe mode and then to switch to surface mode for presentation. The final option is whether to display the boundaries of facets. Figure 12.1 shows a surface plot with facet boundaries included while Fig. 1.2 shows a plot with boundaries removed.

Region/part display

The command displays the dialog of Fig. 12.4. The example illustrates the reward for including *PartName* and *RegName* commands in the assembly script. The descriptive names are displayed in the dialog.

The information can be helpful when organizing complex 3D plots. When the program starts the default is to plot surfaces by part number and to display the part with *PartNo* = 2. The motivation for this choice is that *PartNo* = 1 often represents the solution volume and would not appear in a surface plot. Note that the plotting procedure uses dynamic memory allocation to store facets. Depending on the installed memory of your computer, you may receive an error message if you try to display too many features in a large mesh.

View control

This command was discussed in Sect. 10.3. You can adjust the perspective of the plot by changing values of the parameter *DView*. A good choice is to set *DView* approximately equal to the maximum spatial width of the displayed facet set.

Set cut planes

In a hidden surface plot, internal details may be obscured by surrounding parts or regions. This command brings up a dialog that allows you to adjust the display areas along the *x*, *y* and *z* axes. **MetaMesh** does not display facets that lie outside the limits. With this feature you can create cutaway views — Fig. 2.6 shows an example. When a mesh is processed, the default is to set the cut limits equal to the dimensions of the solution volume so that all facets are included.

Chapter 13. Formats of the MetaMesh output files

Nodes are referenced with the indices [*I, J, K*] where *I* (the index along the *x* axis) extends from 0 to I_{max}, *J* (*y* axis) from 0 to J_{max}, and *K* (*z* axis) from 0 to K_{max}. The number of elements is approximately equal to the number of nodes. A single element (in the direction of positive *x, y* and *z*) is associated with each node for storage.

MetaMesh assigns integer *region numbers* to nodes and elements to associate them with structures in the solution space. For example, in an electrostatic solution all nodes and elements that constitute an electrode would share the same region number. Physical properties are associated with regions only in subsequent solution programs. Therefore, the operation of **MetaMesh** is independent of the nature of the physical solution — the program can be applied to any type of finite-element calculation (electrostatics, magnetostatics, electromagnetics, mechanics, thermal transport,...).

The format of the **MetaMesh** binary output file is simple and compact, making it easy to transfer information to other programs. The following code extract comprises the entire output algorithm:

```
WRITE (OutMDF) IMax,JMax,KMax
DO K=0,KMax
 DO J=0,JMax
   DO I=0,IMax
    WRITE (OutMDF) &
      M(I,J,K).x,M(I,J,K),y,M(I,J,K).z, &
      RegNo(I,J,K),RegUp(I,J,K)
   END DO
 END DO
END DO
```

The recorded quantities in each data line are:

- The spatial coordinates of the node (*x, y, z*)
- The region number of the node (*RegNo*)
- The region number of the associated upper element (*RegUp*)

The **MetaMesh** binary output file has a name of the form `FPrefix.MDF`, where *FPrefix* is the prefix of the script file (1-20 characters). File quantities have the following definitions consistent with FORTRAN 95 for I386 standard computers:

$I_{\max}$, $J_{\max}$: $K_{\max}$: 4 byte integer
x, y, z: 4 byte real
RegNo, RegUp: 1 byte integer

An output file in ASCII format has a name of the form `FPrefix.MTF` (**M**esh **T**ext **F**ile). The following shows the first section of an ASCII output file:

```
MetaMesh Output File - ASCII format (Field Precision)
IMax:       20
JMax:       20
KMax:       40

    I       J       K          x            y            z       RegNo RegUp
=========================================================================
       0       0       0 -5.0000E+00 -5.0000E+00 -1.0000E+01     1     1
       1       0       0 -4.5000E+00 -5.0000E+00 -1.0000E+01     1     1
       2       0       0 -4.0000E+00 -5.0000E+00 -1.0000E+01     1     1
       3       0       0 -3.5000E+00 -5.0000E+00 -1.0000E+01     1     1
       4       0       0 -3.0000E+00 -5.0000E+00 -1.0000E+01     1     1
       5       0       0 -2.5000E+00 -5.0000E+00 -1.0000E+01     1     1
       6       0       0 -2.0000E+00 -5.0000E+00 -1.0000E+01     1     1
       7       0       0 -1.5000E+00 -5.0000E+00 -1.0000E+01     1     1
       8       0       0 -1.0000E+00 -5.0000E+00 -1.0000E+01     1     1
       9       0       0 -5.0000E-01 -5.0000E+00 -1.0000E+01     1     1
      10       0       0  9.0298E-08 -5.0000E+00 -1.0000E+01     1     1
      11       0       0  5.0000E-01 -5.0000E+00 -1.0000E+01     1     1
      12       0       0  1.0000E+00 -5.0000E+00 -1.0000E+01     1     1
      13       0       0  1.5000E+00 -5.0000E+00 -1.0000E+01     1     1
```

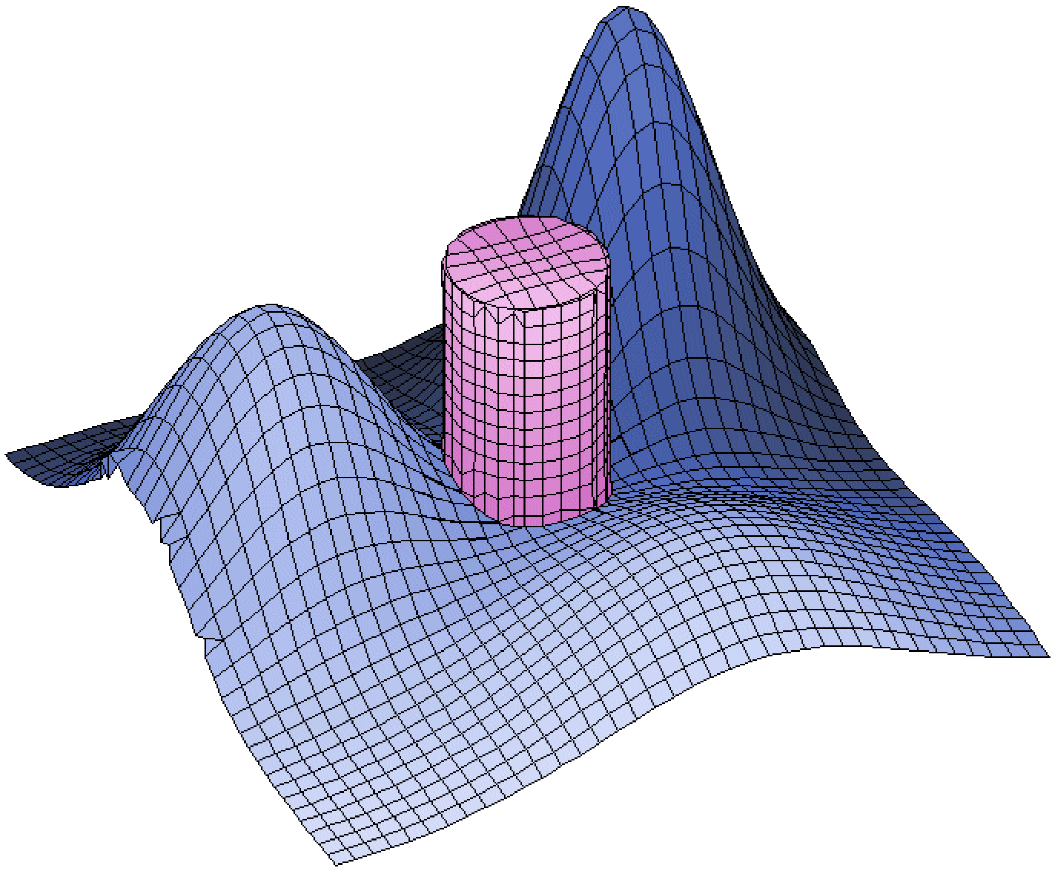


Figure 14.1. Mesh for the `MOUNTAINS` example.

Chapter 14. Fitting mathematically-generated surfaces

Some applications call for complex three-dimensional surfaces that may be difficult (or even impossible) to construct using the models dicussed in the previous sections. Examples include a focusing electrode for an electron gun with non-circular cathode, construction of a topography from map data or a representation of a mathematical function. The *SURF3D* model creates a solid with a surface defined by a table of elevation values. The procedure produces high-accuracy meshes with minimal element distortion.

A *PART* section for a three-dimensional surface has the following components:

```
PART
  TYPE Surf3D DataFileName
  FAB ZMin
  NAME PartName
  REGION RegNo
  SHIFT Xs Ys Zs
  ROTATE XRot YRot ZRot
  SURFACE [Part, Region] [PartNo, RegNo] 1.00
END
```

The first two lines are required, the others are optional. The quantity *DataFileName* is the name of a file in the current directory that contains elevation values $z(x,y)$ in the format described below. The single fabrication parameter *ZMin* gives the lower surface of the solid. The solid extends from *ZMin* to $z(x,y)$ along the z axis of the assembly frame. A three-dimensional surface can be shifted or rotated for placement in the solution space, just like any other part. For example, to define a solid where the mathematical surface faces the $-z$ direction, use *RotX* = 180.0 ° and shift the solid in z to the desired location.

When a *SURFACE* command appears in the *PART* section, fitting is applied only to the $z(x,y)$ surface of the solid. There is no surface or edge fitting on the bottom and sides along x and y. For smooth surfaces, the fitting procedure usually does not introduce element distortions. Therefore, we recommend that you use the maximum fitting tolerance of 1.00 in the *SURFACE* command.

A script may contain up to five *SURF3D* parts. **MetaMesh** issues an error message if any portion of a defined surface is less than *ZMin*. Parts that appear latter in the script may penetrate the surface or may be used to cut off a portion of the surface solid (Fig. 14.1).

The elevation file is in free-form text format. The data are tabulated over a rectangular mesh with the conventions shown in Fig. 14.2. The quantity I (column index, x direction) extends over the range $0 \leq I \leq I_{max}$, while J (row index, y direction) has values in the range $0 \leq J \leq J_{max}$. Elevation values $z(I,J)$ are tabulated at positions $[x(I), y(I)]$ where:

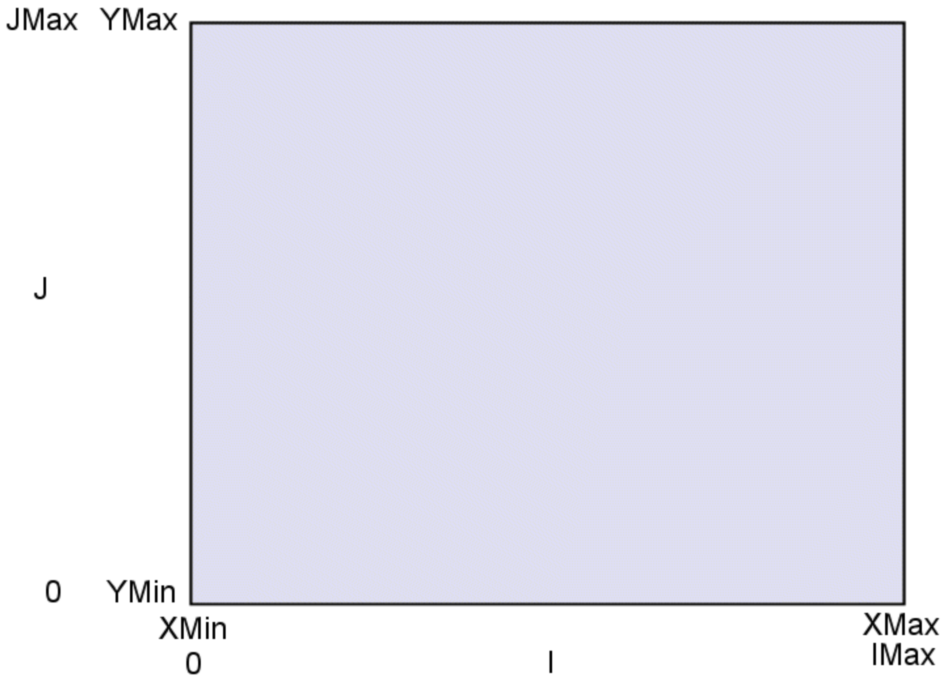


Figure 14.2. Conventions for tabulating values in the elevation data file.

$$\Delta x = \frac{X_{\max} - X_{\min}}{I_{\max}} , \qquad \Delta y = \frac{Y_{\max} - Y_{\min}}{J_{\max}} ,$$

$$X(I) = X_{\min} + I\Delta x ,$$

$$Y(J) = Y_{\min} + J\Delta y .$$

The file consists of two header lines and $(I_{\max}+1)(J_{\max}+1)$ data lines:

```
IMax,JMax
XMin,YMin,XMax,YMax
Z(0,0)
Z(1,0)
   ...
Z(IMax,0)
Z(0,1)
   ...
Z(I,J)
Z(IMax,JMax)
```

The quantities Imax and Jmax are integers. All other quantities are real numbers in any valid format. The following rules apply:

- ■ Numbers in the header lines can be separated by any of the valid **AMaze** delimiters (space, comma, equal sign, tab, left paren, right paren).

- ■ Comment lines beginning with '*" (asterisk) can be inserted before the header.

- ■ Annotations in any format can be included after the last data line.

MetaMesh has comprehensive syntax-checking features. The program reports any errors encountered reading the file.

We have included a Perl script (`surfgen.pl`) that can be modified to construct elevation files for any mathematical function *z*(*x*,*y*). As supplied, the program creates the elevation file `MOUNTAINS.DAT` used as input for the demonstration file `MOUNTAINS.MIN`. The results are illustrated in Fig. 14.1. All files are included in the **MetaMesh** example library.

Type: SURF3D
Dimensions: 1) *ZMin* (lower bound)
Comments: The solid extends from *ZMin* to *z*(*x*,*y*) in the assembly frame, where the elevation values are specified in a data file. The solid covers a rectangular region in the *x*-*y* plane. The limit of the rectangle (*XMin, YMin, XMax, YMax*) are specified in the data file header.