Field Precision LLC

MetaMesh 4.0

Three-dimensional conformal mesh generator

PO Box 13595, Albuquerque, NM 87192 U.S.A.
Telephone: +1-505-220-3975
Fax: +1-617-752-9077
E mail: techinfo@fieldp.com
Internet: http://www.fieldp.com

Contents

1 Introduction

1.1 Program function

The **MetaMesh/Geometer** package is the core of the **AMaze** series of three-dimensional finite-element programs. The function of the programs is to divide solution spaces into conformal, hexahedron[1] elements to match system geometries that you specify. There are two reasons why finite-element calculations always begin with the process of *mesh generation*:

- The shapes and material identities of the elements encode information on the geometry of the physical system.
- Dividing the solution space into small pieces makes it possible to convert the continuous differential equations that govern field variations into a set of coupled linear equations that can be solved on a digital computer.

Geometer is an interactive program with a graphical interface. In the program you can construct complex objects as you move freely through three-dimensional space with advanced visualization tools. The end product is a text script of information that describes the geometry. **MetaMesh** analyzes the script and divides space into a set of conformal elements with unique material identities. The output file of **MetaMesh** provides geometric information to the **AMaze** solution programs (**HiPhi**, **Magnum**, **OmniTrak**, **RFE3** and **HeatWave**) or the **GamBet** Monte-Carlo program. You can also use **Geometer/MetaMesh** as a stand-alone package to create meshes for your own finite-element programs.

Although we live in a three-dimensional world, describing precise three-dimensional structures is a challenging process. Three-dimensional simulations take careful planning and methodical work. Within this constraint, we believe that the **MetaMesh/Geometer** package offers unparalleled ease-of-use and a minimum of frustration. **Geometer** has several unique features:

- Employs OpenGL to give you sense of being inside the system with complete freedom to move about. The program also offers a variety of orthographic views for precision work.
- Has an extensive set of interactive dialogs that you can use to add, to modify or to reorder parts in a few seconds.
- Includes an interactive two-dimensional CAD system to specify cross-sections of extrusions and turnings.

In comparison to other available mesh generators, **MetaMesh** has the following advantages:

- The input script constitutes a readable and succinct format to record generalized three-dimensional geometries. The script gives you complete access to input information – you can directly create or modify scripts with an editor.

[1]A *hexahedron* is a generalized six-sided solid figure, basically a box with arbitrary facets

- The operational strategy is to create complex objects step-by-step from simple parts. The activity of fabricating, orienting and positioning parts is similar to the process of building an assembly in a machine shop.
- Features direct import of complex solids from three-dimensional CAD programs like SolidWorks, ProE, AutoCAD and NX through STL files.
- Has advanced capabilities to integrate non-mechanical objects. The program can generate parts with arbitrary surfaces that are specified mathematically or numerically. The program can also represent biological systems by direct conversion of voxel data sets (such as MRI scans) in Analyze format.
- Generates unique structured conformal three-dimensional meshes, giving top speed in solution programs.
- Powerful block operations enable you to paste standard assemblies into a script, moving and rotating the collection of parts as a group.
- Open and well-documented output formats.
- Comprehensive visualization features for checking the accuracy and integrity of meshes.
- Runs in the interactive mode or autonomously under batch file control.
- Incorporates extensive error checking and element correction routines.

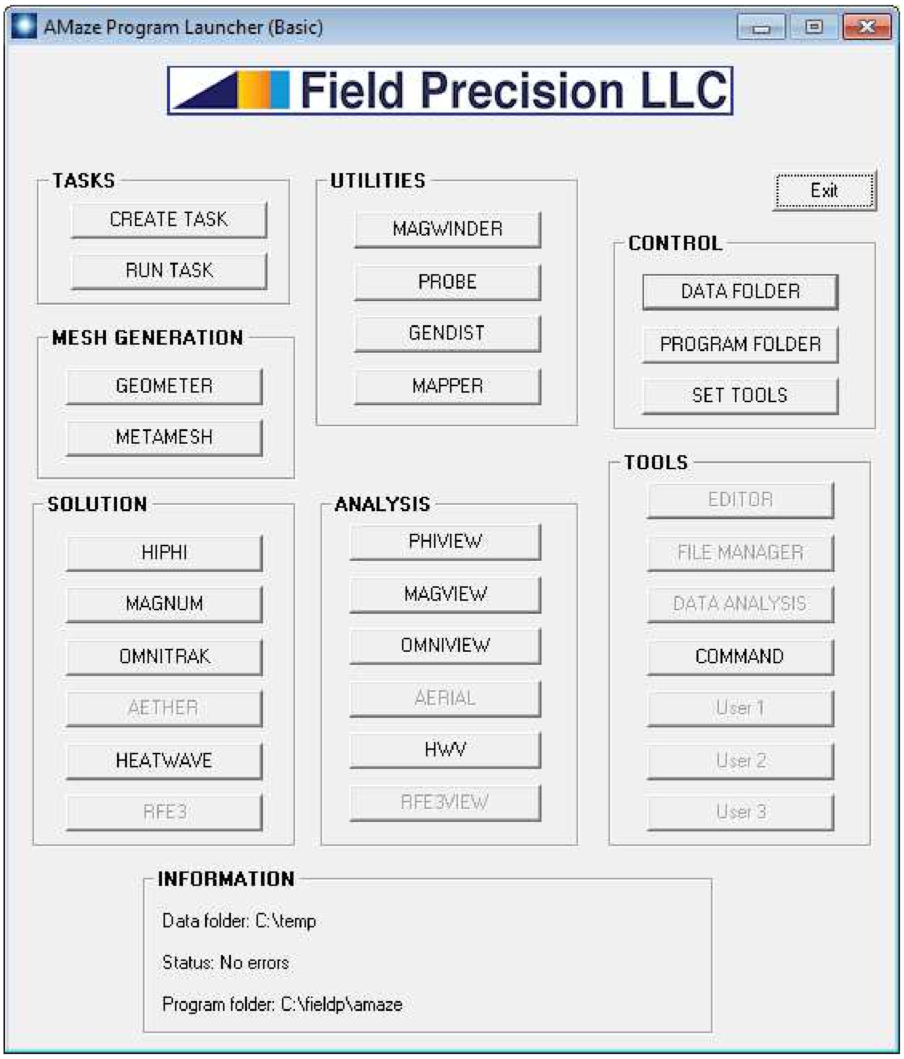
1.2 AMaze program controller

The first step to run programs is to set up the **AMaze** program controller. **AMaze** helps you to organize your work in two ways:

- Only one shortcut is required on your desktop, minimizing clutter.
- All programs open in a specified data directory, eliminating redundant trips through the directory tree.

The installation package has created a shortcut to `\fieldp\amaze.exe` on the Windows desktop and start menu. Figure 1 shows a screen shot of the **AMaze** dialog. It is divided into seven functional groups: *Mesh generation*, *Solution*, *Analysis*, *Control*, *Utilities*, *Tools* and *Information*. The first three groups reflect the activities of an **Amaze** simulation: first create a mesh and then proceed to the physical solution and analysis.

Note that only some of the command buttons are active. **AMaze** activates buttons only if it detects the corresponding executable in the *Program Folder*. The button for **MetaMesh** should always be active, while entries in the *Solution*, *Analysis* and *Utilities* sections depend on which programs you have purchased. If no buttons are active, **AMaze** is not pointing to the directory where you installed the executable programs. In this case, click the *Program Folder* button. In the dialog move to the directory that contains the executables and click *OK*. The corresponding buttons in the *Mesh*, *Solution*, *Analysis* and *Utilities* areas will become active.



Data folder: C:\temp

Status: No errors

Program folder: C:\fieldp\amaze

Figure 1: **AMaze** program controller

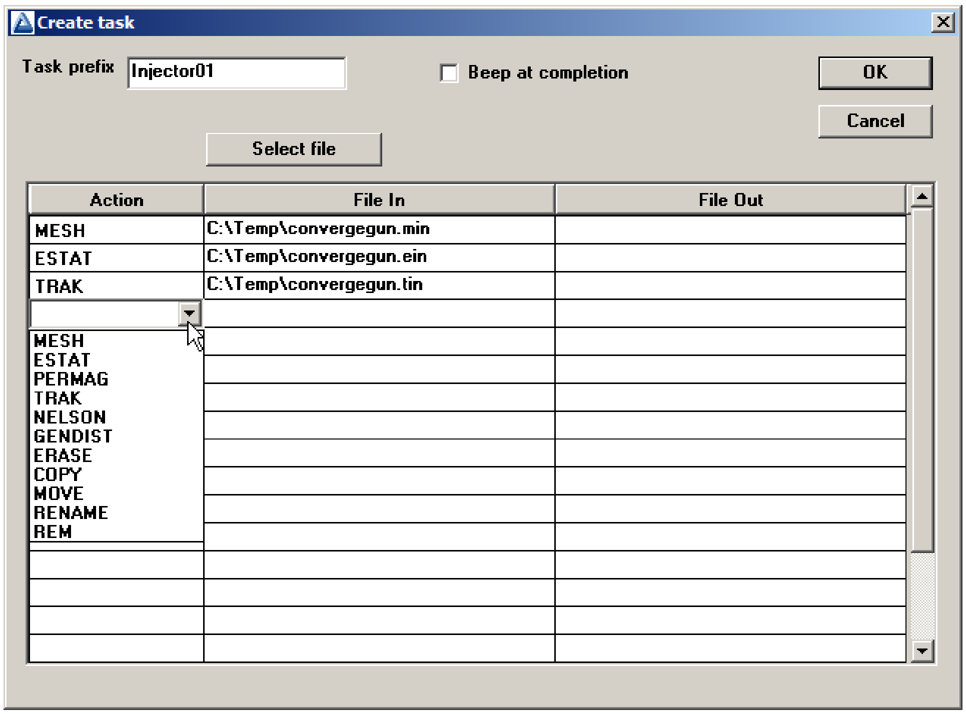
Table 1: **AMaze** solution and analysis programs

Program	**Function**
HiPhi	Electrostatics with dielectrics or conductors.
PhiView	Post-processor for **HiPhi**.
Magnum	Magnetostatics with coils, iron and permanent magnets.
MagView	Post-processor for **Magnum**.
OmniTrak	Charged-particle transport and gun design. Uses field input from **HiPhi** and **Magnum**.
Aether	Time-domain and frequency-domain electromagnetic radiation, pulsed magnets.
Aerial	Post-processor for **Aether**.
MagWinder	Drive coil definition for **Magnum** and **Aether**
HeatWave	Static and dynamic thermal transport in solid materials with temperature-dependent properties. Option to accept input power distributions from **RFE3**.
HWV	Post-processor for **HeatWave**.
RFE3	RF electric fields in the non-radiative limit for RF heating and biomedical applications.
RFE3View	Post-processor for **RFE3**.
Probe	Utility to plot history records from **HeatWave**.
GenDist	Utility to plot and to analysis particle distributions from **OmniTrak**.

It is a good practice to collect input and output files for related **AMaze** solutions in a specific data directory. Use the *Data Folder* button to set the current location. Programs that are opened after the change will read and write to the directory. (Note that the setting does not affect previously-opened programs.) The *Information* area at the bottom of the dialog shows: 1) the current data directory, 2) the last operation performed and 3) the program directory.

You will probably use additional software tools working with the **AMaze** programs and other technical programs. You can set the program to launch your favorite utility programs. To define a text editor, press the *Set tools* button in the *Control* section. In the following dialog, choose the *Editor* option. Then, move to the appropriate directory and select the program. When you exit the procedure, the *Editor* button in the *Tools* section becomes active. You can use the *Set tools* command to define two other utilities. For the file manager option, select a program like **Windows Explorer**. Use the *Data analysis* button for a spreadsheet, plotting program or mathematical analysis software. The **GCon** button in the *Tools* section is preset to run `gcon.exe`, a utility to perform batch-file runs autonomously. The *Command* button opens a DOS window if you want to run programs from the command prompt or under the control of your own batch files.

Table 1 summarizes the functions of programs included as buttons in the *Solution*, *Analysis* and *Utilities* sections.



Task prefix: Injector01

☐ Beep at completion

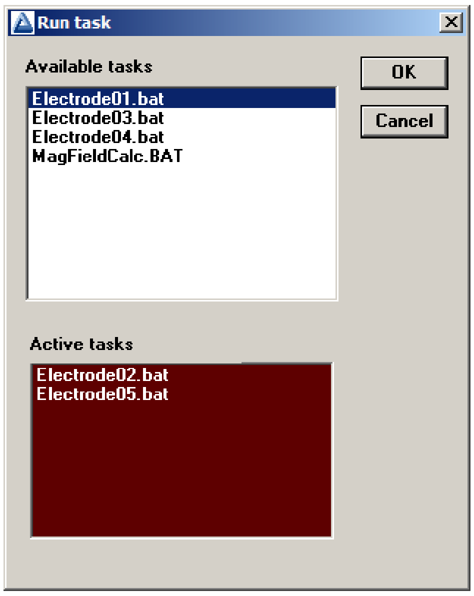
Action	File In	File Out
MESH	C:\Temp\convergegun.min	
ESTAT	C:\Temp\convergegun.ein	
TRAK	C:\Temp\convergegun.tin	

Figure 2: Create task dialog

1.3 Creating and running tasks

AMaze programs are optimized for the latest generation of multi-core or multi-processor PCs. The Professional solution programs (**HiPhi**, **Magnum**, **Aether**,) feature parallel operation. A second feature in both the Basic and Professional programs is the capability to run multiple independent calculations simultaneously. All solution programs can run in the background if launched from a Windows batch file. Background operation is automatic and faster than running in a window. The *Create task* and *Run task* commands in **AMaze** make it easy to use batch files. With the commands you can 1) quickly define multi-step calculations (*tasks*) in an interactive dialog, 2) launch simultaneous tasks in the background and 3) find out which tasks are running.

The *Create task* button calls up the dialog of Figure 2. Supply a file prefix `FPREFIX` that indicates the function of the task. The task information will be stored in a DOS batch file `FPREFIX.BAT` created in the current **AMaze** data directory. Commands in the file are compatible with all recent Windows versions including Windows 7. Each row represents an operation (batch file command). The first column defines the action. Clicking on a cell brings up a menu that includes all **AMaze** programs capable of background operation that are installed on the users computer. In addition, several relatively safe DOS commands are included (`ERASE`, `COPY`, `MOVE`, `RENAME` and `REM`). All commands operate on a file (*FileIn* column). The DOS commands `COPY`, `MOVE` and `RENAME` require a second file name (*FileOut* column). You can type file names in the cells. By default, the files are in the **AMaze** working directory, but you can include path information if the files are in other directories. Alternatively, you can click in a cell and then pick the *Select file* command to use the standard Windows dialog for choosing files anywhere on the computer. For the **AMaze** programs, the dialog displays only files with appropriate



Available tasks:

Electrode01.bat
Electrode03.bat
Electrode04.bat
MagFieldCalc.BAT

Active tasks:

Electrode02.bat
Electrode05.bat

Figure 3: Run task dialog

suffixes (*e.g.*, `*.HIN` for **HiPhi**).

Click the *OK* button when the sequence is complete to create the batch file. Here is an example:

```
REM AMaze batch file, Field Precision
START /B /WAIT C:\fieldp\amaze\metamesh64.exe C:\Temp\convergegun
START /B /WAIT C:\fieldp\amaze\hiphi64.exe C:\Temp\convergegun
START /B /WAIT C:\fieldp\amaze\omnitrak64.exe C:\Temp\convergegun
ERASE *.?ls
START /B /WAIT C:\fieldp\amaze\notify.exe
IF EXIST Electrode01.ACTIVE ERASE Electrode01.ACTIVE
```

The operations listed perform a complete **OmniTrak** calculation in the background and then erase all listing files. The example has some notable features:

- The operations are performed sequentially because data from one operation may be used in the next. To run calculations in parallel, define and run multiple tasks.
- You can modify the file with an editor if you are familiar with DOS commands.
- The DOS commands recognize the standard wildcard conventions (* for any character grouping, ? for any character).
- The programs adds the command `notify.exe` to the task sequence if *Audio alarm* was checked. In this case, the computer beeps when a task is completed.
- The final command to erase a file `FPREFIX.ACTIVE` is added to all batch files. The presence of the file indicates that the task is running.

Click the *Run task* button when you have created tasks or moved predefined task files to the data directory. The dialog (Figure 3) organizes tasks into two groups: ones that are available to run and ones that are currently running (*i.e.*, `FPREFIX.ACTIVE` has been detected). To launch a task, choose one from the top list and click *OK*. The program creates a file `FPREFIX.ACTIVE` and runs the batch file. The program sequence runs silently in the background. In the meantime, you can prepare other inputs or run other tasks.

1.4 Learning Geometer and MetaMesh

Geometer and **MetaMesh** include comprehensive sets of tools to help you represent structures in three-dimensional space. At first, the volume of information in this manual may appear overwhelming. The purpose of this section is to give you an overview and to point out critical material. You should be ready to start your own meshes after spending a few hours understanding essential concepts.

The following list summarizes the content of chapters that follow in this manual and separates them into three categories: *Essential*, *Reference* and *Advanced*. You should read the *Essential* chapters before attempting your own solutions. Chapters marked *Reference* contain information that can be skimmed and then studied in detail when you need a particular feature. *Advanced* chapters discuss specialized techniques for fine-tuning the mesh-generation process.

- **Chapter 2. MetaMesh walkthrough example** [*Essential*]. Working through a solution with a prepared script will give you insights into the function of **MetaMesh** and the sequence of operations to convert the geometric information into a conformal mesh.
- **Chapter 3. 3D meshes - concepts and terminology** [*Essential*]. The most important chapter in the manual. It contains non-mathematical descriptions of the concepts of mesh generation and the specific procedures in **MetaMesh**.
- **Chapter 4. Geometer - visualizing 3D structures** [*Essential*]. The **Geometer** program makes it easy to prepare **MetaMesh** input scripts. You can quickly determine part placements and visualize assemblies in a graphical environment. The walkthrough example of this chapter illustrates many program features and the basic operations to define a solution geometry.
- **Chapter 5. Geometer file operations** [*Reference*]. Menu commands to load scripts for viewing or editing and to save geometric information as a script.
- **Chapter 6. Geometer viewing options** [*Reference]*. Menu commands to create orthographic or perspective plots and to change the appearance of parts.
- **Chapter 7. Geometer - adding and editing parts** [*Essential*]. How to add simple geometric parts (boxes, spheres, cones, ...) to the solution space.
- **Chapter 7. Geometer/MetaMesh - standard parts bin** [*Reference*]. A list of the basic solid and planar models available for building assemblies.
- **Chapter 9. Geometer – turnings and extrusions** [*Essential*]. Turnings and extrusions are versatile models that greatly extend your ability to create complex shapes. A

turning is created by rotating an arbitrary cross section (*outline*) around an axis. An extrusion is formed by extending the outline a linear distance. This chapter summarizes the conventions for representing outlines.

- **Chapter 10. Geometer – 3D CAD program import** [*Advanced*]. How to create STL files with three-dimensional CAD programs and incorporate the models into assemblies.
- **Chapter 11. Geometer - outline editor** [*Reference*]. How to use the CAD capabilities in **Geometer** to create outlines for extrusions and turnings.
- **Chapter 12. Geometer – transitions and the neon model** [*Advanced*]. The chapter describes solid models for specialized applications. A *transition* changes smoothly from one arbitrary cross section to another over a specified linear distance. The *neon model* is a circular tube that can follow any path in three-dimensional space.
- **Chapter 13. Geometer - defining the foundation mesh** [*Reference*]. How to divide the solution space into a *foundation mesh*, a set of box elements that will be used in **MetaMesh** as the basis of the conformal mesh.
- **Chapter 14. MetaMesh - script organization and conventions** [*Essential*]. The **MetaMesh** script is a text file that describes the geometry of your application. You must work directly with the script to specify surfaces in the solution space that should be conformalized.
- **Chapter 15. MetaMesh script - global commands** [*Reference*]. Using an editor to specify the foundation mesh, to change region names and to set program controls.
- **Chapter 16. MetaMesh script - defining parts** [*Reference*]. How to make entries directly in the script to to fabricate, to orient and to position parts.
- **Chapter 17. MetaMesh script directives and block operations** [*Advanced*]. Your work will be more efficient if you create a library of standard parts and assemblies. You can paste a set of parts into the script and use the block operations to rotate and to move them as a group.
- **Chapter 17. Advanced shapes in MetaMesh scripts** [*Reference*]. Direct script editing to specify STL models from CAD programs, turnings, extrusions, transitions and neon objects.
- **Chapter 19. Modeling mathematically-generated surfaces** [*Advanced*]. **MetaMesh** can read a file giving elevation data $z(x, y)$ and create a part with a smooth surface that follows the mathematic prescription.
- **Chapter 20. Creating meshes from images** [*Advanced*]. The program can read voxel information in a standard format, using point values to separate foundation mesh elements into regions. The program then smooths the resulting region boundaries. This feature allows direct conversion of MRI images or mathematically-generated volumes to conformal meshes. Mechanical parts can be combined with a voxel mesh.

- **Chapter 21. Importing parts represented by unstructured tetrahedron meshes** [*Advanced*]. Automatic conversion of shape information from unstructured tetrahedron meshes in a neutral format.

- **Chapter 22. MetaMesh - loading, editing and processing scripts [Reference]** How to run **MetaMesh** in a window or in the background with a review of commands for file operations.

- **Chapter 23. MetaMesh interactive mode - 2D plot menu** [*Reference*]. Creating plots of the completed mesh in a slice normal to one of the Cartesian axes.

- **Chapter 24. MetaMesh interactive mode - 3D plot menu** [*Reference*]. Creating three dimensional views of the completed mesh.

- **Chapter 25. Formats of the MetaMesh output files** [*Advanced*]. **MetaMesh** can record information on the completed mesh in text or binary format. The output file contains node locations and element identities. You can transfer the information to the **AMaze** solution programs or use it for your own finite-element applications.

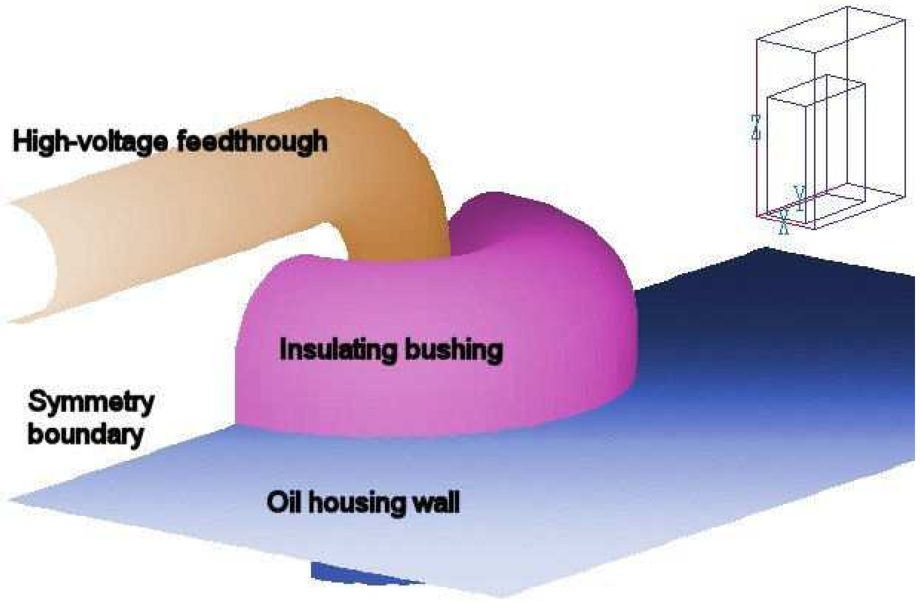
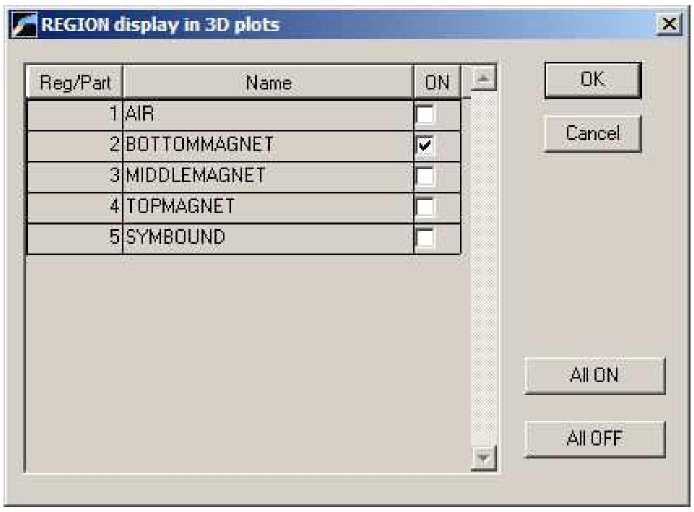


Figure 4: Geometry of the walkthrough solution

2 MetaMesh walkthrough example

To create your own meshes it is important to understand the concepts discussed in the following chapters. Before getting deeply involved with the programs, you probably want to run them on your computer and to get a sense of what they do. In this section, we will follow a solution using a prepared script. Subsequent sections describe how to generate scripts for your applications using either the interactive capabilities of **Geometer** or a text editor. Figure 4 shows the test geometry. The goal is to construct a mesh for an electrostatic solution where 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, it is sufficient to 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 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 then performs the following operations: 1) sets up a foundation mesh, 2) assigns elements of the foundation mesh to approximate the specified geometry, 3) shifts nodes so that element faces fit on material boundaries and 4) checks 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`. You can view the information later with an editor. Mesh generation should take a few seconds. At the end of the analysis, press a mouse button or any key to proceed


REGION display in 3D plots

Reg/Part	Name	ON
1	AIR	☐
2	BOTTOMMAGNET	✓
3	MIDDLEMAGNET	☐
4	TOPMAGNET	☐
5	SYMBOUND	☐

Figure 5: Region display dialog

The plot menus become active when a mesh has been processed successfully. 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. 5. Activate Regions 3 and 4 to display the insulator and high-voltage rod. You should see a plot similar to that of Fig. 6. It will be necessary to rotate and to shift the plot to achieve the same view. You can control the view by moving the mouse cursor into the plot area. To zoom in, move the cursor to the center of the plot and briefly hold down the left button. The orange box in the orientation area (top-right) shows the narrowing view. The main plot is updated when you release the mouse button. To walk around the object, move the cursor the sides, top or bottom of the plot and use the left button. To shift the view position, use the right button. (**Note**: The plot routines in **MetaMesh** show actual mesh facets. The plot requires substantially more work than the approximate surfaces created in **Geometer**, so the plot is updated only when you complete a viewpoint operation.)

At this point you can experiment with the three-dimensional plot capabilities. For example, pick the *Plot type* command in the *Plot control* menu and check 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.

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 *View/Ortho* menu click on *Normal plane* 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. 7. 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 23 gives a complete description of the slice plot algorithms and their

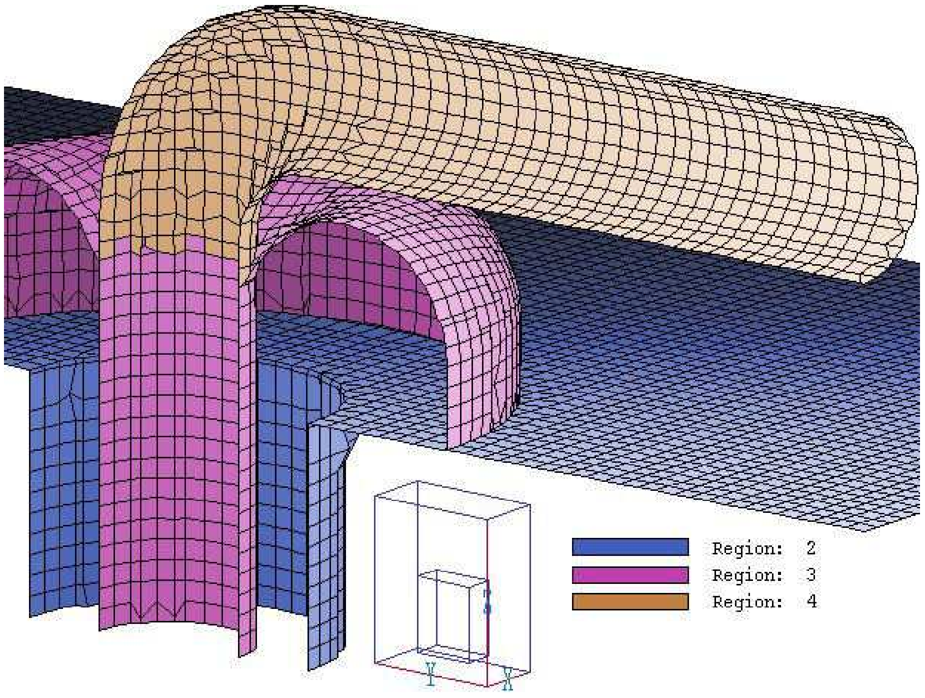


Figure 6: 3D view of region surfaces in the completed mesh

limitations. The *Hardcopy* command sends plot information to the Windows default printer.

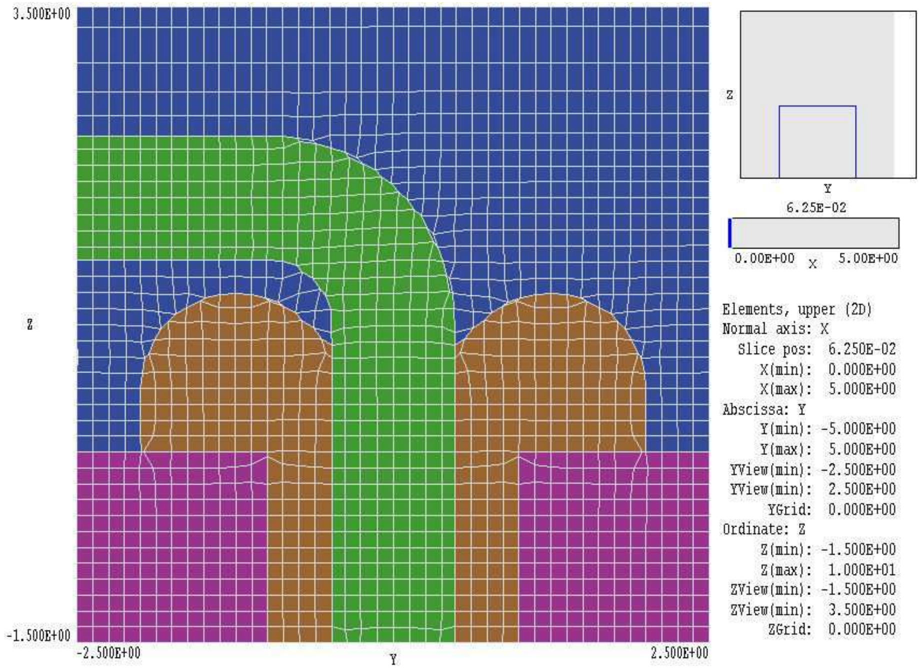
The plot of Fig. 7 illustrates the remolding of the foundation mesh boxes by the fitting process. The region where the high-voltage rod enters the insulator (center of plot) is worth noting. The insulator profile does not exactly follow the theoretical shape because the element size was not small enough to resolve the sharp corner. The example illustrates two useful characteristics 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 $z > 4.0$ was included to approximate a free-space boundary condition. Figure 8 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 $V_0 = 120$ kV. The equipotential surface has $\phi = 5$ kV. The color coding shows the magnitude of the electric field.

At this point you may wonder what type of information was required to specify the mesh geometry. To conclude the walkthrough we shall inspect the script contents. 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. we shall consider some features of the script.

Optional commands of the form:

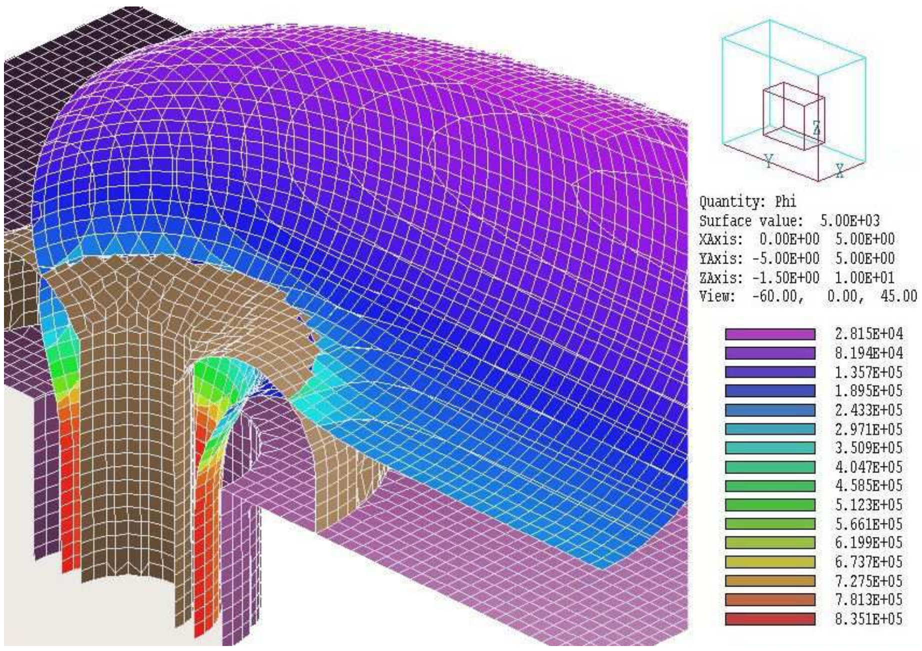
```
RegName 1 TransformerOil
```

are helpful to organize the mesh construction session. Note that the names appear in the dialog of Fig. 5. The following statements define the foundation mesh, the initial division of the solution space into box elements:



```
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 7: View of the mesh in a slice normal to the x axis



```
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 8: Electrostatic solution using the HiPhi program. Surface of $\phi = 5$ kV color-coded by $|\mathbf{E}|$.

```
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 frame and then shifted -0.75" in z to its proper position in the assembly. 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 the letters A and L are a set of line and arc vectors that outline the insulator shape in r-z space (Fig. 7). The outline is rotated about the z axis of the reference frame from 0.0° to 360.0° to form a solid. **MetaMesh** performs automatical clipping and ignores the portion of the part outside the solution volume ($x < 0.0$). 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. For situations like this with multiple rotations and translations, the interactive visualization capabilities of **Geometer** can be invaluable.

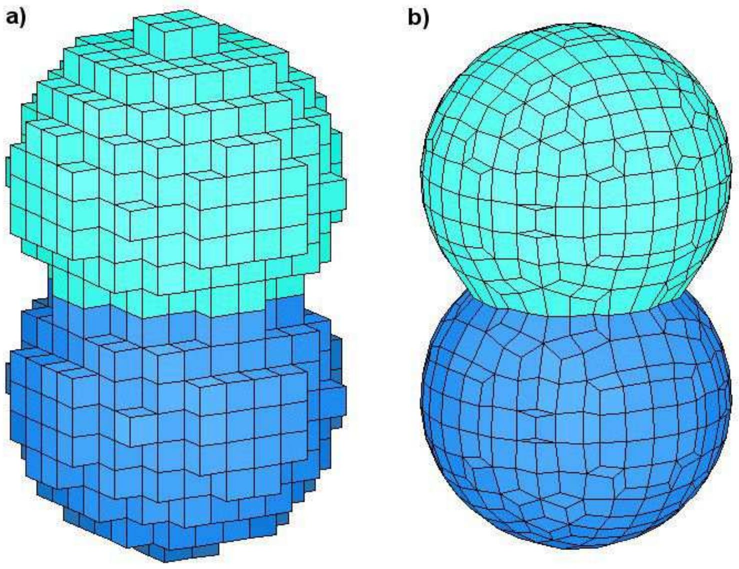


Figure 9: Representation of adjoining spheres with *a*) a regular mesh and *b*) a conformal mesh.

3 3D meshes - concepts and terminology

3.1 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.

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. 9*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. 9*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 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 may be long.

In contrast, the elements in a conformal mesh (Fig. 9*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 most tasks without your intervention.

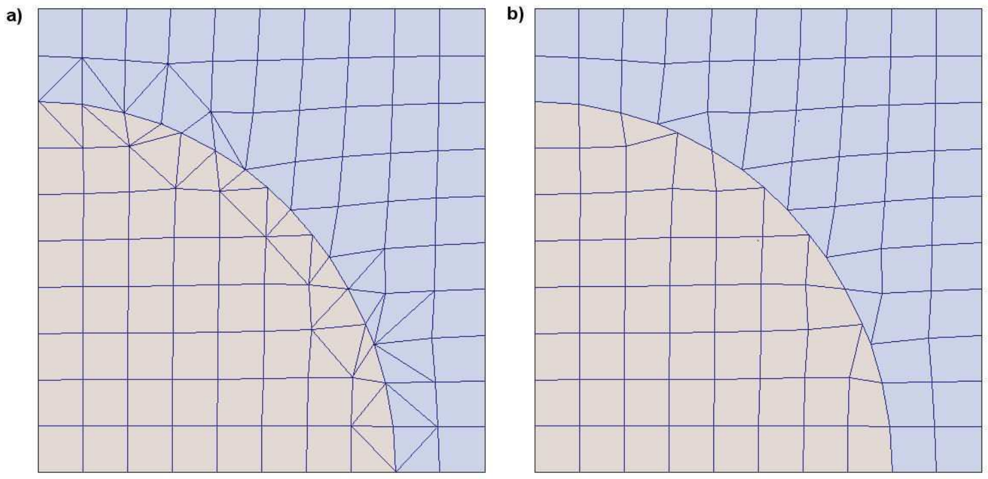


Figure 10: Two-dimensional representation of a curved surface with (a) an unstructured mesh and (b) a structured mesh.

Conformal meshes divide into two categories: *structured* and *unstructured*. Figure 10 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 nodes are close together) Four elements surround each node and each node connects to four neighbors.

With the exception of **MetaMesh**, commercially-available 3D conformal mesh generators create unstructured meshes. The reason is that it is quite difficult to fit an arbitrary three-dimensional geometry while preserving strict logical order. It is easier to patch elements in where they are needed. The unstructured approach does have one advantage. Because a sub-volume can be divided into any 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 variable interconnections between elements adds to the complexity of the solution programs and increases 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 finite-element coefficients in hexahedron elements must be determined by three-dimensional numerical integration.

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 11 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

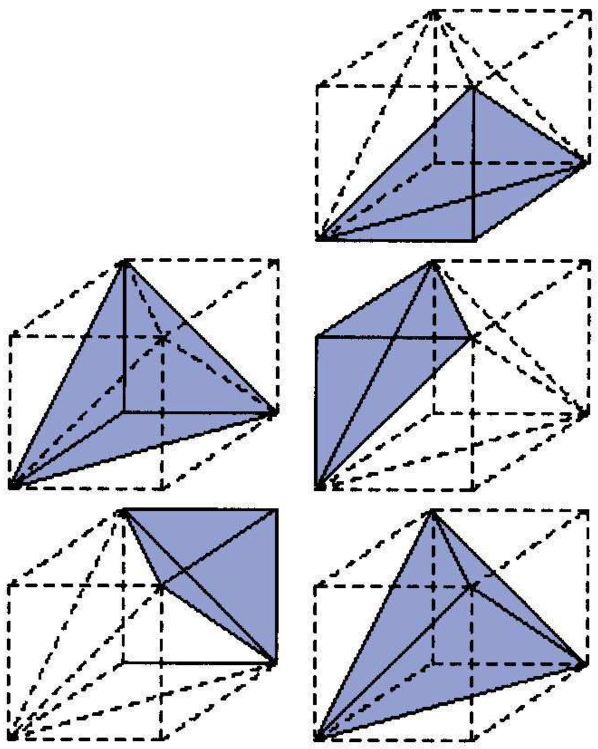


Figure 11: Filling a box with five dissimilar tetrahedrons.

use hexahedron elements in **AMaze** 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 integrals (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 more accuracy than those derived from tetrahedrons with linear shape functions.

3.2 Metamesh strategies

MetaMesh addresses the challenge of creating structured conformal meshes by proceeding in ordered steps to the final refined product. We will review the procedure in this 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 flexible and will usually give a useful starting mesh even with relatively large elements. You can fine-tune the mesh later. In the example, we pick a solution

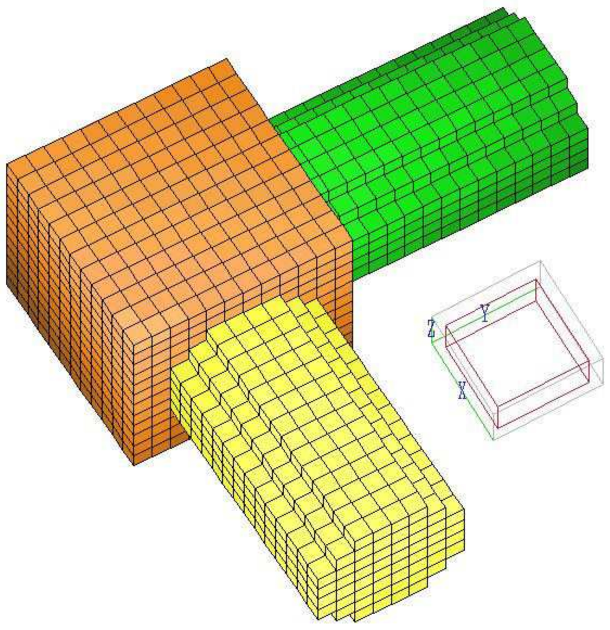


Figure 12: Example `EBLOW_DEMO` at the volume-fitting stage.

volume in the range

$$0.0 \leq x \leq 5.0, \quad 0.0 \leq y \leq 5.0, \quad 0.0 \leq z \leq 2.0,$$

and divide it into cubes with sides of length 0.125.

Stage 2 - volume fitting. Based on geometric specifications in the 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 within the geometric definition of the part. Figure 12 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 a good choice of mesh resolution, 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.

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 example 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 model. 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 13 shows the state of the mesh after surface fitting.

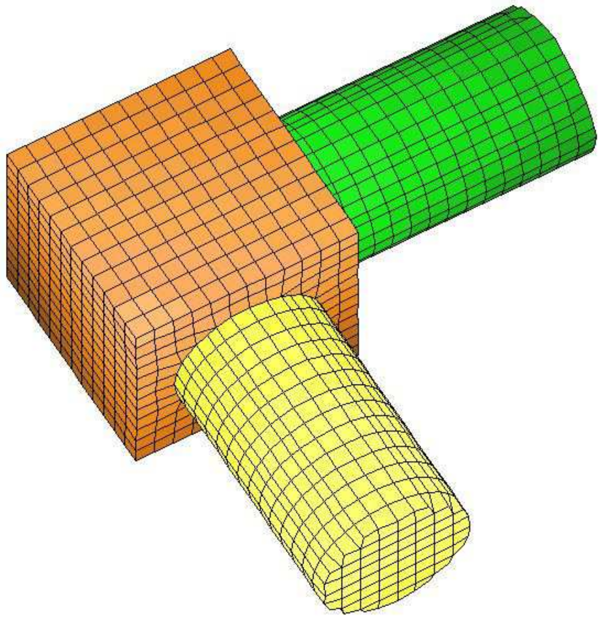


Figure 13: Example EBLOW_DEMO after surface fitting.

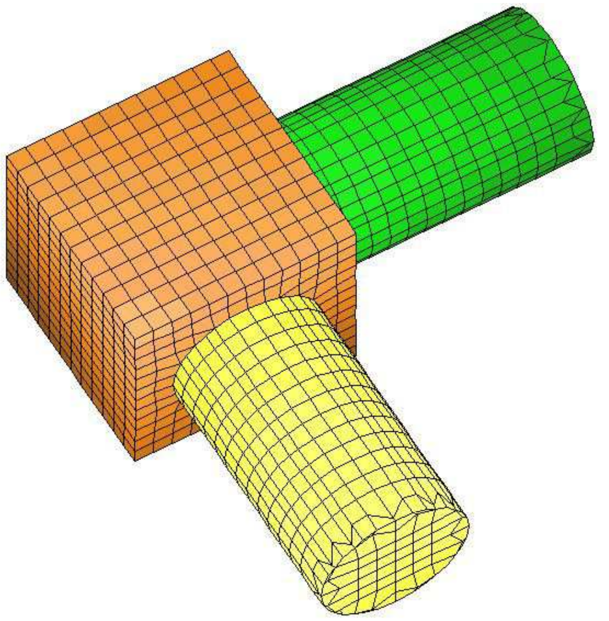


Figure 14: Example EBLOW_DEMO after surface and edge fitting.

Stage 4 - edge fitting. Surface fitting involves moving nodes to the closest geometric surface of the part model. As shown in Fig. 13, 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 rare occasions where a precise edge is required. When the option *Edge* appears in the *Surface* statement, **MetaMesh** generates a list of nodes adjacent to the edges and then shifts them incrementally to reduce the bevel effect. Figure 14 shows the state of the mesh with edge fitting on both cylindrical parts.

Stage 5 - node fitting. **MetaMesh** can also set the region identity and positions of individual nodes after processing volumetric parts. 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 surface and (optionally) volume integrals over regions performed during the tests to indicate fitting accuracy.

The following sections summarize some details about the stages that you need to know in order to use **Geometer**. Chapters14-20 cover script commands to control the mesh generation process in detail.

3.3 Foundation mesh quantities

The first stage in the mesh-generation process is to construct a foundation mesh. 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 15 shows a slice of a mesh normal to the z−axis.

The mesh is initially divided into box 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

$$N_{max} = I_{max} \times J_{max} \times K_{max}.$$

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 with labels

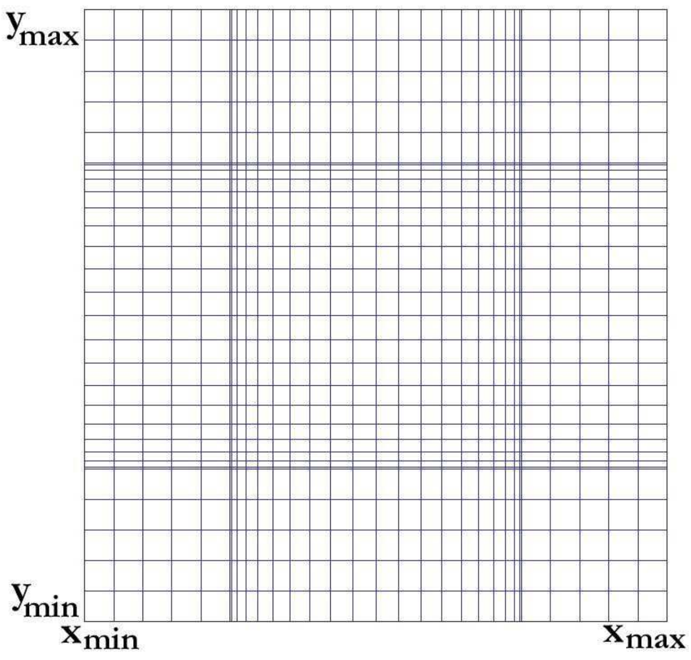


Figure 15: Foundation mesh projected in the z plane

```
[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. 16, element $[I, J, K]$ of the foundation mesh extends from $x(I\text{-}1)$ to $x(I)$ along x, from $y(J\text{-}1)$ to $y(J)$ along y and from $z(K\text{-}1)$ to $z(K)$ along z. The sizes of elements of the foundation mesh may vary along each axis. Size variations can be used to improve resolution in critical areas and to provide a better match to the boundaries of objects.

3.4 Assigning element and node region numbers

The second stage of the mesh-generation process is volume fitting. Here **MetaMesh** assigns region and part numbers to elements and nodes of the foundation mesh based on the specified

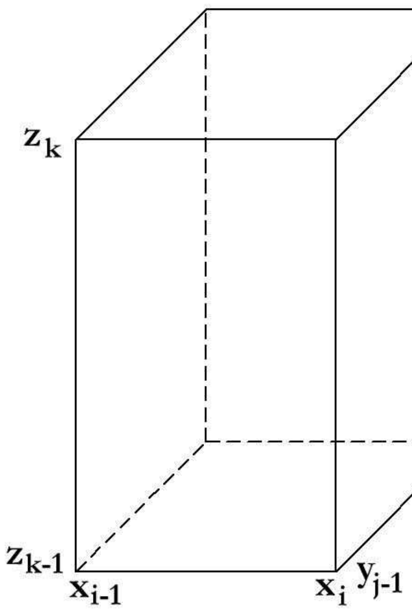


Figure 16: Boundaries of the foundation element $[I, J, K]$.

geometry. The region number, an integer in the range 1 to 250, 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, assigning part numbers 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 boundaries of the part model. If the element is inside, **MetaMesh** assigns the part number and associated region number to the element. The program also assigns the current region and part numbers to the eight nodes connected to the element. ***It is important to note that region and part numbers over-write any previous definitions***.

3.5 Fitting and coating surfaces

The third and fourth stages of mesh-generation (surface and edge fitting) are carried out only for parts that contain one or more *Surface* commands. The commands have the following actions.

SURFACE REGION RegNo
SURFACE REGION 5
SURFACE(REGION) = AirVolume

After volume fitting, a solid part consists of a set of box elements. Some of the element facets lie on the boundary of the part, as in Figure 12. In the figure, some of the facets of each cylinder border on the surrounding air region and some border on the cube. In response to the `SURFACE REGION REGNO` command, **MetaMesh** collects all facets on the part boundary that border on elements with region number *RegNo* and adjusts the associated nodes so the facet lies close to the ideal model boundary of the part. In the case of Fig. 12, we fit to the air region. The result is shown in Fig. 13.

SURFACE PART PartNo
SURFACE PART 12 SURFACE(PART) = Pipe01
This command is similar to the *Surface region* command. The difference is that **MetaMesh** picks boundary facets that border on elements with a specified *part* number. An example in the next section shows why this distinction may be important.

The identity of nodes is significant in the **AMaze** solution programs. For example, suppose that an electrode and dielectric in an electrostatic simulation share a common boundary. To obtain the correct solution to Poisson's equation, the common nodes must have the fixed-potential condition of the electrode. Usually, we can ensure that node assignment is correct by placing all dielectric parts at the beginning of the script and all fixed-potential parts at the end. In this case, the over-write principle ensures that the shared nodes will have the fixed-potential condition. Sometimes we cannot use this ordering. For example, suppose we wanted to drill a hole through an electrode. In this case, the part representing the hole must appear after the electrode. For this situation, **MetaMesh** has a special command to correct node identities.

COAT RegNo RegNew
COAT 3 5
COAT AirVolume UpperElect
Search the boundary of the part and find all facets that border elements with region number *RegNo*. Then, change the region number of the associated nodes to *RegNew*.

3.6 Logic of mesh generation process

A quick way to illustrate the logic of the **MetaMesh** script is to consider a specific example. Figure 17 shows the geometry defined by the file `MESH_LOGIC.MIN` listed in Table 2. The assembly contains an offset spherical electrode inside a shaped vacuum chamber. We have included a hole through the electrode to illustrate how to drill parts.

Although the inside wall of the grounded vacuum chamber is cylindrical, the solution volume must be a box. The first part in the script is a box representing the vacuum chamber that fills the entire solution volume:

```
Part 1
  Type Box
  Name GroundedWall
  Region 1
  Fab 10.50 10.50 100.0
End
```

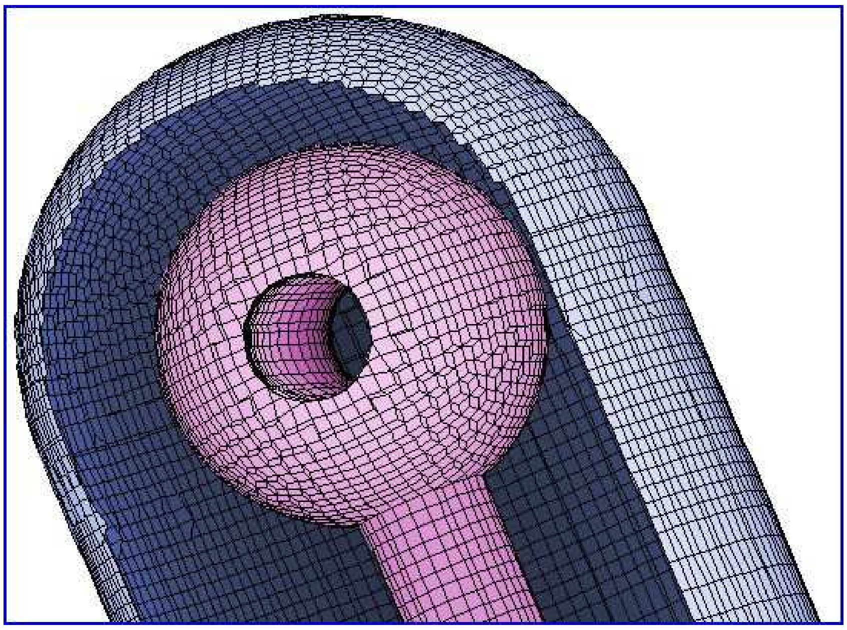


Figure 17: Cutaway view of the geometry for the simulation MESH_LOGIC.MIN. Blue facets represent the inner wall of the vacuum chamber. Violet facets show the surface of the electrode.

In processing the part, **MetaMesh** sets $RegNo = 1$ and $PartNo = 1$ for all elements and nodes of the foundation mesh. In **HiPhi** the region will be associated with the condition $\phi = 0.0$.

The next part is the inner volume of the shaped vacuum chamber bored into the metal block. We use the *Turning* model to create a cylindrical chamber with a spherical dome. The vectors in the TYPE TURNING structure outline the shape shown in Fig. 18.

```
Part 2
  Type Turning
    L 20.00 0.00 20.00  5.00
    L 20.00  5.00  5.00  5.00 S
    A  5.00  5.00  0.00  0.00  5.00  0.00 S
    L  0.00  0.00 20.00  0.00
  End
  Region 2
  Surface Region 1
  Coat 1 1
End
```

The processing operation assigns $RegNo = 2$ and $PartNo = 2$ to elements and nodes in the volume-fitting process. In the electrostatic solution, the region represents a dielectric with $\epsilon_r = 1.0$. In response to the command SURFACE REGION 1, **MetaMesh** locates all facets on the border of the dielectric that contact vacuum chamber elements and moves the associated nodes to the part surface defined by the shape of Fig. 18. The result is the smooth wall shown in Fig. 17. After the cutting operation, the nodes on the inside of the wall have the dielectric region number. In response to the command COAT 1 1, **MetaMesh** finds all facets on the border between the region and sets the node number to that of the vacuum chamber.

The next two parts combine to create the high-voltage electrode:

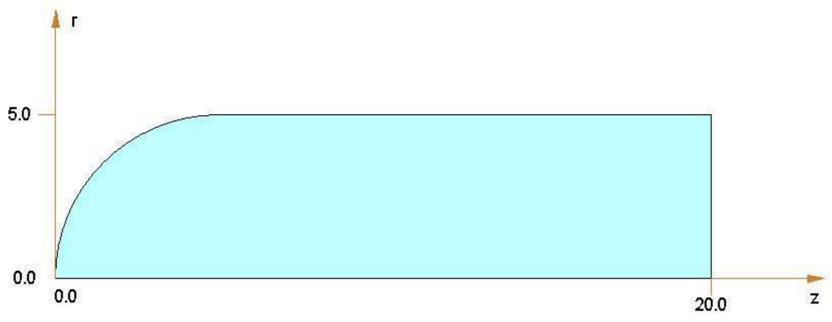


Figure 18: Vacuum chamber cross-section defined by three line vectors and an arc.

```
PART 3
  Type Cylinder
  Region 3
  Fab 1.00 15.00
  Shift 0.50 0.00 12.50
  Surface Region 2
END
PART 4
  Type Sphere
  Region 3
  Fab 3.00
  Shift 0.50 0.00 5.00
  Surface Part 2
End
```

After the parts are combined, the facets that border on vacuum are moved to the model boundaries. We leave facets on the border between Part 3 and Part 4 unchanged because they are internal to the region.

The final part is the most interesting. We use it to drill a hole through a diameter of the spherical 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 Region 3 1.00
  Coat 3 3
End
```

Elements of the part have the region number of vacuum, $RegNo = 2$. We choose the length of the cylinder to be somewhat larger than the diameter of the sphere so that it extends completely through. Fitting is performed on the circular wall of the cylinder for facets that contact elements of the electrode. For these facets, the *Coat* command changes node assignments on the shared boundary with the electrode. Figure 19 shows a detailed view of elements in the region around the hole.

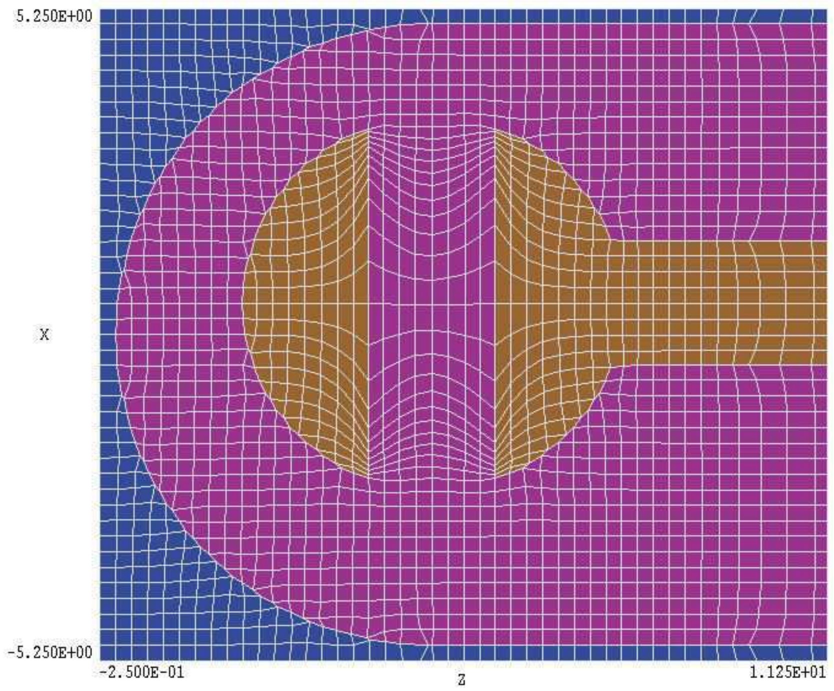


Figure 19: Orthographic view of a portion of the finished mesh in the plane $y = 0.0$.

In review, here are some important points to note about the logic of this part:

- Because only region numbers are recorded in the **MetaMesh** output file, the vacuum chamber and the hole through the electrode are treated as the same entity in **HiPhi**.
- According to the over-write principle, the hole must appear *after* the electrode in the script. The *COAT* command was required to correct the identity of shared nodes for a valid electrostatic solution.
- We made the cylinder larger that the sphere so that **MetaMesh** would not waste effort fitting the end-faces. The shapes of facets on the cylinder end-faces are not important because they lie between elements with the same region number.
- Consider the effect of replacing `SURFACE PART 2` with `SURFACE REGION 2` in the definition of the spherical electrode ($PartNo = 4$). In this case, **MetaMesh** would collect all facets that border on vacuum (including those *inside* the cylindrical hole) and try to fit them to the *outside* of the sphere. The result would be highly distorted elements near the hole.

3.7 Mesh-generation guidelines

The example in the previous section illustrates that you can create sophisticated objects in **MetaMesh** with moderate effort. 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:

Table 2: Part sections in the MESH_LOGIC script

```
PART 1
  Type Box
  Region 1
  Fab 10.50 10.50 100.0
END
PART 2
  Type Turning
    L 20.00 -5.00 20.00  5.00 S
    L 20.00  5.00  5.00  5.00 S
    A  5.00  5.00  0.00  0.00  5.00  0.00 S
    A  0.00  0.00  5.00 -5.00  5.00  0.00 S
    L  5.00 -5.00 20.00 -5.00 S
  End
  Region 2
  Fab 0.0  360.0
  Surface Region 1
  Coat 1 1
END
PART 3
  Type Cylinder
  Region 3
  Fab 1.00 15.00
  Shift 0.50 0.00 12.50
  Surface Region 2
END
PART 4
  Type Sphere
  Region 3
  Fab 3.00
  Shift 0.50 0.00 5.00
  Surface Part 2
END
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 Region 3
  Coat 3  3
END
ENDFILE
```

- Build the script step-by-step. Add and test a new part before proceeding to the next.
- Use comment lines to organize and to document 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.
- If you observe severe mesh distortions, there may be a problem with the order of parts or the logic of how *Surface* operations are applied.

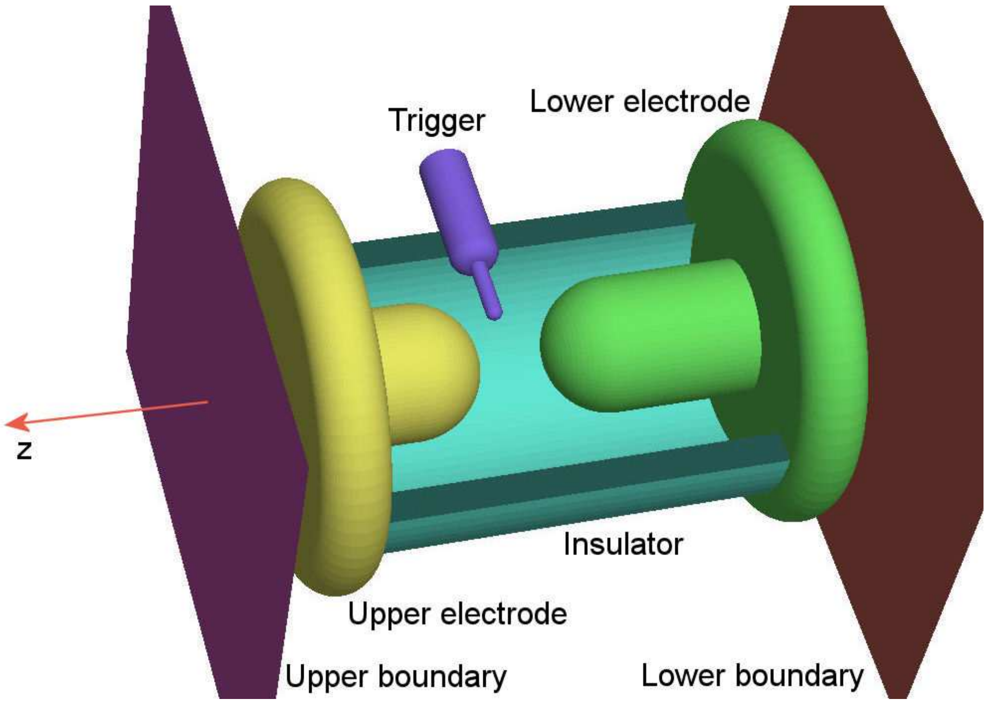


Figure 20: Geometry of the walkthrough solution

4 Geometer - visualizing 3D structures

In this chapter we shall work through an example that illustrates many of the features of **Geometer**[2]. The goal is to create a script that describes the high voltage field-distortion spark gap shown in Fig. 20. The resulting mesh could be used for an electrostatic calculation with the **HiPhi** program. The assembly consists of two electrodes immersed in sulfur-hexafluoride at high pressure. The insulator (shown in a cutaway view) is the gas housing. The trigger electrode is normally biased to a potential intermediate between that of the electrodes. When a high-voltage pulse is applied to the trigger, the resulting electric-field distortion causes an arc breakdown. To approximate free space boundary conditions, the solution volume has fixed-potential boundaries in z at the same voltage as the electrodes and default Neumann boundaries on the sides normal to x and y.

To begin, move the files `UPPERELECTRODE.OTL`, `LOWERELECTRODE.OTL` and `TRIGGER.OTL` to a working directory. These files contain outlines that define the cross-sections of the electrodes and trigger pin. If necessary, set the *Data folder* in the **AMaze** program launcher and run **Geometer**. Choose the command *File/New script* to bring up the dialog shown in Fig. 21. Type `WALKTHRU` in the prefix box. The prefix will be applied to the output script. The dimensions (in inches) give the boundaries of the solution box. We shall use the ranges $-4.5 \leq x \leq 4.5$, $-4.5 \leq y \leq 4.5$ and $-1.0 \leq z \leq 9.0$. Although you may use any

[2]**Geometer** uses OpenGL, the standard employed in many video games. Three-dimensional effects with real-time animation are possible because most of the image processing is performed on the graphics card. A low-end card may give poor images and slow regeneration

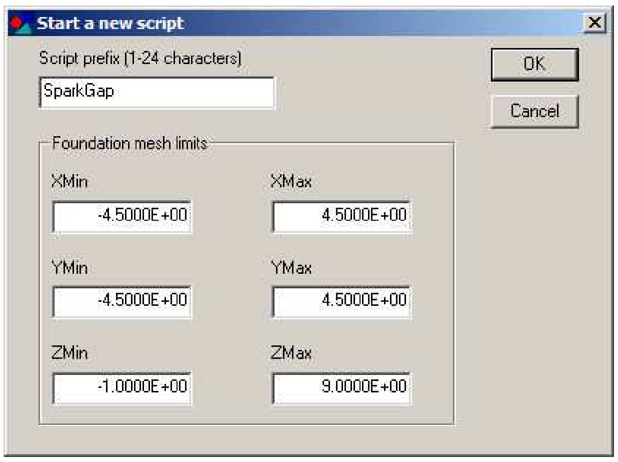


Figure 21: Dialog to begin a new script

convenient units, program accuracy is best when coordinate values are near unity. When you click *OK*, the program shows a default plot normal to the z axis with no objects.

Geometer creates a default for part 1, a box that fills the solution volume. The part is associated with region 1. Both the part and the region have the default name *SolutionVolume*. Before adding other parts, we shall define region names to organize our work. To review, *regions* correspond to different physical objects or material types in the **HiPhi** solution. Each *part* has an associated region, and a physical object may be composed of several parts with the same region number. Click the *Edit/Edit region names* command to bring up the dialog of Fig. 22. Initially, the first region is named *SolutionVolume* and the other 126 available regions have been assigned generic names. Type in the following names for regions 1-5:

1. `Air`. The first region number is applied to gas elements inside the insulator and air elements outside. There is no need to define separate regions for the electrostatic solution because gases have $\epsilon_r \cong 1.0$.

2. `Insulator`.

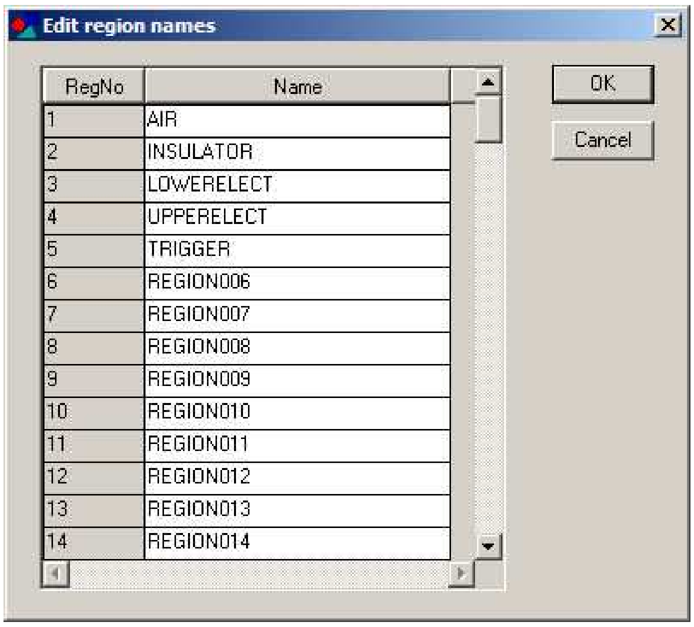
3. `LowerElect`. The fixed potential condition of this region will be applied to both the lower electrode and lower boundary.

4. `UpperElect`. The fixed potential condition of this region will be applied to both the upper electrode and upper boundary.

5. `Trigger`.

The names are case insensitive. Note that they do not contain spaces or other standard **AMaze** delimiters. Click *OK* to record the names and exit the dialog.

To begin, we shall check the parameter definitions for the first part. Click *Edit/Edit part* and choose part number 1 to bring up the dialog of Fig. 23. This dialog is one of the most important tools in **Geometer**. The *Part type* field set to *Box*. Click the arrow on the right side of the menu to see the other options (the available set of **MetaMesh** models). Several fields in the dialog appropriate to the box model are active. Assigning a descriptive name to



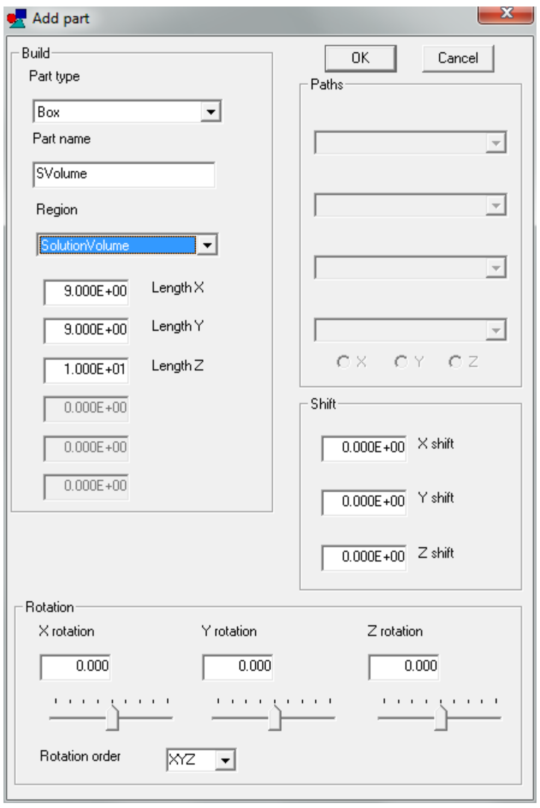
RegNo	Name
1	AIR
2	INSULATOR
3	LOWERELECT
4	UPPERELECT
5	TRIGGER
6	REGION006
7	REGION007
8	REGION008
9	REGION009
10	REGION010
11	REGION011
12	REGION012
13	REGION013
14	REGION014

Figure 22: Dialog to define region names

the part is optional but recommended. The dimensions to fabricate the part appear below the *Region* field. For the box model, the fabrication parameters are the lengths of the sides along x, y and z. Finally, The *Shift* fields contain displacements to center the box in the solution volume. You could change any of the parameters for a calculation where part 1 does not fill the solution volume. For this example, click *Cancel* to exit the dialog.

You will the box in the plot display because **Geometer** does not display part number 1 by default. The default assignment of an encompassing box would obscure the view of subsequent parts. Before proceeding, we can check that the box exists. Click *View/Displayed parts/Regions* to bring up the dialog of Fig 24. Initially, only the first part appears on the list. Activate *Visible* (but not *Solid*) and click *OK*. The default orthographic plot now includes a wireframe plot of the OpenGL display facets. To get a better view, click *View/Orthogonal perspective*. The program switches to the interactive 3D mode. Figure 25 shows the appearance of the box. Move the mouse cursor to the left side of the screen and hold down the left mouse button to move around the box. Try holding down the left mouse button at the top and bottom of the screen. Move the mouse cursor to the middle of the screen and use the left and right buttons to zoom in and out. The active screen zones are shown in red in Fig. 25. The central zone (A) is used for zooming in (left button) and out (right button). Hold down the left mouse button in zones B, C, D and E to walk around the object. Hold down the right mouse button in zones B, C, D and E to move the viewpoint to the right, upward, to the left and downward. To proceed to the next step, return to the orthographic view and uncheck the *Visible* attribute of the first part.

We shall next construct the insulator (part 2). The cylindrical part has inner radius 2.0, outer radius 2.5 and length 6.0. In the assembly the insulator extends from 1.0 to 7.0 along z. The *Turning* model is ideal for the part. To create the part, we need an *outline* (vector set) to define the cross section that will be rotated about the z-axis in the workbench frame. From the main menu, click *Outline* to enter the outline menu and then click *Outlines/New*



Part type: Box

Part name: SVolume

Region: SolutionVolume

9.000E+00 Length X

9.000E+00 Length Y

1.000E+01 Length Z

0.000E+00

0.000E+00

0.000E+00

0.000E+00 X shift

0.000E+00 Y shift

0.000E+00 Z shift

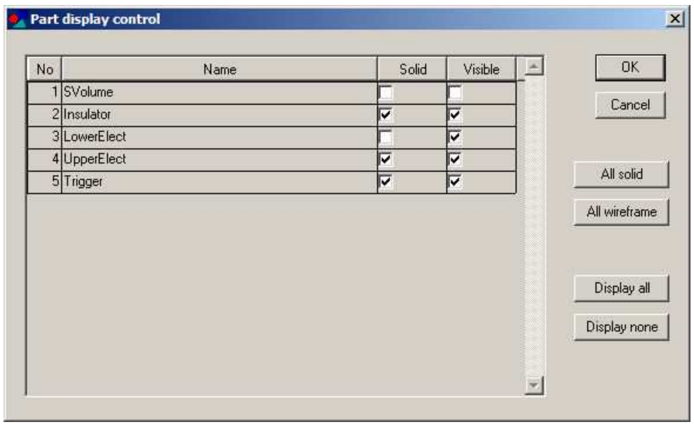
X rotation 0.000

Y rotation 0.000

Z rotation 0.000

Rotation order XYZ

Figure 23: Dialog to add a new part to the assembly



No	Name	Solid	Visible
1	SVolume	☐	☐
2	Insulator	✓	✓
3	LowerElect	☐	✓
4	UpperElect	✓	✓
5	Trigger	✓	✓

Figure 24: Dialog to change the visibility and plot style of parts and regions

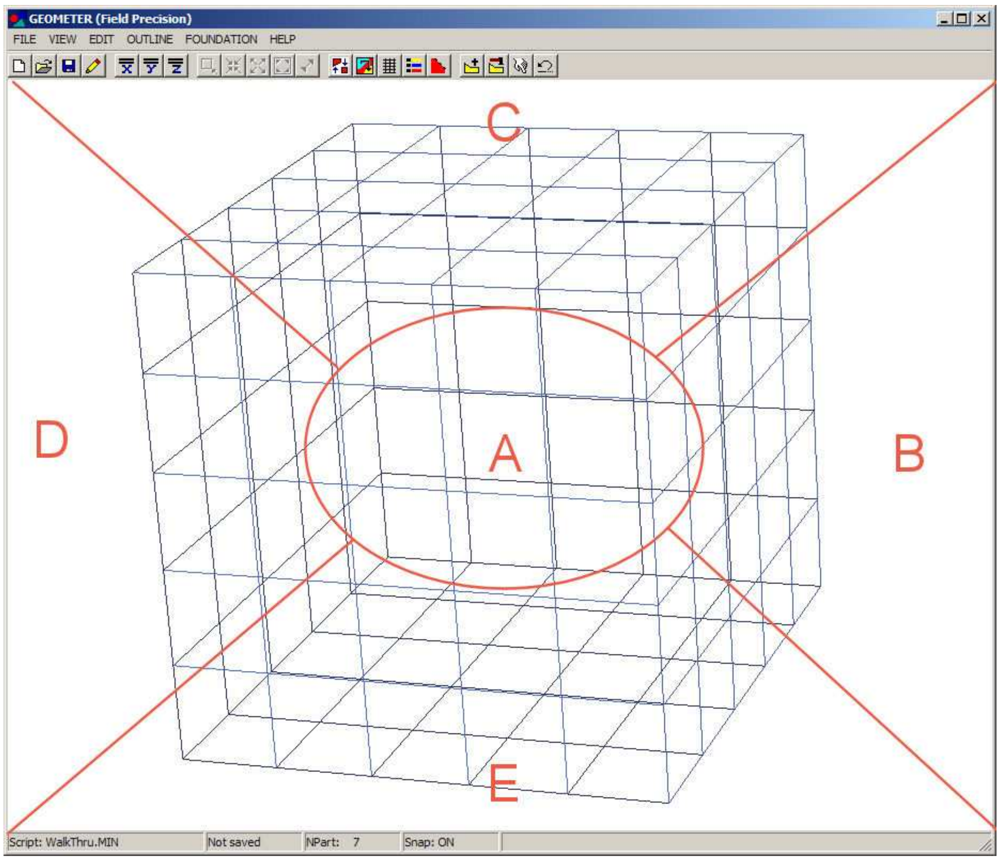

GEOMETER (Field Precision)
FILE VIEW EDIT OUTLINE FOUNDATION HELP
C
D
A
B
E
Script: WalkThru.MIN
Not saved
NPart: 7
Snap: ON

Figure 25: Perspective view of the first part (a box that fills the solution) volume showing the active zones for mouse control.

outline/Graphics. In the dialog, give the outline the name `Insulator`. The coordinates define a convenient initial viewing area in (z-r) space. They have no effect on the actual dimensions of the part. A good choice is $0.0 \leq z \leq 8.0$ and $0.0 \leq r \leq 5.0$. When you click *OK*, the program displays the drawing space. The visible grid shows intervals of 0.5. The status window at the bottom of the screen shows that the snap grid interval has been automatically set to 0.25 in both the z and r directions and that the mouse snap mode is set to *Grid*. The default choices are sufficient for the present outline. We shall now proceed to draw the rectangular cross-section of the insulator using the mouse. Click *Draw/Insert rectangle*. Notice that the status bar switches to coordinate input mode. Use the coordinates in the status bar to move the mouse to the position $x, z = 1.0, y, r = 2.0$. Note how the values change discontinuously to multiples of the snap distances. Click the left mouse button to fix one corner of the rectangle, move the mouse to the position $x, z = 7.0, y, r = 2.5$ and click the left mouse button again to complete the outline. Click *Save/Exit* to save the outline. The border returns to white to show you are back in the standard outline menu. At this point, the outline exists only in memory. Click *Outline/Save outline*. The first dialog shows a menu listing available outlines. Since there is only one outline, click *OK* to accept the default. The second dialog gives you the option to enter a name for the saved file. If you accept the default, **Geometer** creates a file `INSULATOR.OTL` with the following content:

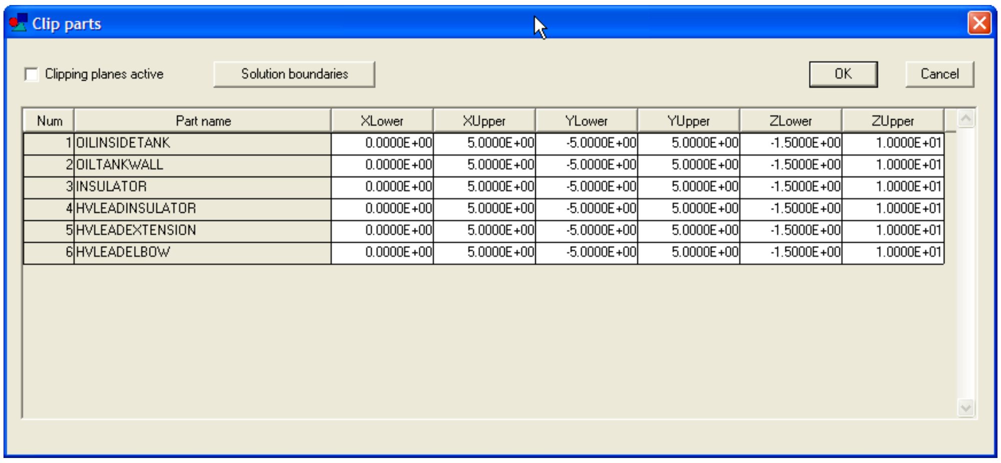
```
   L    1.0000E+00  2.0000E+00  7.0000E+00  2.0000E+00  S
   L    7.0000E+00  2.0000E+00  7.0000E+00  2.5000E+00  S
   L    7.0000E+00  2.5000E+00  1.0000E+00  2.5000E+00  S
   L    1.0000E+00  2.5000E+00  1.0000E+00  2.0000E+00  S
END
```

Each data line gives the starting and ending coordinates of line vectors that outline the rectangle. The format is compatible with both the **Mesh** and **MetaMesh** programs.

Click *Return* to exit the outline editor. We shall now construct the insulator. In the main menu, click *Edit/Add part*. From the *Part type* box choose *Turning*. For the part name use `Insulator` and choose the region *Insulator* in the region menu box. Notice that the *Outline* box on the right-hand side of the dialog is active. No change is necessary – we shall use the default choice *Insulator*. The two fabrication parameters are *AngMin* and *AngMax*. If both entries are left at the default of 0.0, the turning covers 360°. Click *OK* to exit the dialog and save the part specifications.

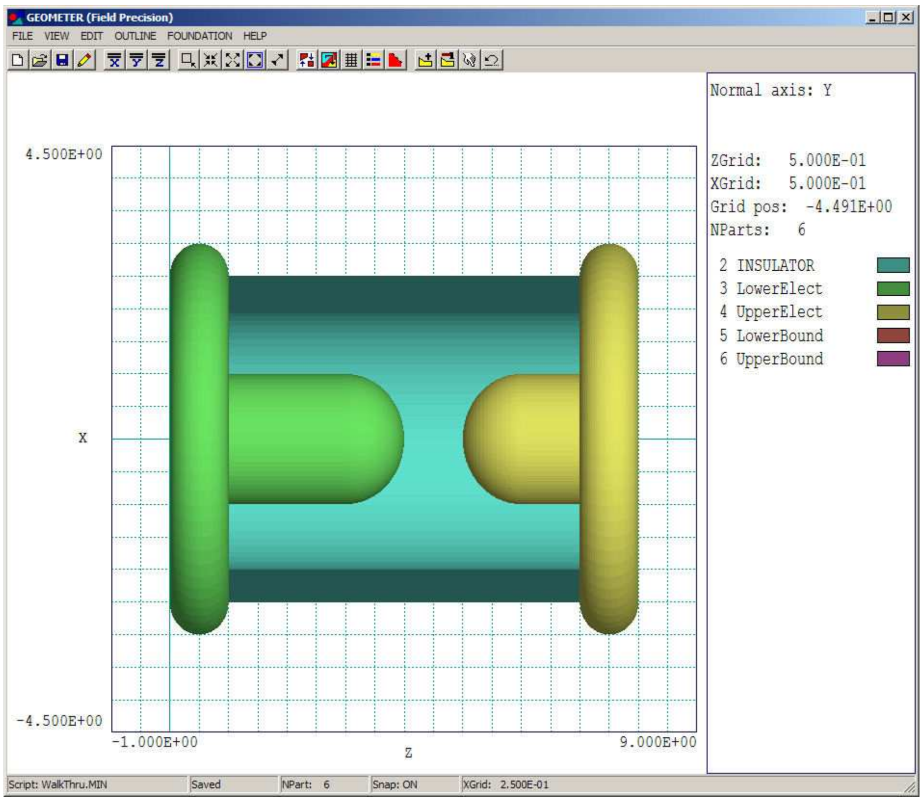
The default view along the z axis (with the viewer at $+z$) shows the end face of the insulator. To get a better three-dimensional picture, click *View/Y normal axis*. You are now looking at the insulator from the $+y$ direction. The insulator will obscure our view when we add the electrodes and trigger, so we shall change the display to a cutaway view. Click *View/Part clipping* to display the dialog shown in Fig. 26. On the *YUpper* row, change the numerical entry to $y = 0.0$. When you exit the dialog, you will be looking inside half of the insulator. Note that this setting affects only how the insulator is displayed, not how it will be recorded in the script.

We shall now add the electrodes. For the turnings We shall use more complex outlines that have already been prepared consisting of arc and line vectors. Go to the *Outline* menu and click *Outlines/Load outline*. Choose `LOWERELECTRODE.OTL` and click *OK*. The outline is loaded into the program and displayed on the screen. Follow the same procedure to load the outline `UPPERELECTRODE.OTL`. There are now three available outlines: *Insulator*, *LowerElectrode* and



Num	Part name	XLower	XUpper	YLower	YUpper	ZLower	ZUpper
1	OILINSIDETANK	0.0000E+00	5.0000E+00	-5.0000E+00	5.0000E+00	-1.5000E+00	1.0000E+01
2	OILTANKWALL	0.0000E+00	5.0000E+00	-5.0000E+00	5.0000E+00	-1.5000E+00	1.0000E+01
3	INSULATOR	0.0000E+00	5.0000E+00	-5.0000E+00	5.0000E+00	-1.5000E+00	1.0000E+01
4	HVLEADINSULATOR	0.0000E+00	5.0000E+00	-5.0000E+00	5.0000E+00	-1.5000E+00	1.0000E+01
5	HVLEADEXTENSION	0.0000E+00	5.0000E+00	-5.0000E+00	5.0000E+00	-1.5000E+00	1.0000E+01
6	HVLEADELBOW	0.0000E+00	5.0000E+00	-5.0000E+00	5.0000E+00	-1.5000E+00	1.0000E+01

Figure 26: Dialog to set clipping planes for individual parts.


GEOMETER (Field Precision)
FILE VIEW EDIT OUTLINE FOUNDATION HELP
4.500E+00
X
-4.500E+00
-1.000E+00
Z
9.000E+00

Normal axis: Y

ZGrid: 5.000E-01
XGrid: 5.000E-01
Grid pos: -4.491E+00
NParts: 6

2 INSULATOR	🟦
3 LowerElect	🟩
4 UpperElect	🟨
5 LowerBound	🟥
6 UpperBound	🟪

Script: WalkThru.MIN | Saved | NPart: 6 | Snap: ON | XGrid: 2.500E-01

Figure 27: Assembly with insulator, upper electrode and lower electrode with the insulator displayed in a cutaway view.

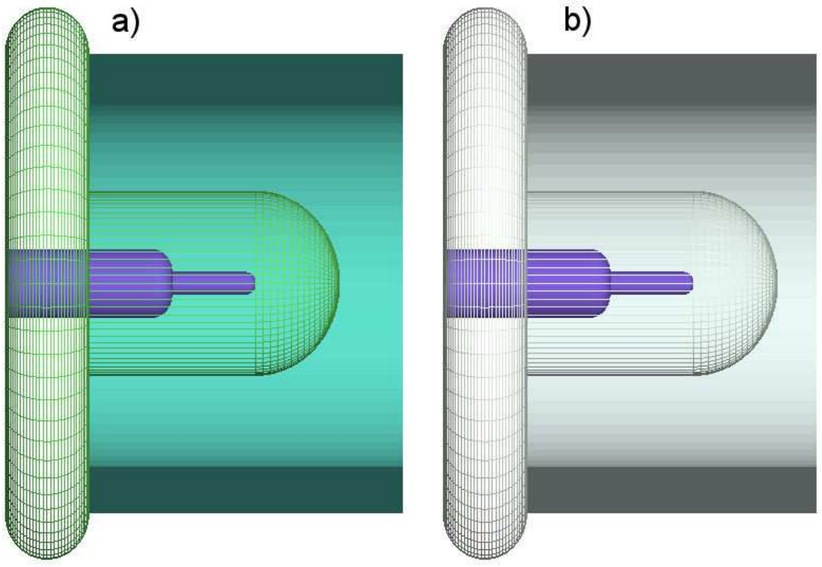
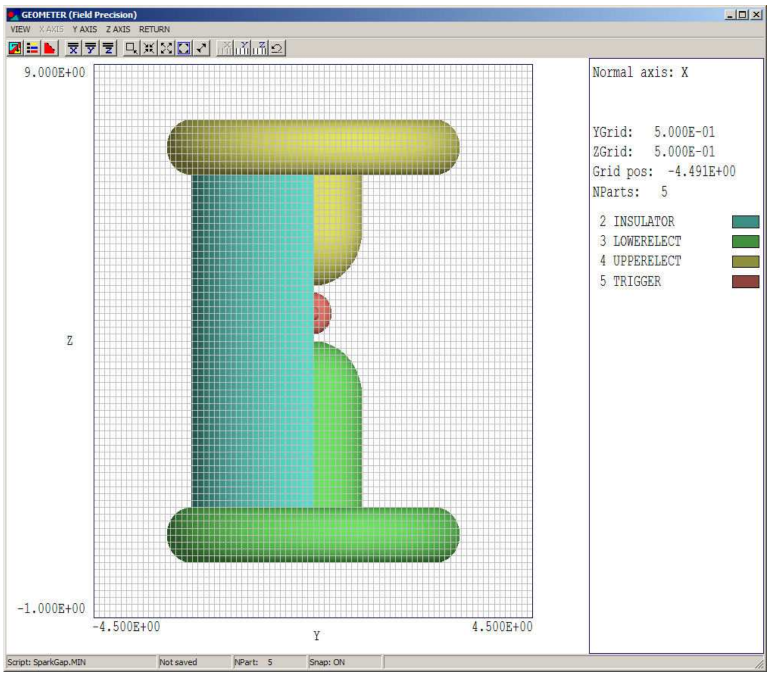


Figure 28: Adding the trigger in the default workbench position. *a*) Lower electrode displayed as a wireframe. *b*) Trigger selected.

UpperElectrode. You can use the *View/Displayed outline* command to inspect them. Return to the main menu and click *Edit/Add part*. Choose the *Turning* model, assign the part name *UpperElectrode* and choose the region *UpperElect*. In the *Outline* box, choose *UpperElectrode* and exit the dialog. Perform the same set of operations to construct the lower electrode. If all went well, the display should look like Fig. 27.

By a judicious choice of the system layout, it was not necessary to include rotations and translations for the three turnings already in the model. Clearly, this is not the case for the trigger electrode (Fig. 20). Following the procedure we used for the electrodes, load the outline `TRIGGER.OTL` and then create a new turning with part name *TRIGGER* associated with region *TRIGGER*. By default, the part is created in the workbench frame aligned along z. The trigger is not immediately visible because it is inside the lower electrode. Go to *View/Displayed parts/Regions*. In the row for the lower electrode, click the button in the *Solid* column and press *OK*. The lower electrode is displayed as a wireframe and you can see the trigger inside (Fig. 28*a*). We must introduce translations and rotations so that the assembly looks like Fig. 20. In the final configuration, the trigger should enter from the $-x$ direction at $z = 4.50$. The trigger tip is displaced 0.75 from the switch axis. To make the work easier we can highlight the trigger. Click *Edit/Select part(s)* and choose the trigger. You will see the display of Fig. 28*b*. Unselected parts are plotted in neutral colors. Click *Edit/Edit part*. The dialog automatically loads the selected part. You will need to make changes in boxes of the *Shift* and *Rotation* groups. Experiment with different values – bad parameters will cause no harm. A correct set of parameters is listed at the end of the chapter.

The final activity is to make sure that the foundation mesh parameters correspond to element sizes that will give a good representation of the objects in the solution space. Click on the *Foundation* command to enter the menu for editing the foundation mesh. Click on the *View/X normal axis* command to see the view of Fig. 29. A default division with an element width of 0.125 is superimposed on the orthogonal plot of the assembly. The choice is sufficient except



```
Normal axis: X

YGrid:    5.000E-01
ZGrid:    5.000E-01
Grid pos:  -4.491E+00
NParts:    5

 2 INSULATOR     🟩
 3 LOWERELECT    🟩
 4 UPPERELECT    🟩
 5 TRIGGER       🟥
```

```
Script: SparkGap.MIN   Not saved   NPart: 5   Snap: ON
```

Figure 29: Screen display in the *Foundation* menu

near the trigger tip. To modify the distribution of element sizes along z, we need to divide the axis into three zones and change the element size in the center zone near the trigger:

- $\Delta z = 0.1250$ in the range $-1.0 \leq z \leq 4.0$.
- $\Delta z = 0.0625$ in the range $4.0 \leq z \leq 5.0$.
- $\Delta z = 0.1250$ in the range $5.0 \leq z \leq 9.0$.

Click the command *Zaxis/Divide zone*, move the mouse cursor into the solution volume and press the left button. **Geometer** highlights the single zone in z. You could enter the position of the zone boundary by moving the mouse cursor and pressing the left button. To specify an exact position, press the *F1* key. Enter the coordinates $x = 0.00$ and $z = 4.00$ (the x value is arbitrary). The screen display is updated to show the new zone division. Perform a similar set of operations to set a zone boundary at $z = 5.0$. Then, click the command *ZAxis/Change element size*, move the mouse pointer into the central zone and press the left button. **Geometer** highlights the zone and displays a dialog box for the new element size. Enter the number 0.0625 and press OK. The vertical axis of Figure 30 shows the approximate distribution of element sizes along z. Perform a similar set of operations to set zone boundaries along the y direction at $y = -0.5$ and $y = 0.5$. Set the element size in the central zone to 0.0625. Figure 30 shows the resulting division of the foundation mesh.

To conclude, return to the main menu and click *File/Save script*. The command creates the file `WALKTHRU.MIN` in the standard **MetaMesh** script format. To view the contents, use the *File/Edit script* command. If we process the script in its present form with **MetaMesh**, we obtain the non-conformal mesh shown in Fig. 31*a*. We need to enter conformalization commands directly into the script to achieve the results of Fig. 31*b*. Conformal fitting specifications require

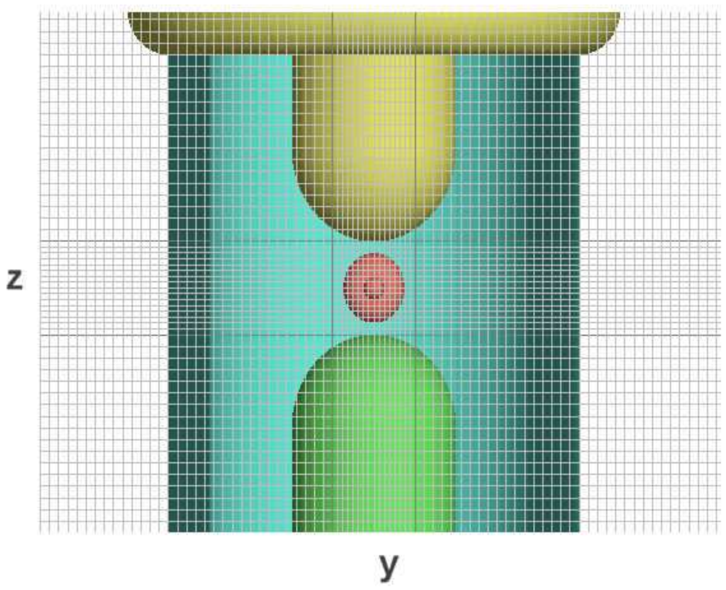


Figure 30: Detail of foundation mesh with zones of small elements defined along y and z for good resolution of the trigger electrode.

logical decisions based on insight into the geometric relationships of parts. It would be difficult and unreliable to make such decisions automatically in **Geometer**.

Answers: $XShift = -3.75$, $YShift = 0.00$, $ZShift = 4.50$, $XRot = 0.0^o$, $YRot = 90.0^o$, $ZRot = 0.0^o$.

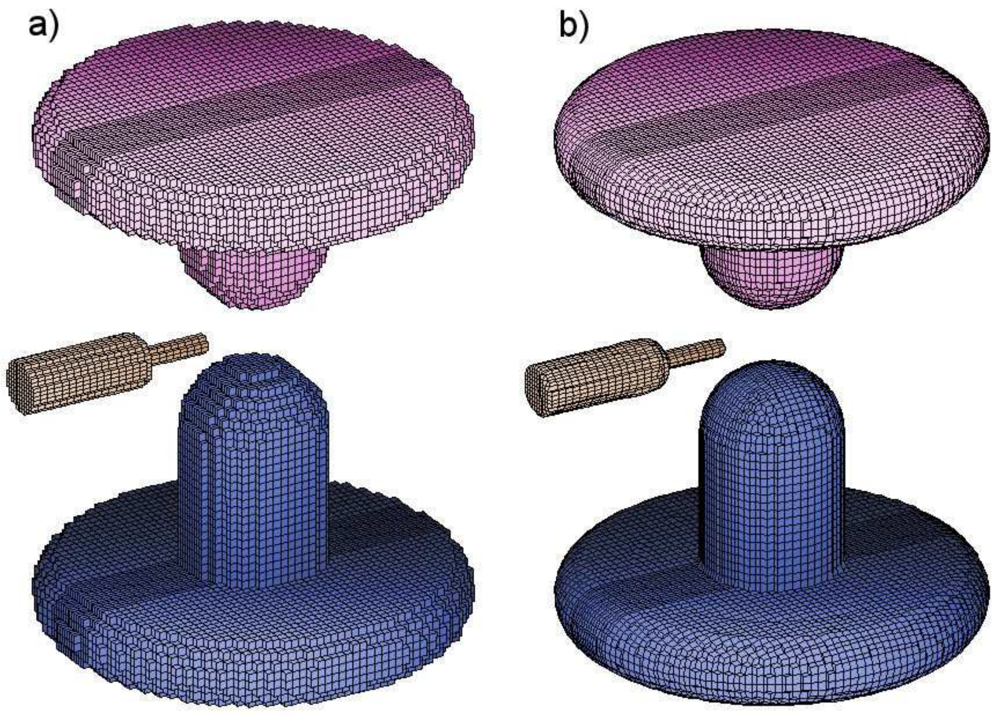


Figure 31: Mesh created by **MetaMesh** with using the script prepared in **Geometer**. *a*) Direct processing. *b*) With the additional of *Surface* commands.

5 Geometer file operations

The previous section gave a glimpse of **Geometer** capabilities. This chapter and the following ones contain detailed information on program operations. This chapter reviews commands in the *File* popup menu of the **Geometer** main menu.

NEW SCRIPT

Start a new script for development in **Geometer**. The dialog requests a file prefix (*FPrefix*) and dimensional limits in x, y and z. The prefix should be from 1 to 20 characters in length and should not contain standard delimiters (space, comma, tab, colon, equal sign, left parenthesis, right parenthesis). The script will be saved with the name `FPREFIX.MIN`. The dimensional limits give the size of the **MetaMesh** foundation mesh (solution volume). You can change these values with the *Change limits* command. **Note**: although you can choose any units, numerical accuracy is best when coordinates have values near unity. For example, the choice $x_{min} = -20.5$ μm is acceptable but $x_{min} = -2.5 \times 10^{-5}$ m is not. Also, avoid large offsets (for example, $x_{min} = 245050.0$ mm and $x_{max} = 245055.0$ mm is not acceptable. After exiting command, **Geometer** shows an orthographic plot of the solution region viewed from the $+z$ direction.. The program assigns a default first part, a box that fills the solution volume. The part is associated with region 1. Both part 1 and region 1 have the default name *SolutionVolume*. By default, the *Visible* attribute of the encompassing box is unchecked so that it will not obscure other parts that are added inside the solution volume. You can change the default assignment of part 1 using the *Edit/Edit part* command.

CHANGE LIMITS

Change the size of the foundation mesh. **Geometer** warns you that any detailed zoning information that you may have entered in the *Foundation* menu will be lost. Enter the new dimensions in the dialog.

LOAD SCRIPT

You can load any valid **MetaMesh** script (prepared with either an editor or **Geometer**) for viewing and editing. The dialog shows a list of files in the current directory with names of the form `FPREFIX.MIN`. Changing directories in the dialog changes the working directory of the program. In **Geometer** you can add parts to an existing script, modify the characteristics of existing parts, or change the order of parts. After exiting the command, **Geometer** shows a default orthographic plot of the assembly viewed from the $+z$ direction.

SAVE SCRIPT

Save the current geometry as a **MetaMesh** script.

VIEW REGION/PART LIST

This command writes information on the current geometry to the file
`GEO_LIST.DAT` and opens the file in the internal editor. The listing contains numerical information about zones of the foundation mesh, region names, and part properties. Table 3 shows a portion of the list.

EDIT SCRIPT

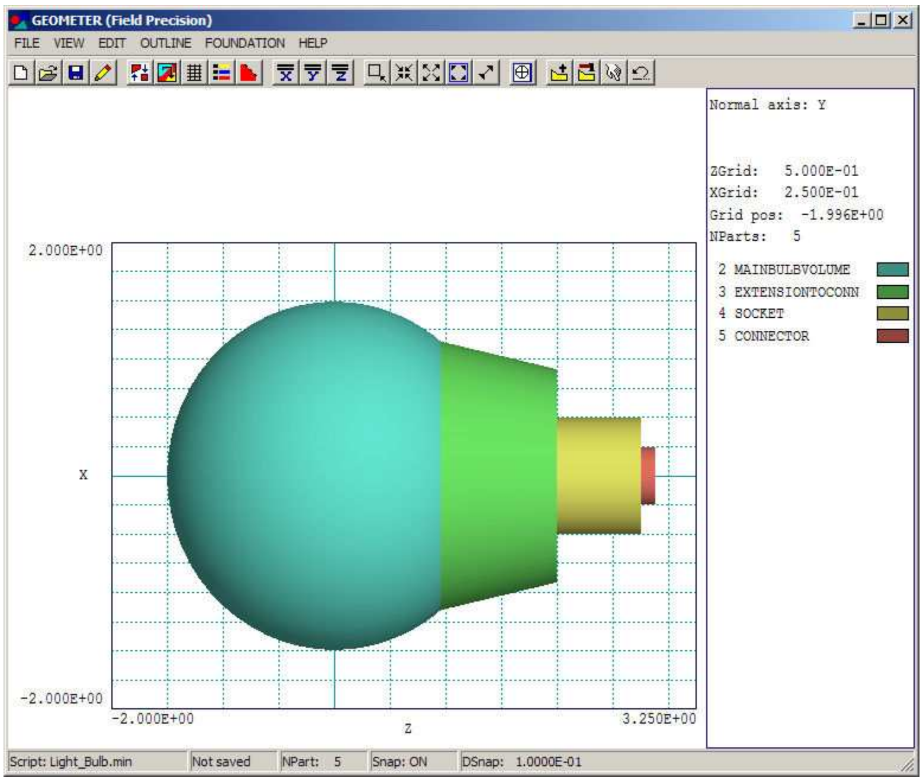
Start the program's internal editor with the option to load files with names of the form `FPREFIX.MIN`. Changing directories does not change the working directory of the program.

EDIT FILE

Start the internal program editor with the option to load any file.

Table 3: Portion of the Region/Part list for the walkthrough example

```
GEOMETER (Field Precision, Copyright 2005)
   Formatted list of foundation mesh parameters, regions
   and parts for script: C:\SH_FIELDP\SoftwareDevel\geometer\WalkThru
   Date: 06/11/2005
   Time: 15:58:12
------------------------------------------------------------
                 FOUNDATION MESH PROPERTIES
Limits
    XMin:  -4.5000E+00     XMax:   4.5000E+00
    YMin:  -4.5000E+00     YMax:   4.5000E+00
    ZMin:  -1.0000E+00     ZMax:   9.0000E+00

Zones along x-axis
      XStart       XEnd        Dx
  ====================================
   -4.5000E+00  4.5000E+00  1.1250E-01

Zones along y-axis
      YStart       YEnd        Dy
  ====================================
   -4.5000E+00 -5.0000E-01  1.1250E-01
   -5.0000E-01  5.0000E-01  6.2500E-02
    5.0000E-01  4.5000E+00  1.2500E-01

Zones along z-axis
      ZStart       ZEnd        Dz
  ====================================
   -1.0000E+00  4.0000E+00  1.2500E-01
    4.0000E+00  5.0000E+00  6.2500E-02
    5.0000E+00  9.0000E+00  1.2500E-01
------------------------------------------------------------
                    REGION PROPERTIES

  RegNo   Name
  ============================
      1   AIR
      2   INSULATOR
      3   LOWERELECT
      4   UPPERELECT
      5   TRIGGER
------------------------------------------------------------
                     PART PROPERTIES

Part number:   1   Type: Box
   Name: SVOLUME
   Region number:   1
   Dimensions
      Lx:   9.0000E+00  Ly:   9.0000E+00  Lz:   1.0000E+01
   Displacement   XOff:   0.00E+00  YOff:   0.00E+00  ZOff:   4.00E+00
   Orientation   XRot:   0.00E+00  YRot:   0.00E+00  ZRot:   0.00E+00  RotOrder: XYZ

...
```


GEOMETER (Field Precision)
FILE VIEW EDIT OUTLINE FOUNDATION HELP
2.000E+00
-2.000E+00
X
-2.000E+00
3.250E+00
Z

```
Normal axis: Y

ZGrid:    5.000E-01
XGrid:    2.500E-01
Grid pos:  -1.996E+00
NParts:    5

 2 MAINBULBVOLUME
 3 EXTENSIONTOCONN
 4 SOCKET
 5 CONNECTOR
```

```
Script: Light_Bulb.min   Not saved   NPart:  5   Snap: ON   DSnap:  1.0000E-01
```

Figure 32: **Geometer** working environment in the orthographic view mode

6 Geometer viewing options

Visualization is the key to working effectively in three dimensions. **Geometer** has rich set of viewing options to help you construct parts and to understand their relationships in an assembly. This chapter reviews commands of the *View* popup menu in the main menu. There are two points to note about the graphical environment of **Geometer**:

- The program employs OpenGL routines where many imaging tasks are carried out on your graphics card. The quality of the display may be poor with a low-end card.
- Because image processing is performed for the screen display, there are no available OpenGL routines to make direct hard copies or graphics files. To preserve images you can use a screen copy utility. We supply the freeware program MWSnap and highly recommend Paintshop Pro.

ORTHO/PERSPECTIVE

Switch between the two **Geometer** viewing styles. The *Ortho* viewing mode (Fig. 32) is similar to a mechanical drawing of a three-dimensional assembly. The screen shows an orthographic

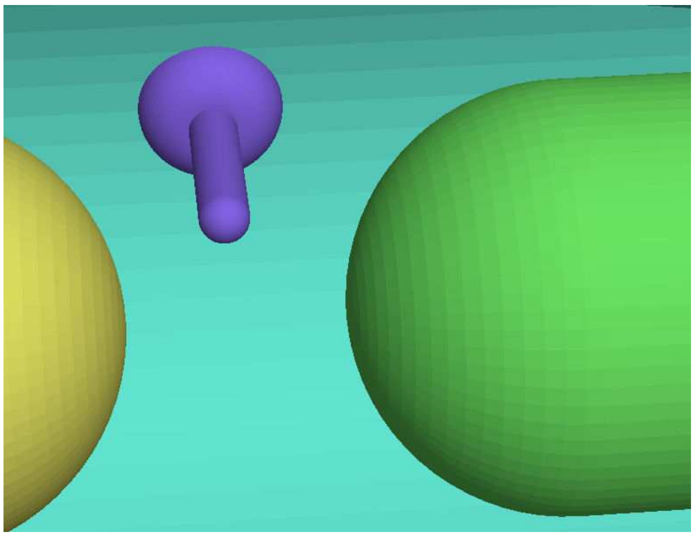


Figure 33: Perspective view of the walkthrough example – the zoom factor has been set to look inside the insulator.

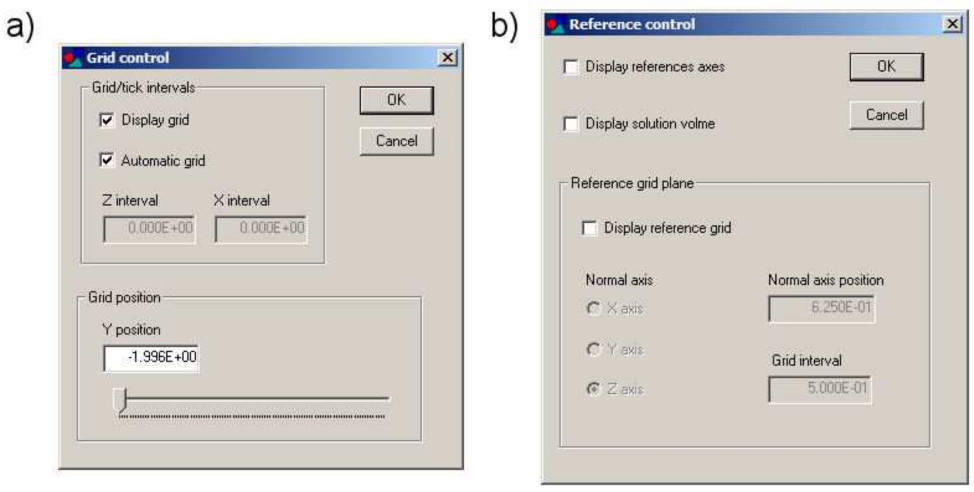
projection of objects viewed from the $+x$, $+y$ or $+z$ directions. The difference from a mechanical drawing is that the program shows the full three-dimensional objects rather than a slice in a plane. You must set up clipping planes to look inside objects. **Geometer** includes reference grids in the plot to check dimensions and positioning. The grid interval is listed in the information window on right-hand side. In the automatic mode, the grid intervals vary in response to magnification changes. You can also set intervals manually using the *Grid control* command. The *Perspective* mode (Fig. 33) is a realistic portrayal of how the objects appear in three-dimensional space. Using mouse controls in an animated environment, you can walk around or even through objects. Figure 25 shows the active screen areas for mouse control. In both modes, objects are color coded by part or region number. In the *Ortho* mode a key to the colors appears in the information window on the right-hand side of the screen.

TOGGLE PARTS/REGIONS

Color coding may be applied by part or region number. This command toggles between the two. The setting affects the display in the *Ortho* information window and also the available choices in the *Displayed parts/Regions* command.

GRID CONTROL

The form of the grid control dialog depends on whether the viewing mode is *Ortho* or *Perspective*. Figure 34*a* shows the dialog in the *Ortho* mode. The first check box turns grid display *ON* and *OFF*. The second check box controls whether the grid intervals are determined manually or automatically by the program. In automatic mode, the program picks a good grid interval that may change as the view zooms in and out. If you uncheck the automatic mode and supply values, the intervals are fixed independent of the view magnification. The final control group determines the position of the grid along the viewing axis. In **Geometer** the grid is a plane in three-dimensional space that may be moved forward and backward relative to the objects in the solution. The default is to place the grid on the back side of the solution volume. Use the


a)

Grid control

Grid/tick intervals

[✓] Display grid

[✓] Automatic grid

Z interval	X interval
0.000E+00	0.000E+00

Grid position

Y position: -1.996E+00

b)

Reference control

[ ] Display references axes

[ ] Display solution volme

Reference grid plane

[ ] Display reference grid

Normal axis: X axis / Y axis / [✓ Z axis]

Normal axis position: 6.250E-01

Grid interval: 5.000E-01

Figure 34: Dialogs to control grid displays. a) *Ortho* mode. b) *Perspective* mode.

slider or type in a value to change the position. Figure 34*b* shows the grid control dialog in the *Perspective* mode. When the *Display reference axes* checkbox is active, **Geometer** includes reference x, y and z axes to help you understand the orientation of the display. Note that the axis labels are three-dimensional letters and may not be visible from some angles. The origin will be at (0.0, 0.0, 0.0) if the point is inside the solution volume. Otherwise the axis origin is set at the center of the solution volume. When the *Display solution volume* checkbox is active, the program includes the boundaries of the solution volume as a wireframe box. You can use the commands of the *Reference grid plane* command to include fiducial lines on a square mesh in the plot. Again, the reference grid is a physical object that may be placed in different positions in the solution volume. Supply values for the normal axis, position along the normal axis and the grid interval (applied in both the horizontal and vertical directions). Figure 35 shows a view with all grid options active. The reference grid is located at $z = 4.0$ with an interval of 0.5.

DISPLAYED PARTS/REGIONS

Use this command to suppress the display of certain parts or regions or to change their plot style. The command affects both the *Ortho* and *Perspective* modes. The row choices will depend on whether color coding is set to *Parts* or *Regions*. In the *Solid* column, depress the button for a hidden-surface surface representation. Raise the button for a wireframe display where you can see through an object. Note that the wireframe represents the facets created for the OpenGL display and is not related to the mesh structure of the object generated in **MetaMesh**. The ends of extrusions and turnings may have a large number of lines because automatic tessellation routines are invoked. You can turn off parts or regions completely using the buttons in the *Visible* column.

PART CLIPPING

You can apply clipping planes to any part in order to expose objects that may be inside. There are six planes that are function when the box *Clipping planes active* is checked. The first plane (normal to the x axis) is at position $XLower$. Only portions of the part with $x \geq XLower$

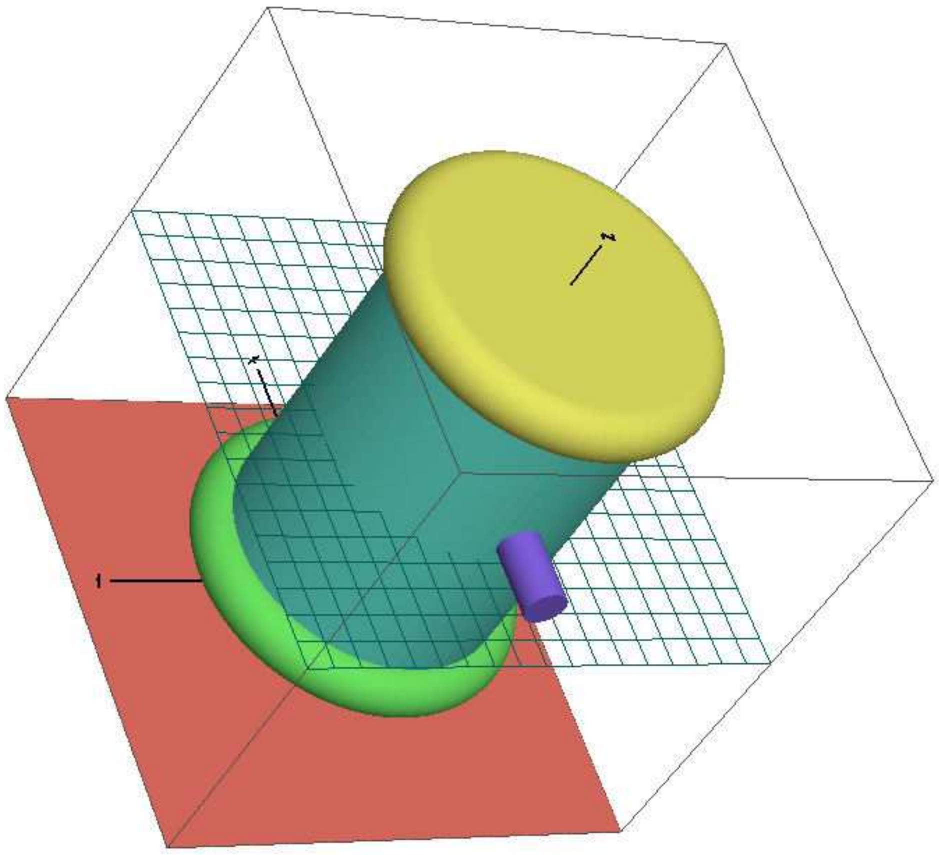


Figure 35: Perspective view with all grid options active.

will be displayed. The second plane (also normal to x) is at position $XUpper$. Only portions of the part with $x \leq XUpper$ will be displayed. The other four planes are normal to y or z and serve similar functions. By default, the clipping planes are initially set to the boundaries of the solution volume. When the box *Clipping planes active* is unchecked, the full extent of objects is displayed in the perspective view.

X NORMAL AXIS
Y NORMAL AXIS
Z NORMAL AXIS

In the *Ortho* mode, these commands determine the position of the viewer ($+x$, $+y$ or $+z$ direction). The command is useful in the *Perspective* mode if you have become disoriented while moving around the solution volume. For example, the *X normal axis* command sets a standard, centered view from the $+x$ direction.

SHOW COORDINATES

Sometimes it may be inconvenient to determine a location in an orthogonal plot from the grid. Use this command to display positions in the normal plane. When you move the mouse into the plot window, the cursor changes to a cross-hair pattern. Coordinates are displayed in the status bar at the bottom of the screen. Press the left button to display coordinates at a point. Press the right mouse button or the *Esc* key to exit the coordinate mode. It is important to note that the returned coordinates depend on the snap mode setting. Turn off the snap mode to estimate points that are not on the snap grid.

The following commands are active only in the *Ortho* mode.

ZOOM WINDOW

Create z zoomed view of an area by defining two points of a box with the mouse or keyboard entries. When the command is issued, the mouse becomes active. When you move it into the plot area the cursor changes from an arrow to a cross-hair pattern. The mouse coordinates are displayed in the status bar. When using the mouse, you may enter coordinates for any point via the keyboard by pressing the *F1* key. Press the right mouse button or the *Esc* key to exit.

MAGNIFY VIEW

Narrow the view around the current center of the plot.

EXPAND VIEW

Expand the view around the current center of the plot.

GLOBAL VIEW

Expand the view to show the full solution volume.

PAN

Move the current plot center by defining two points of a displacement vector with the mouse or keyboard entries.

TOGGLE SNAP MODE

In the snap mode, the mouse moves discontinuously to positions that are an integral multiple of the distance *DSnap*. It may be necessary to turn off the snap to zoom the view to a small region or to use the *Show coordinates* command to find points between the snap grid positions.

SET SNAP LENGTH

Set the quantity *DSnap* for the mouse snap mode.

To conclude, there is a feature of the OpenGL display that you should note. When you look inside a part in the *Perspective* mode, you may see a speckled effect if the part shares a boundary with another. As the view rotates, round-off errors may place some facets of one part closer than those of the other. Therefore, the OpenGL hidden surface routines may plot a mixture of facet colors on the shared boundary. If this bothers you or if you want to create a high-quality plot for a presentation, make the part that will appear earlier in the script slightly larger to remove the ambiguity. Because of the over-writing principle, this should not affect the mesh generation procedure in **MetaMesh**.

7 Adding and editing parts in Geometer

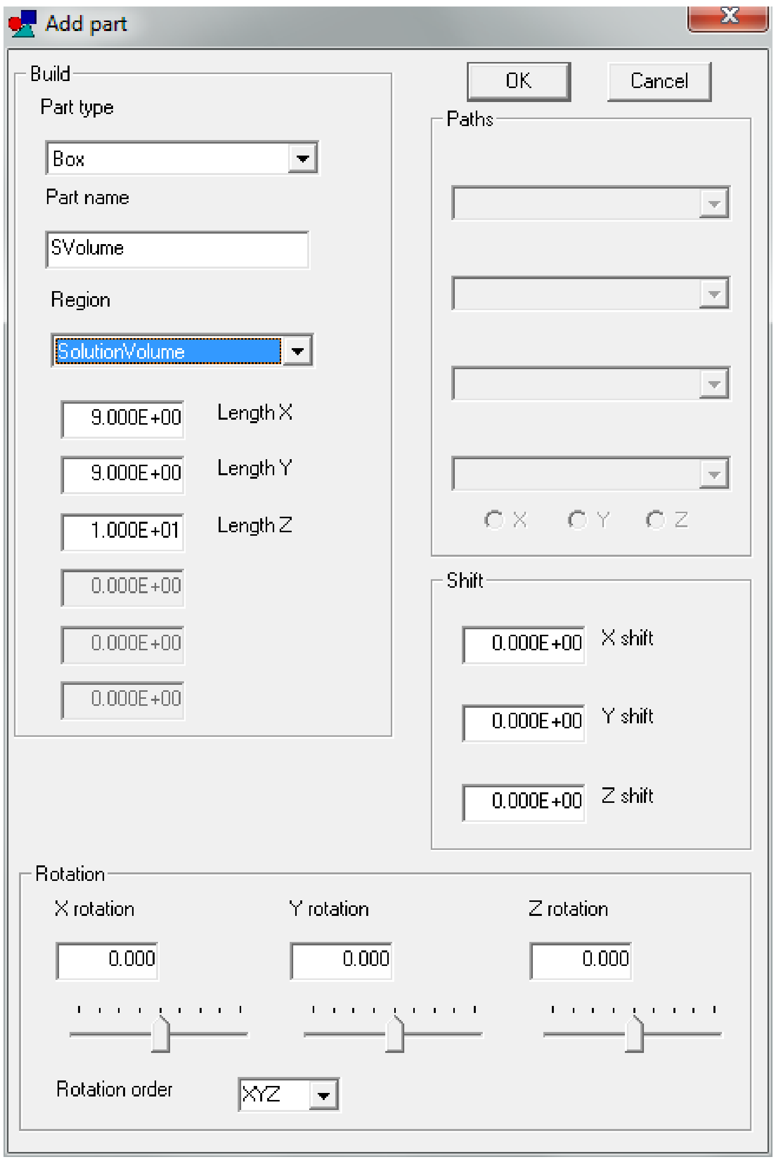
Geometer includes a powerful set of commands in the *Edit* pull-down menu to add and to modify parts. You can can create a script from scratch or change features of an existing script. Modifications are immediately visible in the plots.

EDIT REGION NAMES

This command is recommended as the first step in building a new script. After deciding on the different materials that will enter the physical solution, create descriptive names for the corresponding numbered regions. This effort will make it easier to remember the functions of regions that are assigned to parts. Click the mouse in the appropriate row of the *Name* column and type the region name. Region names should be unique and have a length from 1 to 20 characters. They must not contain any of the standard delimiters. Note that you can add new names or change existing names at any time.

ADD PART

This command brings up the dialog of Fig. 36. Within the dialog you can fabricate, orient and position a part. The part will be added to the display when you exit. There is no need to worry about getting everything correct – all parameters can be changed later with the *Edit part* command. When the dialog first appears, only the *Part type* field is active. To begin, you must choose a model. In this chapter, we shall concentrate on the simple solid and planar models discussed in Chap. 8 (*i.e.*, spheres, cones, boxes, disks,...). Chapter 9 discusses turnings and extrusions with arbitrary cross sections and Chap. 12 covers advanced models. After you pick a model, several fields appropriate to the shape become active. The next step (optional, but recommended) is to give the part a name. Click in *Part name* box and then type a name (1-20 characters). This name will be used in the part selection dialog and also in part plots in both **Geometer** and **MetaMesh**. The next required operation is to pick a region that will be associated with the part in the *Region* field. Click on the arrow to show a menu of the available regions, and then highlight the choice. The fabrication parameter boxes are below the region field. Note that only boxes appropriate to the model are active and that descriptive labels have been added. Enter values in any valid real-number format. With no further input, the part will be constructed at the default workbench orientation and position described in Chap. 8. You can add displacements and rotations to position the part in the assembly space. Remember that rotations are performed before shifts. To enter a rotation angle, you can directly type the value (in degrees) in the appropriate box. You can also use the sliders below the boxes. Click to the side of the slider to advance in increments of 45°. When the slider is active, you can use *PGuP* and *PgDn* on the keyboard to move in increments of 45° or *UpArrow* and *DownArrow* for 5° increments. By default, rotations are performed in the order $\theta_x \rightarrow \theta_y \rightarrow \theta_z$. You can change the order by picking an option in the *Rotation order* menu. We will discuss entries in the boxes of the *Path* group on the right-hand side of the dialog in Chaps.9 and 12. When you click *OK*, **Geometer** performs extensive error checking to confirm that parameters are valid for the model. If the check is successful, the program exits the dialog and updates the display.


Add part

Build

Part type: Box

Part name: SVolume

Region: SolutionVolume

9.000E+00	Length X
9.000E+00	Length Y
1.000E+01	Length Z
0.000E+00	
0.000E+00	
0.000E+00	

Shift

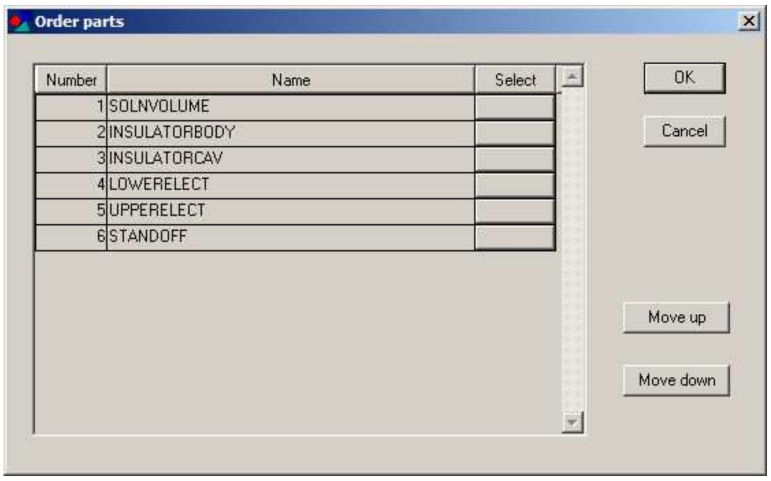
0.000E+00	X shift
0.000E+00	Y shift
0.000E+00	Z shift

Rotation

X rotation	Y rotation	Z rotation
0.000	0.000	0.000

Rotation order: XYZ

Figure 36: Dialog to add or to edit parts


Order parts

Number	Name	Select
1	SOLNVOLUME	
2	INSULATORBODY	
3	INSULATORCAV	
4	LOWERELECT	
5	UPPERELECT	
6	STANDOFF	

Figure 37: Dialog to change the order of parts

EDIT PART

In response to this command, **Geometer** displays a dialog with a list of available parts. It is easy to identify them if you have assigned names. Pick a part and click *OK*. The program then displays the dialog of Fig. 36. The main difference is that the current parameters of the part are entered in the fields. Again, only fields appropriate to the part type are active. You can change any of the parameters including *Part type*. When you click *OK*, **Geometer** does an error check and exits if successful. The current display is updated to show the modified part.

CHANGE PART ORDER

The order of parts in the **MetaMesh** is important. In the constructive-solid-geometry process, parts that appear later in the script over-write any spaces shared with previously-defined parts. Use this command to set the order in which parts will be written to a script by the *Save script* command. The command calls up the dialog of Fig. 37. Choose a part by pressing a button in the *Select* column. You can move the part in the stack by pressing the *Move up* or *Move down* buttons. If the plot is in the *Display parts* mode, the colors of objects will change when you exit the dialog.

SELECT PART(S)

There are two reasons to select parts: 1) selected parts have increased visibility in plots and 2) selected parts are the targets of editing operations. A dialog appears in response to the command. Depress one or more buttons in the *Select* column. When you exit the dialog, selected parts are displayed in bold colors and unselected parts in neutral colors. Note, if a single part is selected, it will automatically appear in the dialog for the *Edit part* command. This feature makes it easier to move a part into position with sequential operations.

INVERT SELECTION

Change the selection criterion so that deselected parts are selected and vice versa.

DESELECT ALL
Deselect all parts in the assembly.

MOVE SELECTED
With this command, you can apply a displacement to all selected parts. In the dialog, enter displacements for the x, y and z components of the displacement. The *XShift*, *YShift* and *ZShift* parameters for the selected parts are modified. You can check the changes by editing the part.

DELETE SELECTED
Remove all selected parts and reorder the parts list.

CHANGE REGION SELECTED
The program displays a dialog with a menu where you can select a region, and then changes the region assignment of all selected parts.

COPY SELECTED
Geometer makes copies of the selected parts and assigns new part numbers $(NParts + 1)$, $(NParts + 2)$, ... The old and new parts overlap in space, so the new ones may not be visible. Note that the old parts are de-selected and the new ones are selected. Therefore, application of the *Move selected* command will displace the new parts as a group from the position of the old parts. You can assign the common region number to the new parts with the *Change region selected* command. Use the *Edit parts* command to change the size, position or rotation of individual new parts. You can position the new parts in the stack with the *Change part order* command.

UNDO LAST
Reverse the last editing operation. This affected operations are (*Add part*, *Edit part*, *Move selected*, *Delete selected* and *Change region selected*).

8 Geometer/MetaMesh - standard parts bin

This chapter reviews models for simple geometric objects used in **Geometer** and **MetaMesh**. We can divide the models into two classes:

- **Filled parts**. Solid objects where a region number is assigned to a set of contiguous elements and their connected nodes.
- **Open parts**. Point or planar objects where the region number is assigned only to a set of nodes.

This chapter gives a reference list of fabrication parameters and default workbench positions.

8.1 Filled parts

There are fives types of available filled parts: 1) basic shapes, 2) extrusions, 3) turnings, 4) transitions and 5) the neon model. 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 filled shapes in this section. Chapter 9 covers extrusions and turnings while Chap. 12 covers transitions and the neon model.

In the following list, the parameter *Type* is the choice in the *Part type* menu of the *Add part* and *Edit part* dialogs of **Geometer**. The quantity is also the string entry that appears in the *Type* command of the *Parts* sections of the **MetaMesh** script. *Dimensions* are the real-number fabrication parameters that appear in the **Geometer** dialogs and the *Fab* script command of **MetaMesh**. The *Comments* describe the default position and orientation of the part before rotations and shifts are applied.

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 along x), 2) L_y (full length along y) and 3) L_z (full length 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.

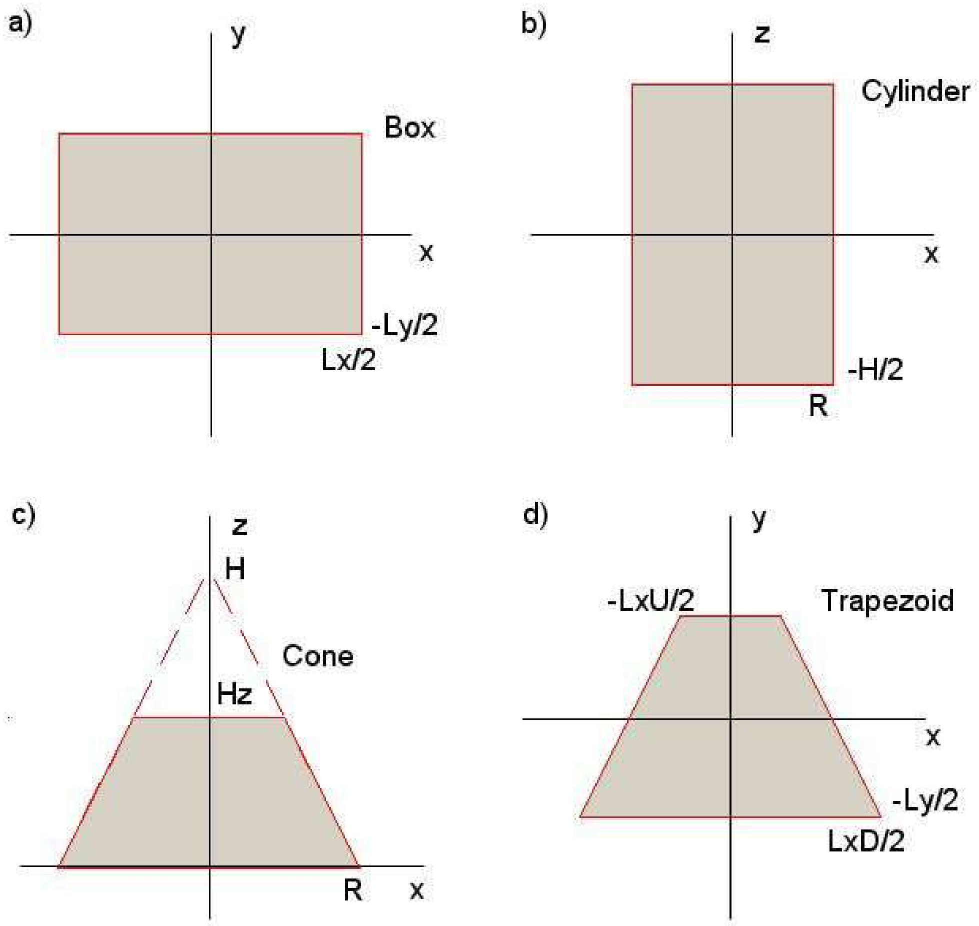


Figure 38: Basic solid shapes in the workbench frame

Type: CYLINDER
Dimensions: 1) R (radius) and 2) H (height)
Comments: The center-of-mass of the cylinder is at (0.0, 0.0, 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 with 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 $z = 0.0$. 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.

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, 0.0, 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

$$\left(\frac{x}{R_x}\right)^2 + \left(\frac{y}{R_y}\right)^2 = 1.$$

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

$$\left(\frac{x}{R_x}\right)^2 + \left(\frac{y}{R_y}\right)^2 + \left(\frac{z}{R_z}\right)^2 = 1.$$

Type: TORUS
Dimensions: 1) R (major radius) and 2) r (minor radius)
Comments: The torus is centered at (0.0, 0.0, 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 helical 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 bounding cylinder extends from 0.0 to H along z. The projection of the helix winding in the x-y plane extends from a radius of $R-r$ to $R+r$. In the workbench space the shape starts on

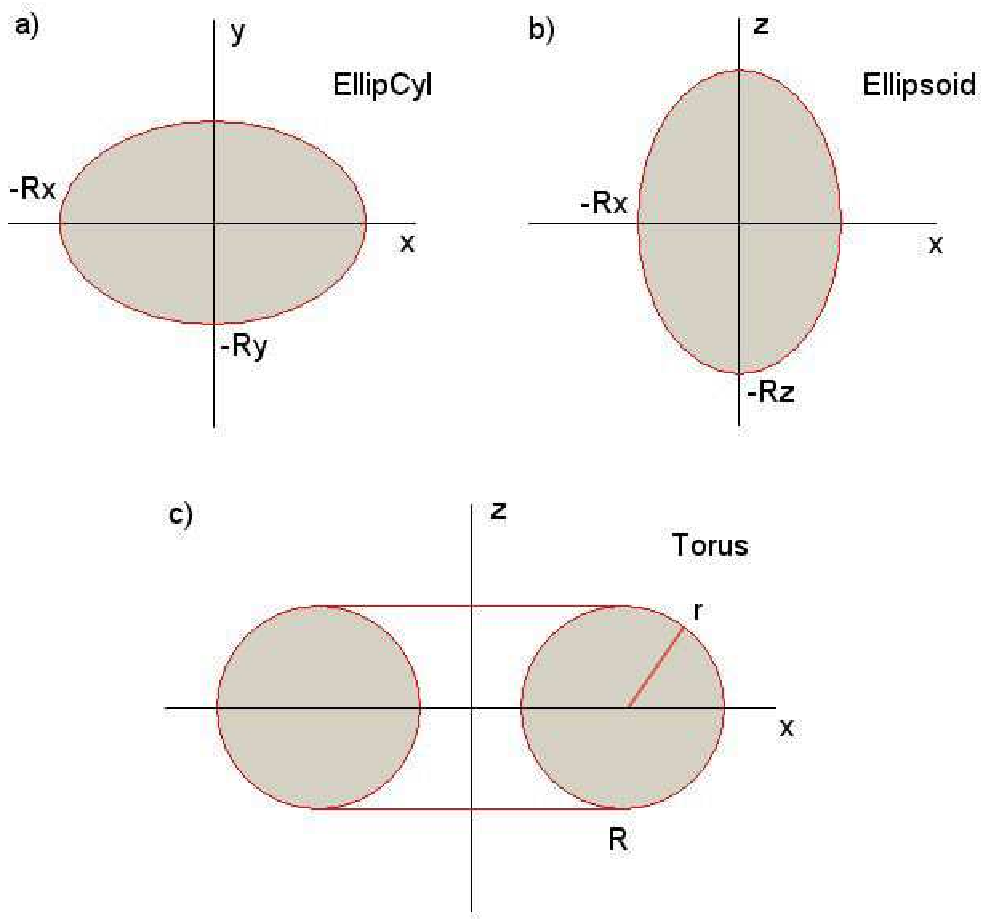


Figure 39: Basic solid shapes in the workbench frame

the x axis. By default, the shape follows the direction of positive rotation (counter-clockwise). To construct a helix with negative rotation, use negative values for both H and H_w.

Type: TRAPEZOID
Dimensions: 1) L_{xU} (full width in x, top side), 2) L_{xD} (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 $-L_y/2$ to $L_y/2$ in y and $-L_z/2$ to $L_z/2$ in z. The full width in x is L_{xD} at $-L_y/2$ and L_{xU} at $L_y/2$.

8.2 Open parts

Open parts have no volume – they consist only of nodes. In **MetaMesh** 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 the workbench space. In other words, a line of length L points along the z axis on the workbench between the points $(0.0, 0.0, -L/2)$ and $(0.0, 0.0, L/2)$. You can apply rotation and shift operations to create general lines in the assembly space.

Type: POINT
Dimensions: None
Comments: A point object is created at the location (0.0, 0.0, 0.0) on the workbench. Apply shifts to position the point in assembly space. Rotation commands have 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 volume to the value specified in the *Region* command for the part. Shifts and rotations 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 workbench space between the points $(0.0, 0.0, -L/2)$ and $(0.0, 0.0, L/2)$. Use rotations and shifts to create general lines in solution space. If you want to connect two particular points in solution space, use the *Line Converter* tool in **MetaMesh** to find L and the 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 shift and rotation 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 the workbench. The arc center is at the origin of the workbench frame, $(0.0, 0.0, 0.0)$. The arc starts at θ_{min} and extends to θ_{max}. Enter the angles in degrees. Note that the angle $\theta = 0.0^o$ corresponds to the x axis. You can use the *Arc Converter* tool in **MetaMesh** to switch between workbench and assembly space coordinates.

Type: CIRCLE
Dimensions: 1) R
Comments: The Circle has radius R and lies in the x-y plane on the workbench. . The center of the circle is at the workbench origin $(0.0, 0.0, 0.0)$.
Type: RECTANGLE
Dimensions: 1) L_x, 2) L_y
Comments: The rectangle lies in the x-y plane of the workbench with sides parallel to the x and y axes. It is centered at the workbench origin $(0.0, 0.0, 0.0)$. 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 the workbench. The center of the disk lies at the workbench origin $(0.0, 0.0, 0.0)$.

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 the workbench with sides parallel to the x and y axes. It is centered at the workbench origin $(0.0, 0.0, 0.0)$. 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: BUBBLE
Dimensions: 1) R
Comments: A bubble is a spherical surface of nodes with radius R centered on the origin of the workbench.

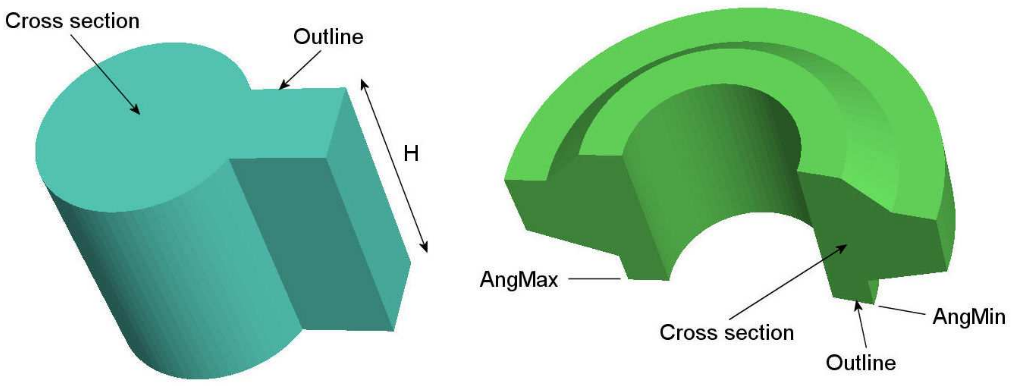


Figure 40: Fabrication parameters for extrusions (left) and turnings (right)

9 Geometer – extrusions and turnings

Extrusions and turnings greatly extend the capabilities of **Geometer** and **MetaMesh** to model complex shapes. An *extrusion* (left side of Fig. 40) is the type of part you would construct on a milling machine. It has an arbitrary cross section in either the x-y, y-z or z-x plane of the workbench frame and extends over the range $-H/2 \leq z \leq H/2$ in the direction normal to the plane. A *turning* (Fig. 40 is the type of part that you would fabricate on a lathe. In this case, a cross section in z-r of the workbench frame is rotated about the z axis. You can also define limits for the azimuthal extent of a turning. In this chapter, we shall concentrate on how to build extrusions, turnings in **Geometer**. Chapter 18 discusses how the models are represented in the **MetaMesh** script.

The cross sections of extrusions and turnings are defined by *outlines*. An outline is a text list of up to 50 line and arc vectors in a plane. Each line of the list gives a vector, and the list must terminate with the *End* command. The vectors must connect to one another and surround a closed region in the plane. **Geometer** contains graphical and text editors for building outlines. These capabilities are described in the next chapter.

In the **MetaMesh** script or the **Geometer** outline text editor, a line vector for an extrusion has the format[3]:

```
L  XStart  YStart  XEnd  YEnd [S]
```

For example, the vector

```
L  0.0  0.0   1.0  2.0
```

extends from the point $(x = 0.0, y = 0.0)$ to $(x = 1.0, y = 2.0)$. A line vector for a turning contains the quantities:

```
L  ZStart  RStart  ZEnd  REnd  [S]
```

[3]The optional S character controls conformal surface fitting in **MetaMesh** (see Sect.18.4)

An arc vector for an extrusion has the format

```
A  XStart  YStart  XEnd  YEnd  XCenter  YCenter  [S]
```

The arc format for a turning is

```
A  ZStart  RStart  ZEnd  REnd  ZCenter  RCenter  [S]
```

An arc must cover less than 90°. Break larger arcs into two or more parts. For example, the following outline defines half a circular disk 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
END
```

To create an extrusion in **Geometer**, one or more outlines must be available. There are several ways to load outlines into the program:

- The outlines for all included extrusions and turnings are added when you load a **MetaMesh** script with the *File/Load script* command in the *Main* menu.
- In the *Outline* menu, you can load outline files. An outline file contains a valid vector set and the *End* command – it has a name of the form `FPREFIX.OTL`.
- Use a text editor to copy and paste outlines from **Mesh** or **MetaMesh** scripts.
- You can create new outlines or modify old ones using the graphical and text editors in the *Outline* menu.

Assuming an appropriate outline is available in **Geometer**, click the *Edit/Add part* command to add an extrusion. Choose *Extrusion* from the *Part type* menu in the dialog. One of the fabrication boxes on the left-side will become active. Enter *H*, the height of the extrusion. Choose the appropriate outline from the active outline menu on the right-hand side. Also, choose the axis normal to the definition plane using the radio buttons below the path. To illustrate, if the x button is checked, the program interprets the outline as a shape in the y-z plane. You can also add rotations and translations. When you click *OK*, the extrusion is added to the current plot. Don't worry about getting everything right – you can always return to the dialog through the *Edit/Edit part* command to make corrections:

- You can incrementally change rotation and translation parameters to move an extrusion into place.
- You can change the height H or even assign a different outline.
- If you modify an outline in the *Outline* menu, all parts that use the outline are automatically updated.

To create a turning, choose *Turning* from the *Part type* menu in the *Edit/Add part* dialog. In this case, two fabrication parameter boxes become active. The parameters are the minimum and maximum angular extent of the turning in the x-y plane of the workbench. The angle 0.0° corresponds to the x axis. To create a full turning, accept the default of ($AngMin$ = 0.0°, $AngMax$ = 0.0°) or enter $AngMin$ = 0.0° and $AngMax$ = 360.0°.

10 Geometer - import from 3D CAD programs

Geometer and **MetaMesh** can import multiple complex objects from SolidWorks, ProE, NX and other three-dimensional CAD programs. The translation procedure is realiable and easy. Information is conveyed through standard STL (STereo Lithography) files. The following section reviews the format and features of STL files. Section 10.3 describes methods to add and to edit STL objects in the interactive environment of **Geometer**. Section 10.4 describes special utilities to modify the properties of STL files.

10.1 STL file format

Stereo lithography files form the basis of an industry for the fabrication of complex solid objects by computer-controlled milling machines. An STL file describes a single solid object by resolving its surface into a set of triangular facets (Fig. 41). There are several reasons why we have chosen STL as the method for importing CAD information into **Geometer** and **MetaMesh**:

- The file format is simple and rigidly defined. Developers of three-dimensional CAD programs cannot add proprietary features or make changes with new program versions.
- Part replication through computer-controlled milling is a widespread application, so there is a strong motivation to support STL export in CAD programs.
- The convention of recording one solid object per STL file is matched to the concept of *parts* in **Geometer** and **MetaMesh**.
- There is extensive information about STL files on the Internet. Furthermore, there are many free or low cost utility programs for viewing and/or modifying the files.

The following article describes the STL file format:

`http://en.wikipedia.org/wiki/STL_(file_format)`

A file contains a sequential list of surface facet parameters. Each facet entry includes the spatial coordinates of the triangle vertices and the components of a vector normal to the facet pointing out of the solid. An STL file has a name of the form `FPREFIX.STL`. The file may be either in text or binary format. **Geometer** and **MetaMesh** can read both types with automatic format detection.

In principle, the facets of a valid STL file should all join together at the vertices, defining a closed surface around a contiguous solid. By *contiguous*, we mean that it is possible to travel between any two points in the object along a path that remains within the object and does not cross a surface. Usually, STL files that you create with a CAD program will fulfill the requirements if you limit export to a single solid object. On the other hand, there is no guarantee that every STL file you encounter will be perfect. You can inspect STL files with the three-dimensional viewer described in Sect. 10.5. The following Internet page reviews procedures to export STL files from a variety of CAD programs:

`http://www.fieldp.com/stl_export.html`

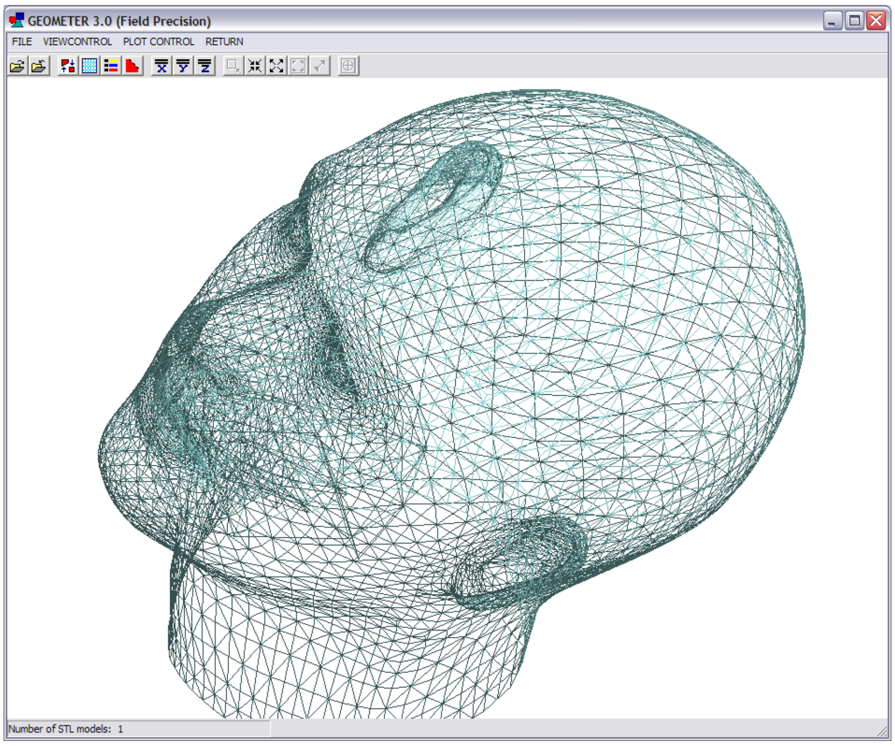

GEOMETER 3.0 (Field Precision)
FILE VIEWCONTROL PLOT CONTROL RETURN
Number of STL models: 1

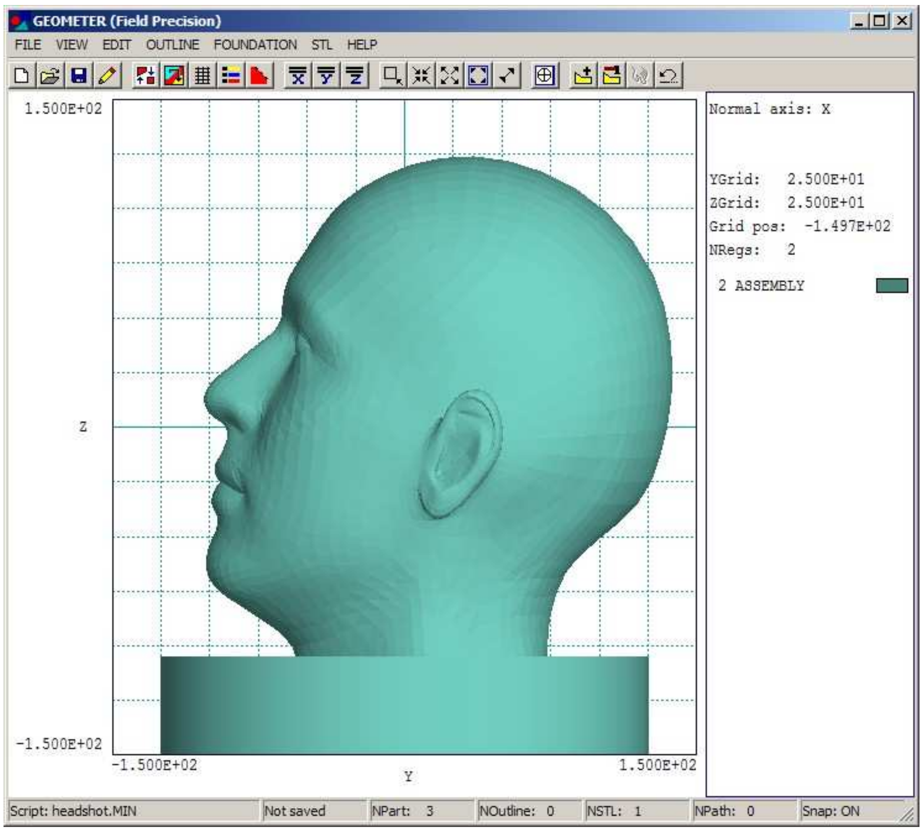
Figure 41: Screenshot of the **Geometer** *STL View* utility, showing a wireframe view of an STL object in perspective mode.

10.2 The prime directive for STL file creation

The generation of meshes with STL objects can be easy and reliable if you follow one simple rule: always break a complex assembly into manageable parts. Of course, this division is necessary if objects in an assembly have different physical properties with respect to the field solution (such as dielectric and electrodes in electrostatic solutions). Even if components have the same properties, it may be useful to break an object into parts. Such a division reduces the possibility of code errors and gives a significant increase in processing speed. For example, dividing a part in two could reduce the total processing time by an order of magnitude. You loose nothing by dividing an object. The STL file contains the absolute coordinates of the facets so that the component parts will be in the correct positions when assembled in **Geometer** and **MetaMesh**. The only limitation is the maximum number of 50 STL models that can be loaded in **Geometer** at any one time.

10.3 Adding STL objects to an assembly

In general, you must preload information to utilize complex part models in **Geometer**. For example, you must load outlines in order to create extrusions and turnings (Chap. 9) and you must load paths before building a part based on the neon model (Sect. 12.2). Similarly, it is necessary to load one or more STL files before adding an STL part. The files must be available in the working directory. As a rule, copy all the STL files that you will need to the working directory before starting a **Geometer** assembly.


GEOMETER (Field Precision)
FILE VIEW EDIT OUTLINE FOUNDATION STL HELP
1.500E+02
-1.500E+02
Z
Y
-1.500E+02
1.500E+02

Normal axis: X

YGrid: 2.500E+01
ZGrid: 2.500E+01
Grid pos: -1.497E+02
NRegs: 2

2 ASSEMBLY 🟩

Script: headshot.MIN | Not saved | NPart: 3 | NOutline: 0 | NSTL: 1 | NPath: 0 | Snap: ON

Figure 42: **Geometer** working environment showing an orthographic view of an STL part combined with a geometric model.

LOAD STL MODEL

In response to this command, **Geometer** displays a dialog listing files with names of the form `FPREFIX.STL`. Choose a file **in the current directory**. After you pick a file and click *OK*, the program performs the following operations: 1) attempts to open the file, 2) determines whether it is in text or binary format, 3) checks for syntax errors and 4) computes spatial limits and the coordinates of the center-of-mass. **Geometer** issues an error message if there is a problem or displays spatial information if the load is successful. The maximum number of loaded STL models is 50. Note that a model may be used in the definitions of multiple parts.

When you load a `MIN` file that includes STL models, **Geometer** clears currently-loaded files and automatically loads the files listed in the script. Information in the files is used for the display and is available for creating additional parts. All referenced STL files must be available in the working directory.

LIST STL MODELS

Show a message box listing loaded STL models.

An STL object is treated like any other part in **Geometer**. All viewing options (such as cut planes and wireframe display) are available (Figure 42). You may include rotations and translations to change the orientation and position of an STL part in the assembly. An assembly may contain multiple STL parts combined with any of the other models described in the preceeding sections. Figure 43 shows the appearance of the *Add part* dialog described in Sect. 7 when adding an STL part. The STL option is at the bottom of the list in the *Part type* menu. The box marked *STL model* on the right is active and the menu lists the available loaded files. The function of the two fitting parameters (*Fit toler* and *CutOff*) are described in Sect. 18.1. They do not affect the operation of **Geometer**. If you do not specify values, the program enters default values when writing the `MIN` file.

10.4 STL utilities

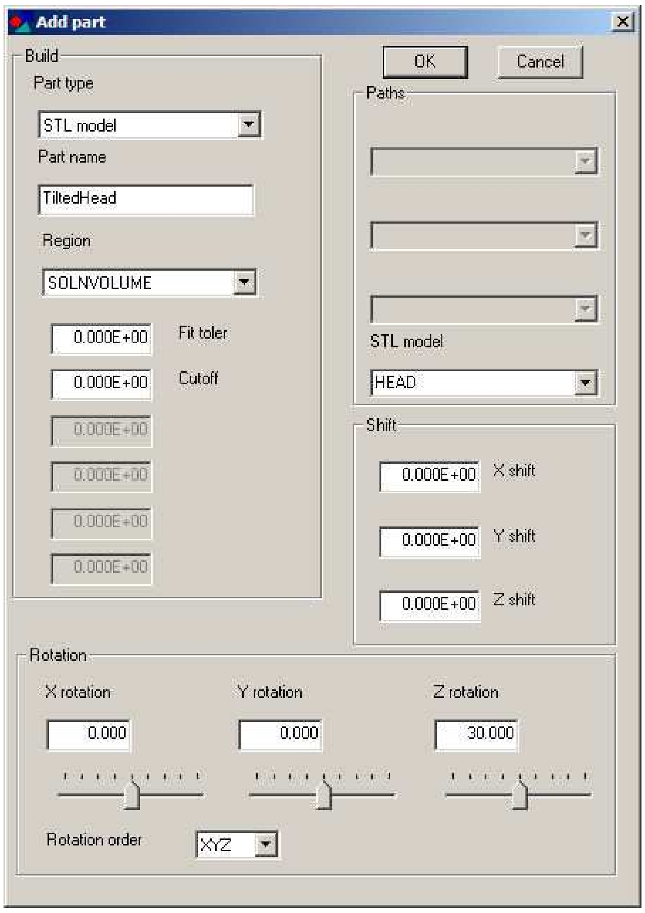
SHOW STL FILE PARAMETERS

Use this command to check files in the working directory or other directories. After you pick a file, a message box shows the file format (text or binary), the number of facets, spatial limits of nodes and the center of mass position.

SCALE STL MODEL

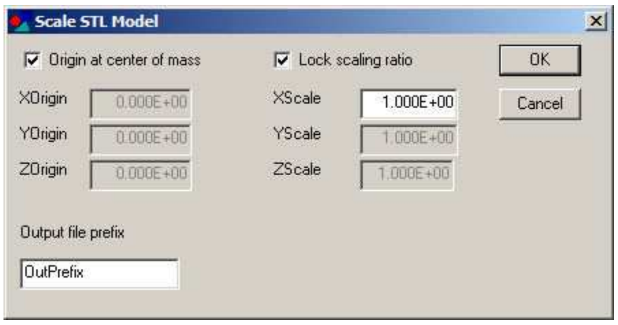
This tool performs transformations on STL files. The dialog of Fig. 44 appears after you pick a file of type `FPREFIX.STL` from the current directory. Use the group of fields on the left to set the scaling factors (S_x, S_y, S_z) and the group on the right to set the scaling origin (x_o, y_o, z_o). The output file is saved in the working directory with the name `FPREFIXS.STL`, where *FPrefix* is the supplied file prefix. Note that **Geometer** updates the normal vectors to reflect any changes in the proportions of the object.

The transformation equations for the scaling operation are:



Part type: STL model
Part name: TiltedHead
Region: SOLNVOLUME
Fit toler: 0.000E+00
Cutoff: 0.000E+00
0.000E+00
0.000E+00
0.000E+00
0.000E+00
STL model: HEAD
Shift
X shift: 0.000E+00
Y shift: 0.000E+00
Z shift: 0.000E+00
Rotation
X rotation: 0.000
Y rotation: 0.000
Z rotation: 30.000
Rotation order: XYZ

Figure 43: Appearance of the *Add part* dialog for an STL part.



[✓] Origin at center of mass
[✓] Lock scaling ratio
XOrigin: 0.000E+00
YOrigin: 0.000E+00
ZOrigin: 0.000E+00
XScale: 1.000E+00
YScale: 1.000E+00
ZScale: 1.000E+00
Output file prefix: OutPrefix

Figure 44: Dialog to transform STL files.

$$x' = x_o + S_x(x - x_o), \tag{1}$$
$$y' = y_o + S_y(y - y_o), \tag{2}$$
$$z' = z_o + S_z(z - z_o). \tag{3}$$

When the *Center of mass* box is checked, the scaling operation maintains the object centroid. In this case, there are several possibilities for using the tool:

- If the *Lock scaling ratio* box is checked and $S_x = 0.5$, the object size is reduced by a factor of two.
- With the *Lock scaling ratio* box checked, use $S_x = 2.54$ to convert object dimensions from inches to centimeters.
- If the *Lock scaling ratio* box is unchecked and $S_x = 1.0, S_y = 1.0$ and $S_z = 0.5$, the height of the object is reduced by a factor of 2.0 while the other dimensions are unchanged.

When the *Center of mass* box is unchecked, you can use the scaling tool to change the size of the object while maintaining a base. For example, suppose the original object extends from $z = -5.00$ to $z = 16.75$. The goal is to reduce the size of the object so that it extends from $z = -5.00$ to $z = 5.00$ while maintaining its proportions. Use the following settings: *Center of mass* = unchecked, $x_o = 0.00, y_o = 0.00, z_o = -5.00$, *Lock scaling ratio* = checked, $S_x = 0.597$.

CONVERT STL TO BINARY

The STL text format includes a large amount of unnecessary information. The format is supported mainly for programs that do not conform to the IEEE floating-point number standard. Binary files are smaller and load much faster in **Geometer** and **MetaMesh**. Use this utility for file conversion. If you pick a file `FPREFIX.STL`, the resulting file is saved in the working directory with the name `FPREFIXB.STL`.

SHIFT STL MODEL

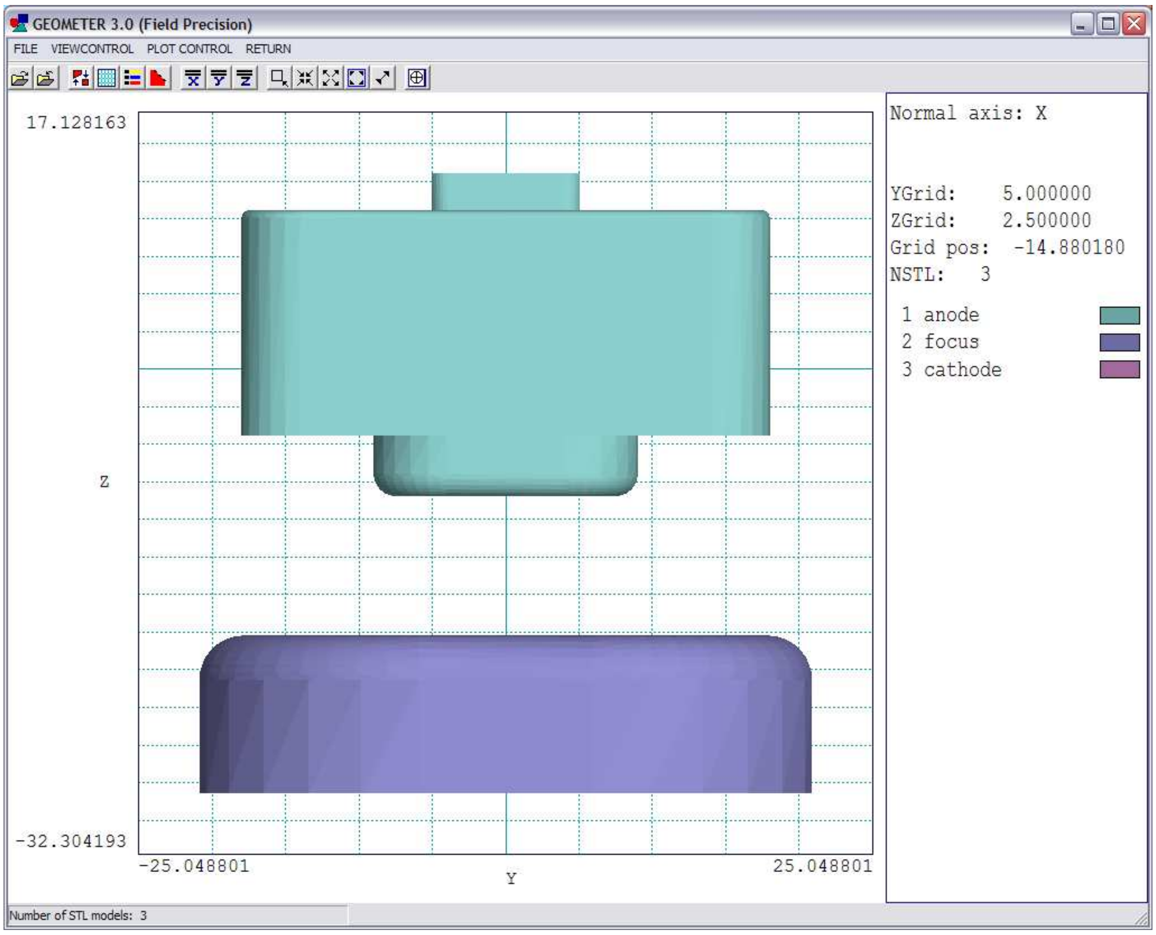
Use this tool to apply a global translation to nodes in a model. In the dialog, specify shifts in x, y and z. The values are simply added to the existing coordinates. The output file is saved in the working directory in the same format as the original with the name `FPREFIX.STL` (where *FPrefix* is the supplied file prefix).

10.5 STL Viewer

To run the three-dimensional view utility, click on *STL Viewer* in the main **Geometer** menu. Before running the viewer, **Geometer** closes any open scripts and loaded `STL` models, so be sure to save your work first. The viewer menu and toolbox and a blank screen appear. The first activity is to load one or more `STL` models.

ADD MODEL

Pick a file `FileName.STL` in the current or a different directory. The program automatically


GEOMETER 3.0 (Field Precision)
FILE VIEWCONTROL PLOT CONTROL RETURN
17.128163
Z
-32.304193
-25.048801
25.048801
Y
Number of STL models: 3

```
Normal axis: X

YGrid:      5.000000
ZGrid:      2.500000
Grid pos:  -14.880180
NSTL:    3
 1 anode
 2 focus
 3 cathode
```

Figure 45: Screenshot of the **Geometer** *STL View* utility, showing a solid view of STL objects, orthographic mode.

detects binary or text format and loads the model. If information display is active (see the *Show information on load* command), the viewer shows model parameters. Click *OK* to continue. The program creates a default orthogonal-style plot normal to the z-axis (Fig. 45). The program picks initial plot limits to encompass all objects and sets convenient grid intervals. The window on the right displays information on the plot and loaded models. The entry *Grid pos* is the position of the grid along the normal axis. The grid is treated as a plot object and may be moved along the normal axis to be in front or in back of other objects.

There are two other commands in the *File* menu:

CLEAR MODELS

Clear all loaded models and display a blank screen.

SHOW FILE INFORMATION ON LOAD

When the toggle is active, Geometer shows a dialog listing the file format, the number of facets, spatial limits of nodes and the center of mass position for each model loaded.

The following commands appear in the View control menu:

ORTHOGONAL/PERSPECTIVE

Switch between the *Orthogonal* mode (Fig. 45) and the *Perspective* mode (Fig. 41). Use the orthogonal mode to make measurements. Here, the three-dimensional objects may be viewed from the $+x$, $+y$ or $+z$ directions with the object to add grid lines. There are several zoom and pan options to get a close-up view. The perspective mode is an interactive environment where you can use the mouse to walk around the objects. The cursor modes and mouse controls for the perspective mode are the same as those described in Chapter 4.

X NORMAL AXIS, Y NORMAL AXIS, Z NORMAL AXIS

Change to a view from the $+x$, $+y$ or $+z$ direction. This command functions in both the orthogonal and perspective modes.

ZOOM WINDOW

In the orthogonal mode, use the mouse to specify a magnified view window by left-clicking at the opposite corners of a box in the plot area. The returned coordinates depends on whether *Snap mode* is active.

MAGNIFY VIEW,EXPAND VIEW

In the orthogonal mode, magnify or expand the view about the current plot center.

GLOBAL VIEW

In the orthogonal mode, expand the plot boundaries to encompass all loaded models.

PAN VIEW

In the orthogonal mode, move the current plot center. Left-click to define a starting and ending point for the translation.

FIND COORDINATES

In the orthogonal mode, find a position in the normal plane. Click on the command and move the mouse cursor into the plot area. The cursor changes to a cross-hair symbol and the corresponding coordinates are displayed in the status bar. Press *Esc* or click the right mouse button to leave coordinate mode and to reactivate the other menu commands. Note that snap mode is turned off while this command is active.

The following commands appear in the View control menu:

MODEL DISPLAY

The command displays the dialog of Fig. 24 to control the display of loaded models. You can choose between Solid and Wireframe plots styles or suppress the display of selected models.

GRID CONTROL

This command controls the display of the grid in the orthogonal mode. Uncheck the Display grid box to remove the display. When the Automatic grid box is checked, Geometer chooses good intervals. To set horizontal and vertical intervals manually, uncheck the box and supply real-number values. Finally, you can change the position of the grid along the normal axis so that it is behind or in front of models.

CLIPPING

The command displays the dialog of Fig. 26 to control the clipping of selected models. The feature is useful for displaying models enclosed by other models. There are six clipping planes for each model, each with a specified position: XDn, XUp, YDn, YUp, ZDn and ZUp. The default positions encompass the full volumes of all models. Clipping applies when the *Clipping planes active* box is checked. Use the Full model volume button to return the limits to the default.

TOGGLE SNAP MODE

This command affects mouse coordinates for Zoom window and Pan commands in the orthogonal mode. It controls whether the mouse returns the coordinates at the point closest to the cursor or at snap points set to convenient intervals. Snap mode is deactivated in the default mode.

SET SNAP LENGTH

Set the length of the snap interval in the horizontal and vertical directions. Enter a real number.

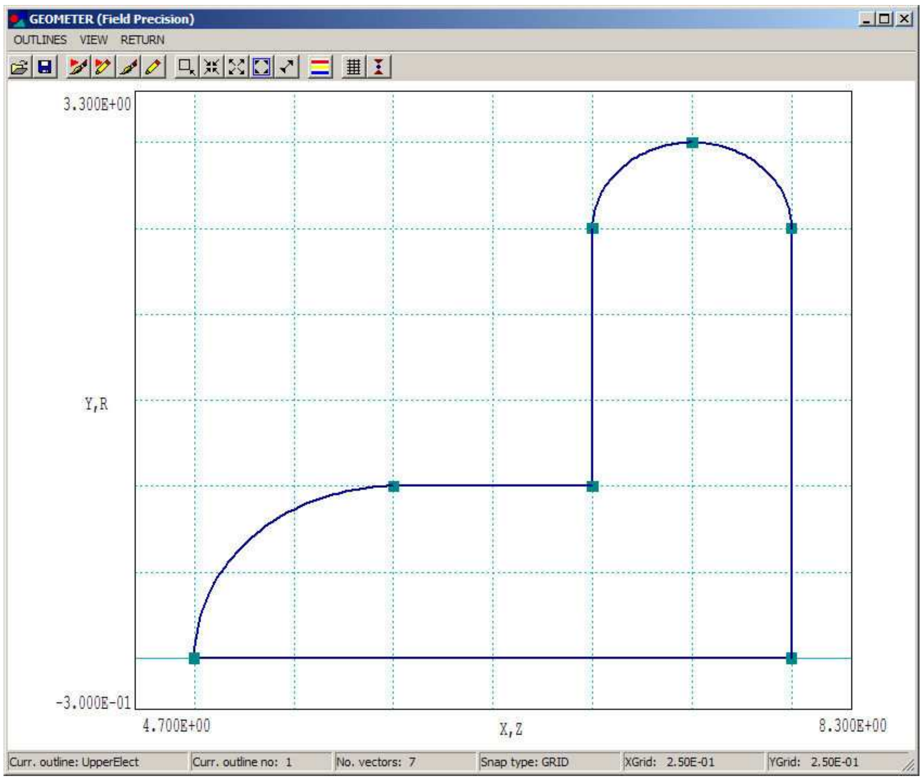


Figure 46: Screen display in the outline menu

11 Geometer – outline editor

We saw in the previous chapter that the cross sections of extrusions and turnings are specified by *outlines*. Here, an outline is a collection of contiguous line and arc vectors that define a closed shape. This chapter describes the extensive features for working with outlines in **Geometer**. They make the process of creating or modifying outlines relatively quick and easy.

11.1 Outline menu

To enter the outline menu, click the *Outline* command in the main menu. Note that this command is always active so you can work on outlines independently of a script. The plot screen is blank if there are available outlines loaded in the program. There are three ways to add outlines:

- Load an outline file (`FPREFIX.OTL`)
- Load a **MetaMesh** script that contains extrusions, turnings or transitions. All outlines in the script are automatically loaded into the program.
- Create a new outline using the text or graphics editors in the *Outline* menu.

Any of the outlines are available to build new parts or to apply to existing parts. There is no problem if there are extra available outlines in the program – only those associated with parts will be saved in response to the *Save script* command.

Figure 46 shows the screen display when one or outlines have been loaded. The *current outline* is plotted. By default, the current outline is the last one loaded into the program or the last one that has been edited. You can change the current outline with the *Displayed outline* command. The status bar at the bottom of the screen shows the name of the current outline and its order in the table of available outlines. The status bar also shows the number of vectors in the current outline, the mouse snap mode and the horizontal and vertical snap distances. The *Outlines* pulldown menu contains the following commands:

LOAD OUTLINE

Load an outline contained in a text file with a name of the form `FPREFIX.OTL`. The file contains up to 100 vector lines and a termination line consisting of the string *End*:

```
Vector01
Vector02
...
Vector03
END
```

The vectors must define a closed figure in x-y or z-r space[4]. The file may also contain blanks lines and comment lines that begin with an asterisk. A data line for a line vector starts with the letter L followed by four real numbers that give the vector starting and ending points and an optional fitting parameter:

```
L  XStart  YStart  XEnd  YEnd  [S]
L  ZStart  RStart  ZEnd  REnd  [S]
```

A data line for an arc vector starts with the letter A followed by six real numbers that give the starting, ending and center points:

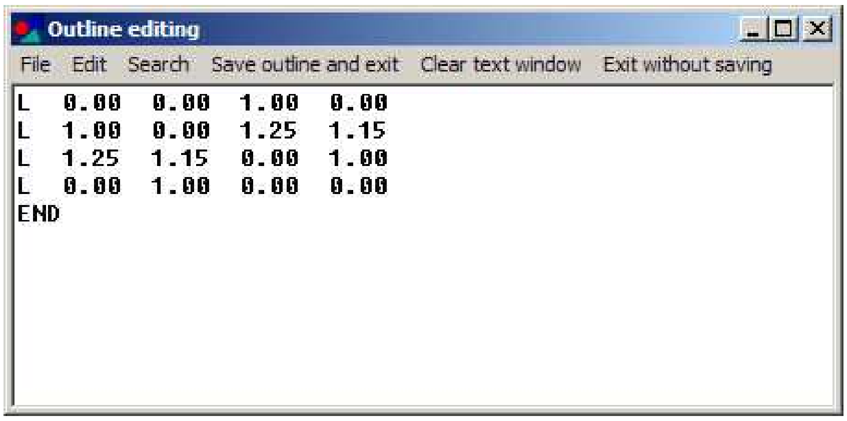
```
A  XStart  YStart  XEnd  YEnd  XCenter YCenter  [S]
A  ZStart  RStart  ZEnd  REnd  ZCenter RCenter  [S]
```

Geometer organizes and checks the vectors. The program will not load a set unless 1) the vectors form a contiguous set that outlines a closed figure and 2) arc coordinates are consistent (*i.e.*, starting and ending points are equidistant from the center point). After a successful load, the vector set is displayed on the screen.

SAVE OUTLINE

Geometer saves the current outline in a the standard format reviewed in the discussion of the *Load outline* command. An initial dialog lets you choose the outline that will be saved. The current outline is highlighted – just click *OK* or press *Enter* to save it. Supply a file prefix in

[4]Although the axes for extrusions are labeled x-y in the editor, the coordinates may apply to y-z or z-x, depending on the choice of the normal axis



```
L  0.00  0.00  1.00  0.00
L  1.00  0.00  1.25  1.15
L  1.25  1.15  0.00  1.00
L  0.00  1.00  0.00  0.00
END
```

Figure 47: Outline text editor

the second dialog or click *OK* to use the default. The file will be saved with a name of the form `FPREFIX.OTL`. Active outlines are retained in the program if you load a script or start a new one. Use the *Clear all outlines* command to start afresh.

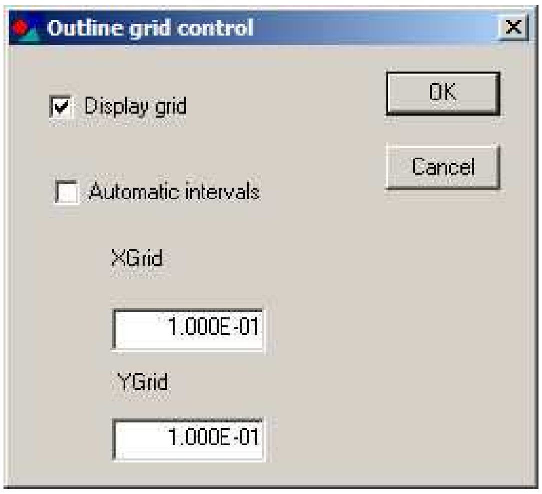
NEW OUTLINE/GRAPHICS
Enter the outline graphical editor to create a new outline – the next section gives a detailed discussion.

NEW OUTLINE/TEXT
This command brings up the editor window shown in Fig. 47. You can type or paste text representing a vector set into the window. The text editor is useful to create simple outlines, to check or to make small changes in outlines created with the graphics editor, or to import outlines directly from **Mesh** or **MetaMesh** scripts. You can also fine-tune the fitting parameters *S* or *SE* for selected vectors. Note that all lines (including the last) must end with a *Return* character. The menu contains three special commands: *Save and exit*, *Clear text window* and *Exit without saving*. If you choose the first command, **Geometer** checks the syntax of the data lines and the continuity of the vectors. If there are no errors, the program requests a name for the outline and then adds it to the list. The new outline is plotted on the screen.

EDIT OUTLINE/GRAPHICS
Load an existing outline into the graphical editor for inspection or modification – the next section gives a detailed discussion.

EDIT OUTLINE/TEXT
Load an existing outline into the text editor.

RENAME OUTLINE
Change the name of an existing outline.


Outline grid control

[✓] Display grid

[ ] Automatic intervals

XGrid: 1.000E-01

YGrid: 1.000E-01

Figure 48: Dialog to set grid properties in the *Outline* menu

REMOVE OUTLINE
After you pick an outline from the dialog menu, **Geometer** deletes it and reorganizes the list.

CLEAR ALL OUTLINES
Delete all active outlines.

ZOOM WINDOW
ZOOM IN
ZOOM LIMITS
EXPAND VIEW
PAN
The functions of these commands are similar to the corresponding commands for orthographic views in the **Geometer** main menu (Chap. 6).

DISPLAYED OUTLINE
Change the displayed outline by choosing from a dialog menu. The displayed outline becomes the current outline.

GRID CONTROL
This command brings up the dialog shown in Fig. 48. The settings in the dialog control the *display* and the *snap* grids. The display grid is the set of lines in the plot of Fig. 46. The intervals are listed in parentheses next to the axis labels. In the automatic mode, the visible grid intervals may change as the view zooms in and out. In this case **Geometer** picks intervals that are easily visible on the screen. The snap grid determines coordinates returned by the mouse. In the automatic mode, the program makes a good choice for snap intervals based on the dimensions of the current outline. To set intervals manually, deactivate the *Automatic* box and supply real-number values for the horizontal and vertical intervals. In this case, both the

display and snap grids are locked to the values you supply. You can suppress the visual grid by unchecking *Display grid*. This setting does not affect the snap grid.

SNAP CONTROL

This command brings up a dialog to set the snap type. The options are *None* (snap mode turned off), *Grid* and *Endpoint*. Under the *Grid* option the mouse snaps to values determined by the intervals listed in the status bar. In the *Endpoint* mode, the mouse moves to the closest vector endpoint. This mode is useful mainly in the graphics editor discussed in the next section.

RETURN

Return to the main menu. The outline set will be preserved unless you start or load a new script. You can return later to create or to update outlines.

11.2 Graphical creation and editing of outlines

Geometer enters the graphical editing environment in response to the following two commands:

NEW OUTLINE/GRAPHICS

Create a new outline using the two-dimensional CAD editor. The initial dialog asks for a name to apply to the outline and a set of dimensions to define a box in (x-y) or (z-r) space. The dimensions should span the outline. The size of the box will determine the initial screen display and settings for the snap grid. The numbers are not critical. You can change the display size and snap settings later. After you press *OK*, the program creates a blank plot. The tan borders indicate that you are in outline edit mode (Fig. 49). **Geometer** also loads a special menu and toolbar.

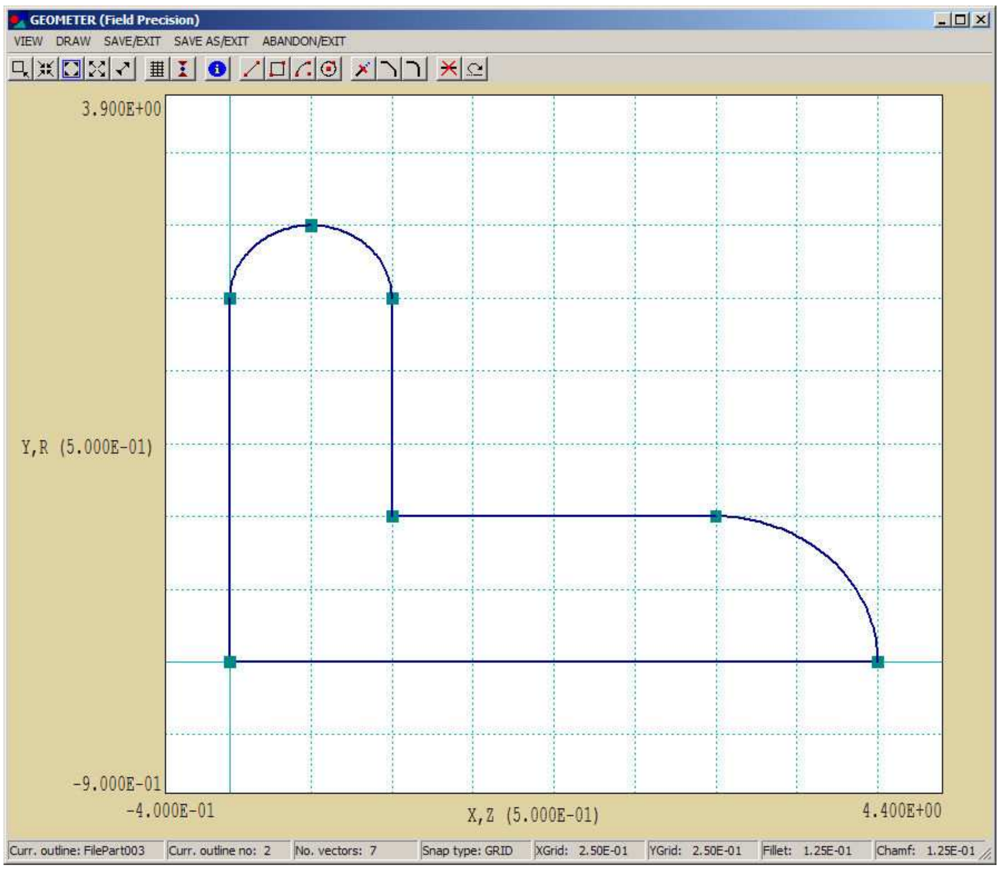
EDIT OUTLINE/GRAPHICS

The initial dialog prompts you to choose an outline. Simply click *OK* or press *Enter* to work on the current outline. The program loads the chosen outline into the graphics editor.

In the graphics editor, the following commands perform the same function as those in the standard outline menu:

ZOOM WINDOW
ZOOM IN
ZOOM LIMITS
EXPAND VIEW
PAN
GRID CONTROL
SNAP CONTROL

New commands appear in the *Draw* popup menu. The first set of commands adds vectors to the outline. In response to the *Line*, *Arc* and *Delete vector* commands, **Geometer** enters the *repeat mode*. Here, the command repeats automatically until you press the *Esc* key or the right


GEOMETER (Field Precision)
VIEW DRAW SAVE/EXIT SAVE AS/EXIT ABANDON/EXIT
3.900E+00
Y,R (5.000E-01)
-9.000E-01
-4.000E-01
X,Z (5.000E-01)
4.400E+00

Curr. outline: FilePart003 | Curr. outline no: 2 | No. vectors: 7 | Snap type: GRID | XGrid: 2.50E-01 | YGrid: 2.50E-01 | Fillet: 1.25E-01 | Chamf: 1.25E-01

Figure 49: Screen display in the outline graphics editor

mouse button. The menu and toolbar are deactivated when **Geometer** is in repeat mode. As an example, the mode is useful if you want to enter multiple lines to outline a boundary. To enter a different type of object, exit the repeat mode and click the appropriate tool or menu entry. It is not necessary to enter vectors of an outline in a particular order – **Geometer** automatically arranges vectors to form a continuous set when you save the outline. Insert commands use mouse input of coordinates. When the mouse is active, the cursor changes to a cross-hair pattern and coordinate values are displayed in the status bar. The values depend on the current setting of the snap mode. You can change the methods for coordinate entry when the mouse is active. To enter coordinates from the keyboard, press the *F1* key. To change the snap mode, press the *F3* key. In the following list, the symbol *[T]* indicates that the command has an entry on the toolbar.

INSERT LINE [T]

Create one or more line vectors by specifying the start and end points with the mouse or keyboard. To create multiple lines, simply continue to click on start and end points. The status window gives brief instructions. When you are finished, press *Esc* or the right mouse button to exit the repeat mode and to reactivate the menu and toolbar.

INSERT RECTANGLE [T]

Enter two points to define a box. The program creates four vectors to outline the rectangle.

INSERT ARC (START-END-CENTER) [T]
INSERT ARC (START-END-RADIUS)

There are two ways to define arcs: start-end-center and start-end-radius. In the first option enter the start-point, end-point and center-point of the arc with the mouse or keyboard. In the second option enter the start and end points and then type a value of the radius in the dialog. Make sure that arcs are less than 180^o so that there is no ambiguity about their direction. Divide longer arcs into two or more parts. Note that the arcs are entered in the repeat mode.

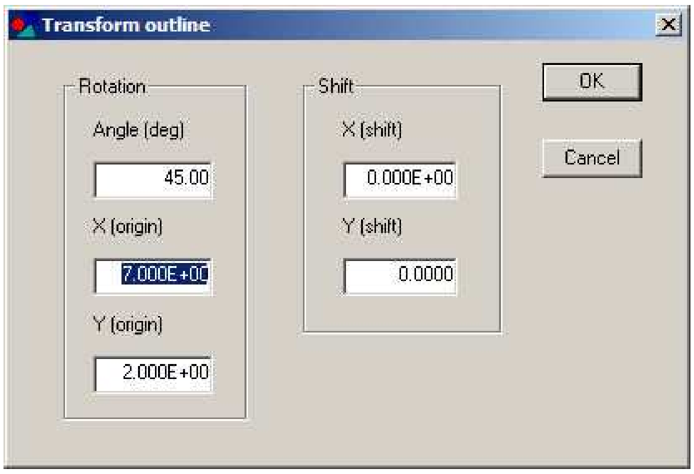
INSERT CIRCLE (CENTER POINT) [T]
INSERT CIRCLE (CENTER-RADIUS)

Create a circle using one of the following methods: center-point or center-radius. In the first option, specify the center point and a point on the radius with the mouse or keyboard. In the second option, specify the center point with the mouse and then type the value of the radius in the dialog. **Geometer** creates four 90^o arc vectors.

The next set of commands is used to modify existing vectors:

TRIM VECTOR [T]

Use the mouse to pick a trimming line vector. **Geometer** confirms the choice by highlighting the line. Then choose a vector to be trimmed. Click near the end of the trimmed vector that should be eliminated. The program cuts the vector from the endpoint to the intersection point


Transform outline

Rotation

Angle (deg) 45.00

X (origin) 7.000E+00

Y (origin) 2.000E+00

Shift

X (shift) 0.000E+00

Y (shift) 0.0000

Figure 50: Dialog to shift or to rotate outlines

with the trim line. The routine reports an error if the trimming and trimmed vectors do not intersect.

FILLET [T]
A fillet is a radius between two intersecting planes of a part. Use the mouse to pick two line vectors that share an endpoint. Note that they need not intersect at a right angle. **Geometer** adds a new arc vector to represent the radius, trims the line vectors, and rearranges the vector list. Use the *Fillet/chamfer width* command to set the fillet radius. The program displays an error message if the operation is impossible with the current radius.

CHAMFER [T]
A chamfer is a bevel between two intersecting planes of a part. Use the mouse to pick two line vectors that share an endpoint. **Geometer** adds a new line vector to represent the bevel, trims the line vectors, and rearranges the vector list. Use the *Fillet/chamfer width* command to set the chamfer width. The program displays an error message if the operation is impossible with the current width.

FILLET/CHAMFER WIDTH
The command brings up a dialog to set the fillet radius or chamfer width. Note that the current values are displayed in the status bar.

DELETE VECTOR [T]
Use the mouse to pick one or more vectors in the repeat mode. Press *Esc* or the right mouse button when you are finished. **Geometer** eliminates the vectors and rearranges the vector list.

TRANSFORM OUTLINE
This operation brings up the dialog of Fig. 50. You can apply translations and/or rotations

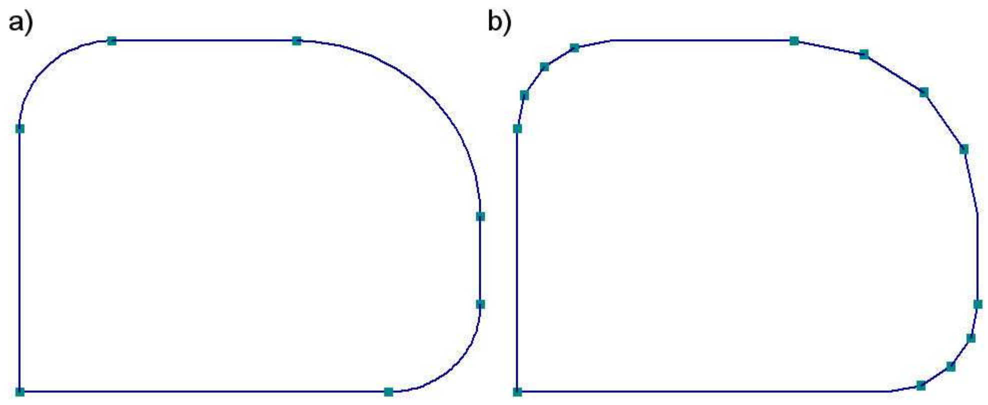


Figure 51: Effect of the *Convert arcs to lines* command with 22.5^o resolution. *a*) Original outline. *b*) After command applied.

about a specified origin to all vectors of the outline. As an example, you could load an existing outline, transform it, and then save it with a different name to create a periodic assembly of extrusions.

CONVERT ARCS TO LINES

Outlines that specify the upper and lower boundaries of transitions may contain only line vectors. If you use an outline that contains arcs in a transition, **Geometer** will replace the arcs with straight lines that connect the starting and ending points. Use this command to automatically convert arcs to a set of lines to give a better approximation. In the dialog, enter the approximate angular interval for lines in degrees. Note that you must ensure that the top and bottom outlines of the transition contain the same number of vectors. Figure 51 shows the effect of the command.

UNDO LAST

Reverse the previous insertion or editing command. Note that this command removes all lines and arcs entered in a repeat mode sequence.

To exit the graphical editing mode, use one of the following commands.

SAVE/EXIT
SAVE AS/EXIT
ABANDON/EXIT

In response to the first two commands, **Geometer** arranges the vector set and makes an extensive check of validity.

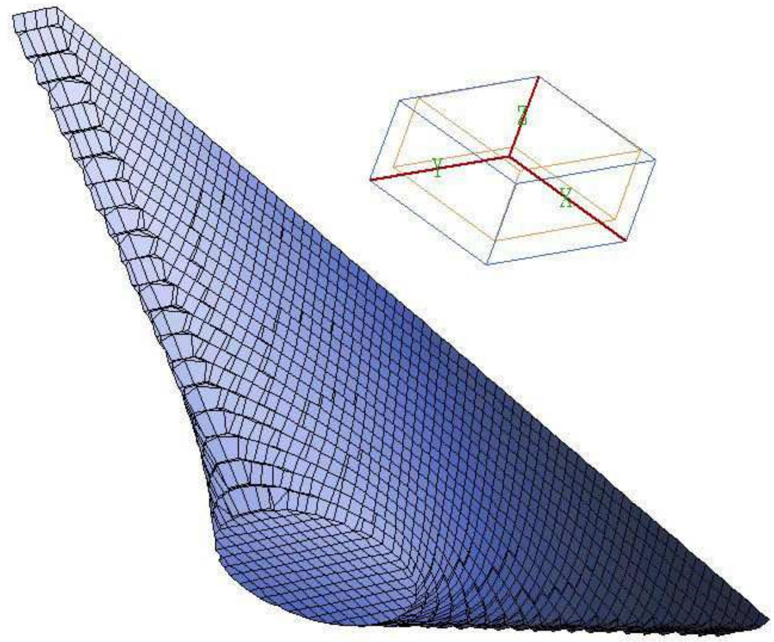


Figure 52: Transition example

12 Geometer – transitions and the neon model

12.1 Transitions

A *transition* is a shape that changes smoothly from one cross-section to another over a specified distance D. In the workbench frame, the cross-sections are defined by outlines in planes normal to the z axis at positions $z = -D/2$ and $z = +D/2$. To create a transition in **Geometer**, click *Edit/Add part* in the *Main* menu. Choose *Transition* from the *Part type* menu in the dialog. One of the fabrication boxes on the left-side will become active. Enter the distance D between the planes. Choose appropriate outlines for the top ($z = +D/2$) and bottom ($z = -D/2$) planes from the two active menus on the right-hand side. You can also add rotations and translations. When you click *OK*, the transition is added to the current plot. You can use the *Edit/Edit part* command to change the length of the transition or to assign the cross-sections to different outlines. Note that the part is automatically updated if you make changes to the outlines in the *Outline* menu.

At least two valid outlines must be available to create a transition. There are some important restrictions on the outlines used for transitions:

- They may contain only line vectors.
- They must have the same number of vectors.
- The vectors in each outline must be in the same order moving around the shape. **Geometer** and **MetaMesh** create the sides of the transition by connecting corresponding vectors in the upper and lower planes with a plane.

Geometer issues an error message if the outlines contain different numbers of vectors. You must ensure that vectors are in correct order. You can convert an outline with arcs to one that contains only lines using the *Arcs to lines* command in the *Outline* menu. In this case, you must keep track of the total number of vectors and ensure that the number is the same on the top and bottom planes.

12.2 Neon model

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 of a neon sign. The model can be used to create complex wiring, heater filaments, spiral coils and a variety of other shapes.

The shape of the neon tube is defined by a *path*. A path is a text list of up to 500 three-dimensional coordinates in the workbench frame. Each line of the list gives a point, and the list must terminate with the *End* command. The uniform-radius tube follows the coordinates in sequence.

A path has the format:

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

The set of points defines a set of $(N-1)$ line segments. A neon object has a single fabrication parameter, the uniform radius R of the tube. As an example, the path in Table 4 (constructed with a spreadsheet) defines the shape of the part in Fig. 53.

At least one path must be available to create a neon object in **Geometer**. Up to 16 paths may be loaded into **Geometer** in several ways:

- The paths for all included neon objects are added when you load a **MetaMesh** script with the *File/Load script* command in the *Main* menu.
- In the *Outline* menu, you can load path files. A path file contains a valid coordinate set and the *End* command – it has a name of the form `FPREFIX.PTH`.
- Using a text editor, copy and paste paths from **Mesh** or **MetaMesh** scripts.
- You can create new paths or modify existing ones using the path editor in the *Outline* menu.

The following commands in the *Outline* menu apply to paths:

LOAD PATH

Load a path contained in a text file with a name of the form `FPREFIX.PTH`. The file contains up to 256 coordinate lines and a termination line consisting of the string *End*. **Geometer** reports an error is a data line does not contain three real numbers.

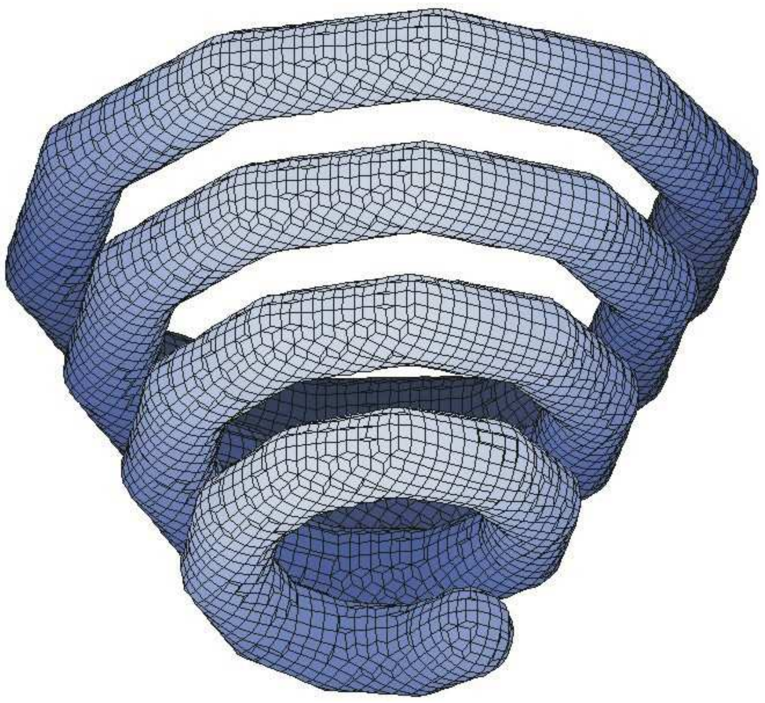


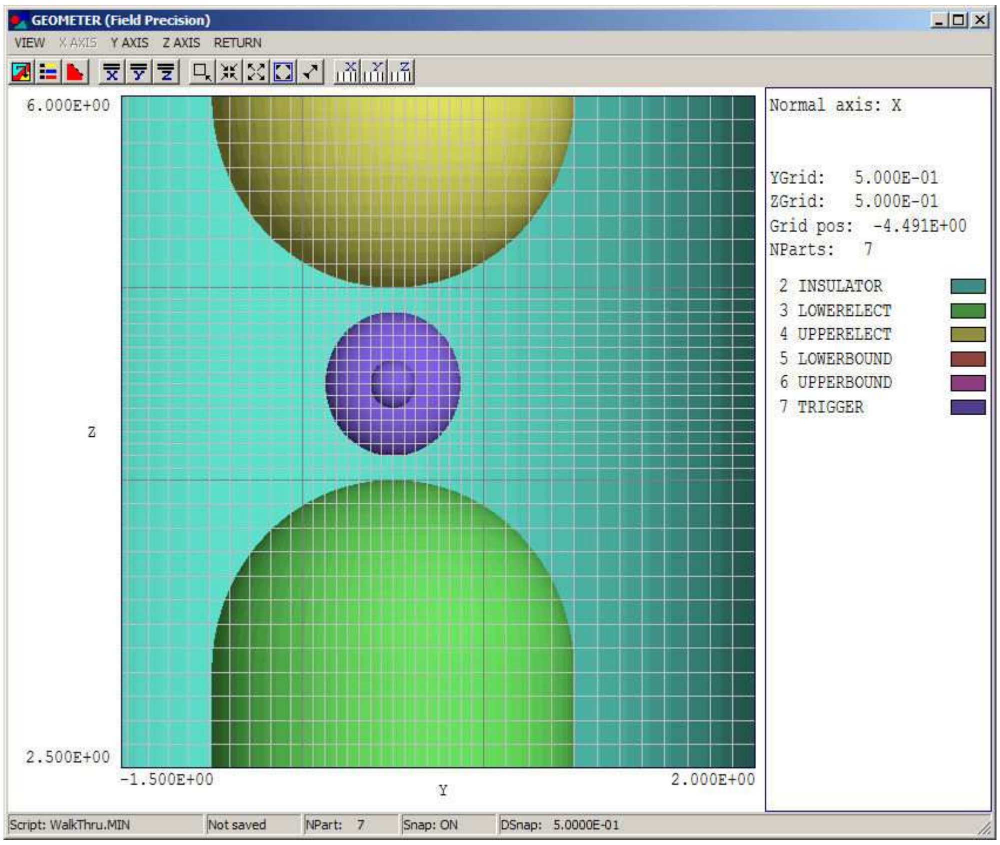
Figure 53: Shape created with the file `NEONDEMO.PTH`

EDIT PATH
Load an existing path into the text editor.

Assuming an appropriate path is available in **Geometer**, click *Edit/Add part* in the *Main* menu to add a neon object. Choose *Neon* from the *Part type* menu in the dialog. One of the fabrication boxes on the left-side will become active. Enter the radius *R* of the tube. Choose the appropriate path from the third the active box on the right-hand side. You can also add rotations and translations. When you click *OK*, the neon object is added to the current plot. You can use the *Edit/edit part* command to change the radius of the tube or to assign the part to a different path. The part is automatically updated if you make changes with the *Edit path* command in the *Outline* menu.

Table 4: Path coordinates to define the shape of Fig. 53

```
    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
```



```
Normal axis: X

YGrid:   5.000E-01
ZGrid:   5.000E-01
Grid pos:  -4.491E+00
NParts:   7

 2 INSULATOR     🟩
 3 LOWERELECT    🟩
 4 UPPERELECT    🟨
 5 LOWERBOUND    🟥
 6 UPPERBOUND    🟪
 7 TRIGGER       🟦
```

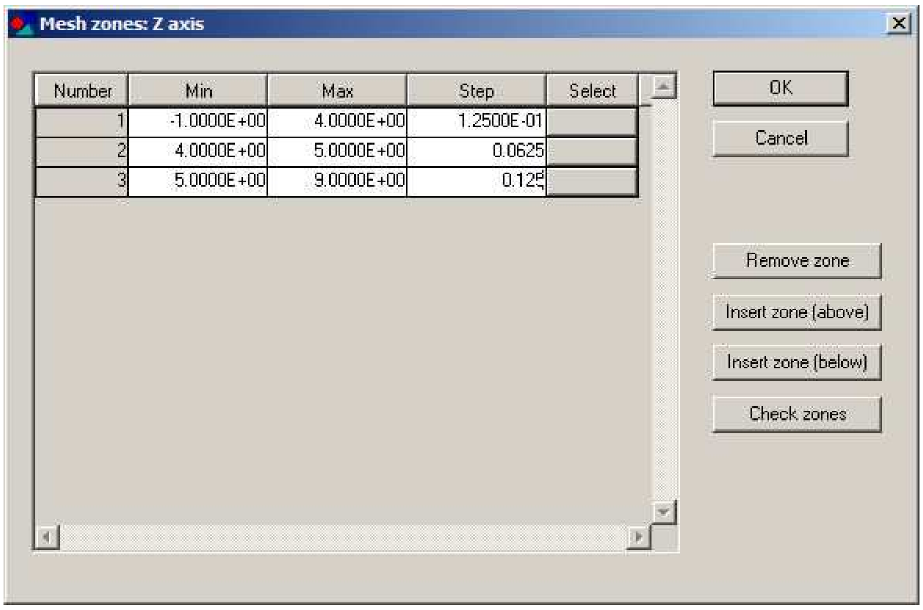
Figure 54: Working environment of the foundation menu

13 Geometer - defining the foundation mesh

The role of the foundation mesh and formats for its specification were discussed in Sect. 3.3. In review, the foundation mesh defines the *solution volume*, a box region in space. The foundation mesh consists of box elements, some of which will be reshaped to define conformal surfaces. You can set the sizes of foundation mesh elements to resolve the physical objects in the solution space by defining zones along the x, y and zx axes. A zone specification has three parameters: start point, end point and element size. Zones for an axis must form a contiguous set that covers the length of the axis. **Geometer** makes choosing zones easier by projecting the current foundation mesh on selected parts of the assembly. You can change zones interactively and immediately see the effect.

When you enter the *Foundation* menu, **Geometer** shows an orthographic plot of the assembly with the current foundation mesh superimposed (Fig. 54). The plot style follows that of the last orthographic plot created in the main menu. If you load an existing **MetaMesh** script, **Geometer** will read the zone specifications. If you start a new script, **Geometer** makes a default division of each axis into 80 uniform elements.

The *View* popup menu of the *Foundation* menu contains the full set of commands to control the orthographic view discussed in Chap. 6. You can zoom in, move the viewpoint, and control



Number	Min	Max	Step	Select
1	-1.0000E+00	4.0000E+00	1.2500E-01	
2	4.0000E+00	5.0000E+00	0.0625	
3	5.0000E+00	9.0000E+00	0.125	

Figure 55: Dialog to modify foundation-mesh zones along an axis

the display of parts and/or regions. Commands unique to the *Foundation* menu are located in the *X axis*, *Y axis* and *Z axis* popup menus. Note that only menus corresponding to axes in the projection plane are active. To change the available axes, use the *X normal axis*, *Y normal axis* or *Z normal axis* commands in the *View* menu.

The popup menus for the active axes include the following commands:

EDIT ZONE LIST [T]

This command displays the zone list for the axis in the dialog of Fig. 55. To change the starting point, ending point or element size of an existing zone, click the mouse in the appropriate box and type in the new value. To delete a zone, press the button in the *Select* column to identify the zone and then press *Remove zone*. After the operation, you must correct values in the *Min* and *Max* columns so that the zones fill the length of the axis along the solution volume. To add a new zone, press the *Select* button for an existing zone and then press either *Insert zone above* or *Insert zone below*. The maximum number of zones along an axis is 25. Fill in values for the new zone and make any necessary corrections to the *Min* and *Max* values of existing zones. When you click *OK*, **Geometer** checks that the boundaries of the zones define a valid set. If the zones along the axis are consistent, the program exits the dialog and updates the display.

SHIFT ZONE BOUNDARY

Use this command to move the boundary of an existing zone interactively. When you choose the command, the program shifts to coordinate entry mode. When the mouse cursor is inside the plot window the shape changes to a cross-hair pattern and coordinate values are listed in the status bar. Move the mouse close to the target boundary and click the left button. To confirm the choice, **Geometer** plots the boundary in red. Move the mouse to a new position and click the left button. Alternatively, press the *F1* key and type the value in the box for the appropriate axis. The zones to left and right of the shifted boundary will maintain approximately the same element sizes. (Note: mouse positions for coordinate entry reflect the current snap mode. It may be necessary to turn snap mode *OFF* before using this command.)

CHANGE ELEMENT SIZE

With this command you can change the size of elements in an existing zone. The mouse enters coordinate mode. Move the cursor into the target zone and press the left button. The zone is highlighted in red to confirm the choice. Type the new element size in the dialog box and click *OK*. The display is updated to show the new conditions.

UNDO LAST

Restore the zone lists to their condition before the previous *Edit zone list*, *Shift zone boundary* or *Change element size* command.

14 MetaMesh – script organization and conventions

The **MetaMesh** 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 with **Geometer** or any editor. Similarly, scripts can be modified using **Geometer** or directly with a text editor. There are several reasons why we use text scripts as the fundamental input medium for **MetaMesh**:

- Nothing is hidden – you can easily access the information used to generate a mesh.
- Scripts are 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 text 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 a text 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 incorporated in a new simulation.

MetaMesh scripts have a name of the form `RUNNAME.MIN`, where the string *RunName* is a descriptive file prefix from 1 to 20 characters in length. Even if you usually work in **Geometer**, it is important to understand the **MetaMesh** script. Several tasks require you to make additions directly to the script, such as fitting and coating operations for parts. Also, you can take advantage of the powerful block operations discussed in Chap. 17.

Scripts contain 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 in commands 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** 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 main 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 sixteen commands may appear in the section:

```
XMESH
YMESH
ZMESH
AUTOCORRECT
AXISSMOOTH
FITTING
FORMAT
NGAUSS
PARALLEL
PHOTOINTERVAL
PRESMOOTH
REGNAME
SMOOTH
SOFTEN
STLMETHOD
VOLUMECHECK
```

Chapter 15 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 constructing of the foundation mesh.

The assembly file may contain up to 250 *Part* sections. 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
REGION
SHIFT
ROTATE
FAB
SURFACE
COAT
```

Within a single part section, the commands may appear in any order. Chapter 16 gives detailed information on commands of the *Part* sections. The *EndFile* command follows all *Part* sections and signifies the end of data entry. The program ignores lines after *EndFile*, so you can append any amount of text annotations 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 ten core commands: *Xmesh, Ymesh, Zmesh, Type, Region, Shift, Rotate, Fab, Surface* and *Coat*. Table 5 shows an example of a complete script.

Table 5: **MetaMesh** script example

```
GLOBAL
  XMesh
    -5.0 5.0 0.5
  End
  YMesh
    -5.0 5.0 0.5
  End
  ZMesh
    -10.0 10.0 0.5
  End
  RegName(1) = SolnVolumeLower
  RegName(2) = SolnVolumeUpper
  RegName(3) = Object
END
PART 1
  Type Box
  Region SolnVolumeUpper
  Fab 10.0 10.0 10.0
  Shift 0.00  0.00  -5.00
END
PART 2
  Type Box
  Region SolnVolumeLower
  Fab 10.0 10.0 10.0
  Shift 0.00  0.00  5.00
END
PART 3
  Type Ellipsoid
  Region Object
  Fab 3.0 3.0 6.0
  Shift 0.0 0.0 0.0
*  Surface Region SolnVolumeUpper
  Surface Region SolnVolumeLower
  Coat 1 4
  Coat 2 5
END
ENDFILE
```

15 MetaMesh script – global commands

15.1 Building the foundation mesh

Because **MetaMesh** creates structured meshes, the solution space must enclose a box. Following Sect. 3.3, 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 initially set as a grounded metal cube, while the active solution space is an enclosed spherical dielectric region. 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 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. (We strongly recommend using appropriate units such that coordinate values are in the range 10^{-4} to 10^{+4} to avoid real-number round-off errors.) All **AMaze** solution programs have provisions for interpreting the coordinate units used in **MetaMesh**.

A script must include one instance each of the *XMesh*, *YMesh* and *ZMesh* commands. The following example completely defines a foundation mesh:

```
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 56 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:

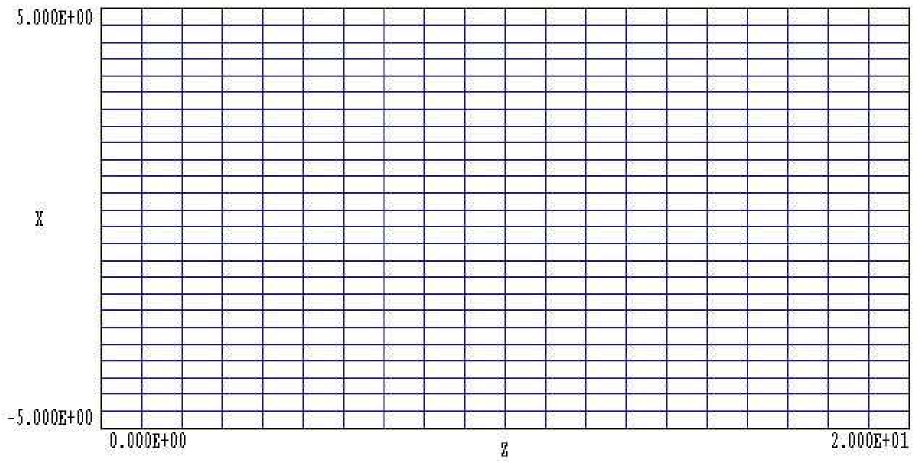


Figure 56: Projection of foundation mesh normal to the y axis – uniform element sizes along x and z.

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

The string parameter *N* signals that the following number will be 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 0.25
End
```

The two 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
```

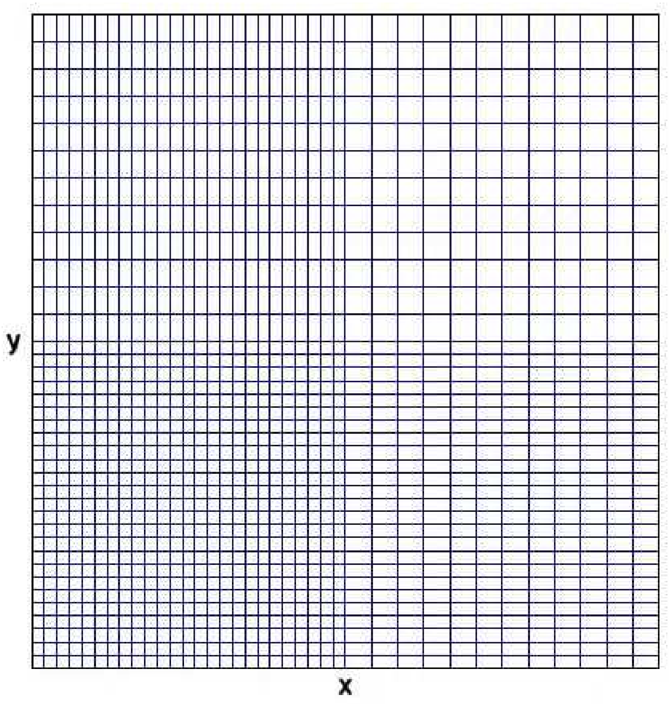


Figure 57: Variable-resolution mesh.

```
  -5.00  0.00  0.20
   0.00  5.00  0.40
End
```

create the mesh (normal to the z axis) shown in Fig. 57.

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. Errors may occur if you try to represent a complex part that has 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.* non-conformal box elements). Although small details may be lost, the mesh is always valid and there are no numerical 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 $x_{min} = xmin1$

to $x_{max} = xmaxn$. Within the region from $xmin1$ to $xmax1$, the approximate element size is $dx1$, 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
xmin1 xmax1 N N1
xmin2 xmax2 N N2
...
xminn xmaxn N Nn
END
Alternate form of the data lines of the *XMesh* structure. The string *N* designates that the following integer gives the number of elements in the interval. Note that both data line formats can be mixed in a single *XMesh* set.

15.2 Symbolic region designations

The *RegName* command is useful to document your runs and to organize your work within **MetaMesh**. The names are transferred to **Geometer** when you load a script.

REGNAME RegNo String
The parameter *RegNo* is the region number, an integer from 1 to 250. The *String* is a descriptive word to associate with the region number. String may contain up to 20 characters with no delimiters. In commands of the PART sections, you can refer to a region either by the number or by the name.

15.3 Advanced commands

The remaining set of allow detailed control of program functions.

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 (*e.g.*, region boundaries). The default value $NSmooth = 0$ gives no smoothing. Larger values give more smoothing.

SOFTEN NSoft
Smooth all nodes of the mesh, including those that have been clamped by fitting operations. The default value $NSoft = 0$ gives no smoothing. Larger values give more smoothing. This command may be useful to improve the appearance of meshes created from images (Chap. 20).

AUTOCORRECT [ON, OFF] NCorrect
AUTOCORRECT ON 9
This command controls automatic correction of meshes by the relaxing the position of nodes connected to deformed elements. The code detects distorted elements by looking for negative values in a calculation of the surface area. The node positions of distorted elements are relaxed in to ensure valid shapes. The process may result in smoothed transitions on the edges of parts. 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 *NCorrect* by supplying an integer parameter after the string parameter *ON*. During the testing process, **MetaMesh** keeps a running total of elements facet areas on the borders between regions. The program uses the information to create a useful list of region surface areas in the file `RUNNAME.MLS`. You can use the values to check the accuracy for representing shapes with known areas or to find the approximate areas of complex shapes. (Default: *ON*, $NCorrect = 7$.)

VOLUMECHECK [OFF, ON]
VOLUMECHECK ON
After generating and smoothing a conformal mesh, **MetaMesh** can crosscheck the integrity of elements by performing element volume integrals and looking for negative values. This calculation is time-consuming and is generally not as sensitive as the surface-area test. Therefore, volume checking is turned *OFF* by default. check. Activate volume integrals to make a more stringent check of a new mesh geometries or to create a list of region volumes in the file `RUNNAME.MLS`. As an example, suppose in an electrostatic calculation that you wanted to place a uniform charge density ρ in a region that sums to a total charge Q. Find the volume V of the region and take $\rho = Q/V$. Default: *OFF*)

FITTING [ON, OFF]
FITTING OFF
When building a new 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 individual parts. (Default: *ON*)

FORMAT [TEXT, BINARY]
FORMAT TEXT
This command controls the format of the output file. The string parameter options are *Text* and *Binary*. The *Binary* format is more compact and it is required to transfer mesh information to the **AMaze** solution programs. Use *Text* if you want to port mesh information to your own applications. Chapter 25 describes the output formats under both options. (Default: *Binary*)

STLMETHOD [LINE, PROXIMITY]
STLMETHOD = PROXIMITY
This command sets the method used to determine whether an element is inside an STL object (Sect. 18.1). The default *Line* method analyzes valid STL files reliably and quickly. For invalid STL files (with surface gaps, overlapping facets, zero-area triangles,...), it is sometimes possible to obtain satisfactory results with the *Proximity* method.

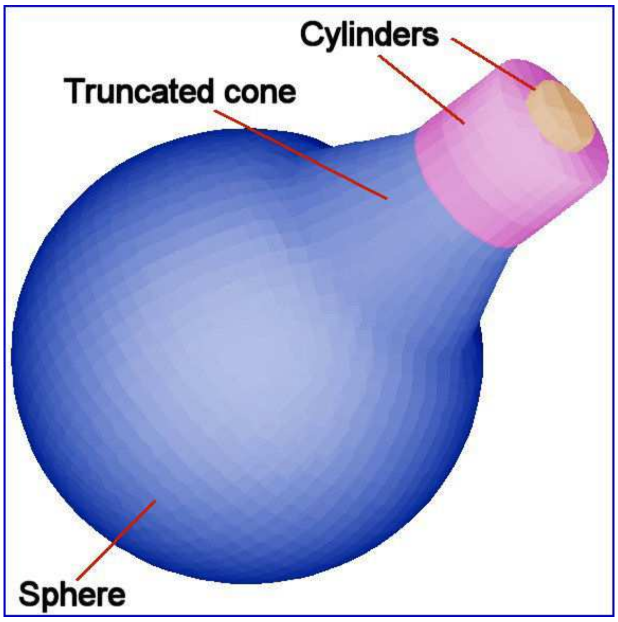


Figure 58: Complex shape (lightbulb) constructed from a geometric sum of simple shapes.

16 MetaMesh script – defining parts

16.1 Three-dimensional assembly procedure

Three-dimensional objects (*regions*) are built in **MetaMesh** by combining simple volumes called *parts*. Figure 58 shows an example where we construct a bulb for a bathroom light fixture from a geometric sum of simple shapes. This chapter covers the logic and syntax of the **MetaMesh** script for constructions using simple parts. Advanced models are discussed in Chap. 18.

The **MetaMesh** program and script format syntax was designed so that the mesh generation process 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 a reference coordinate system where we build parts in 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.

The script must contain 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, a rod electrode with a radius on the end may be constructed from two parts:

a cylinder and a sphere. The region number is equivalent 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, so **MetaMesh** uses different routines to process different part types.
- **Dimensions** One or more real numbers that give the size and shape of the part. The number and interpretation of fabrication parameters 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 define the orientation in the solution space. Optionally, you can change the order in which the rotations are applied.
- **Position** Three displacements (real numbers) along the axes that tell where to put the part after it is fabricated and oriented.

16.2 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 part specifications before starting processing operations. You can change the assembly order by changing the location of the part section in the script or the *Change part order* command in **Geometer**. Note that the part numbers of elements and nodes are used within **Geometer** and **MetaMesh**. Only region numbers are written to the `RUNNAME.MDF` file that is passed to the solution program. If you maintain a library of standard assemblies for your applications, you can paste them in at any location in the script. The capability is similar to blocks in a CAD program. If you define symbolic part numbers using the *Name* command, 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 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 connected nodes of the element.

The example in Table 6 illustrates implications of the rules. The two *Part* sections describe the ceramic insulator and the copper bus bar of a high-current vacuum feed-through. In the **HiPhi** solution, the copper rod is assigned a fixed potential. Because potential values are defined at nodes, we must ensure that all nodes connected to the elements of the rod have the rod region number. This condition occurs automatically if the insulator appears in the file

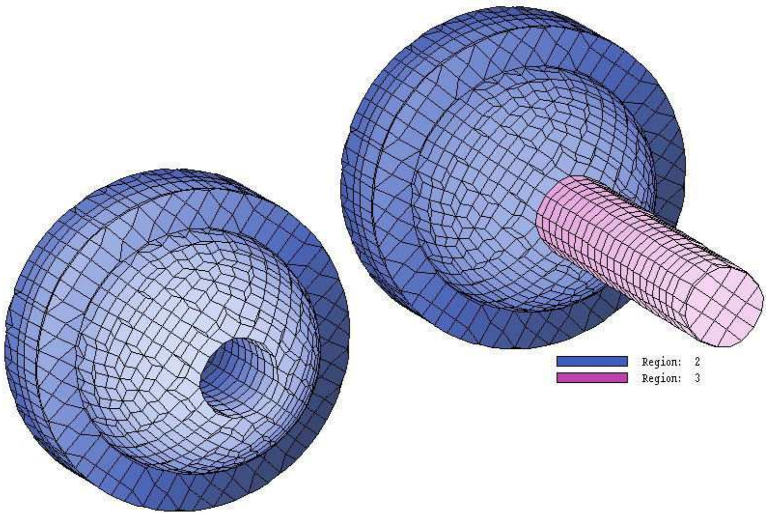


Figure 59: Example of part processing order. The rod (violet) has a higher value of *PartNo* than the insulator (blue). Processing of the rod creates a hole through the insulator.

before the rod. 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 59 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.

16.3 Relationship between workbench and assembly spaces

In **MetaMesh** parts are initially fabricated in the *workbench* space. The workbench space is a coordinate system with the same origin and axes as the coordinates of the assembly space. Depending on its type, a part has a specific orientation and position when created on the workbench. For example, spheres have their centers at the origin of workbench space $(x_w, y_w, z_w) = (0, 0, 0)$. A cylinder of height H has its axis coincident with the z_w axis and has end faces at $z_w = \pm H/2$. Chapter 8 lists conventions for simple parts. Once a part has been fabricated, we rotate it to the correct orientation on the workbench and then shift it to its location in assembly space.

While the shift operation is simple, three-dimensional rotations are often more difficult to envision. The *Rotate* command 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 60 shows two examples: $\theta_y = +30^o$ and $\theta_z = -45^o$. Rotation operations are not commutative; therefore, 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 $\theta_x = 90^o$ followed by $\theta_z = 90^o$ gives a vector pointing along the x axis, $\mathbf{u} = (1, 0, 0)$. On the other hand, a rotation of $\theta_z = 90^o$ followed by $\theta_x = 90^o$ gives a vector pointing in the $-y$ direction, $\mathbf{u} = (0, -1, 0)$. The default order of rotations in **MetaMesh** is $\theta_x \rightarrow \theta_y \rightarrow \theta_z$. It is possible to change the order to achieve

Table 6: Example to illustrate the importance of part order in the **MetaMesh** script

```
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 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 1.00
  Surface Part 2 1.00
End
```

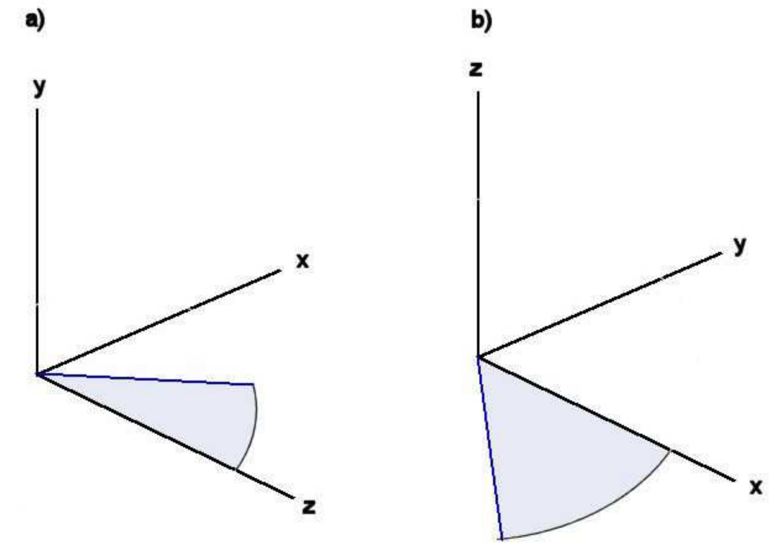


Figure 60: Three-dimensional rotations. *a*) $\theta_y = +30^o$. *b*) $\theta_z = -45^o$.

Table 7: Rotation and shift parameters for Fig. 61

Part	θ_x	θ_y	θ_z	S_x	S_y	S_z
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

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 of assembly space with 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) on the workbench; therefore, we need to apply both rotations and translations. Figure 61 shows the completed ornament with numbered cylinders and spheres. Table 7 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 *Tools* menu of **MetaMesh**. This useful utility takes starting and ending points for an arbitrary line in assembly space and determines values of displacement and rotation for an equal-length line parallel in workbench space. The workbench equivalent line is parallel to z with midpoint at the origin.

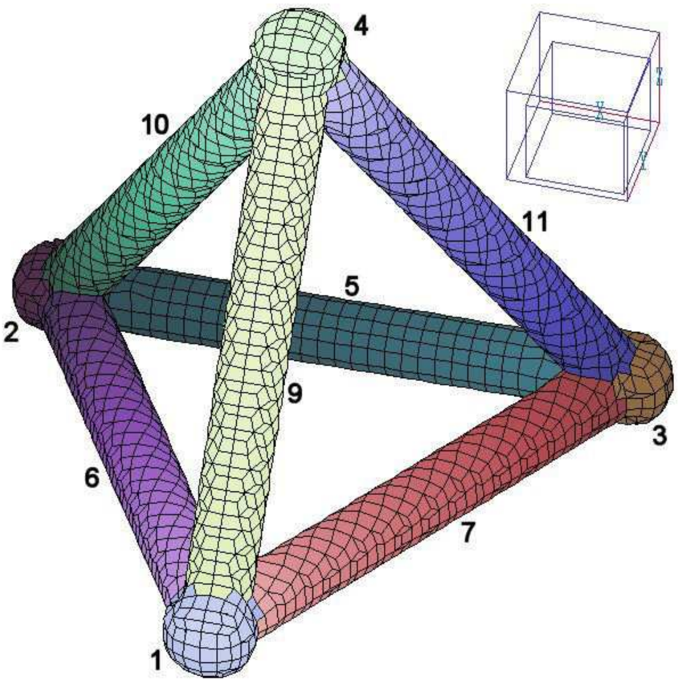


Figure 61: Illustration of shift and rotation operations

16.4 Volume fitting procedure

Part types in **MetaMesh** can be divided into two classes: *filled parts* and *open parts*. 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. Therefore, 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 goal is to provide a good estimate of the division of the foundation mesh to represent the set of parts in the script. The program makes an initial division of the foundation mesh, associating elements with the parts. It analyzes the parts in sequence according to their position in the script (*PartNo*). For each part, the program searches all elements of the foundation mesh over the surrounding volume. If the center 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 results in a regular mesh of box elements. Figure 62*a* illustrates the appearance of a regular mesh for the following part:

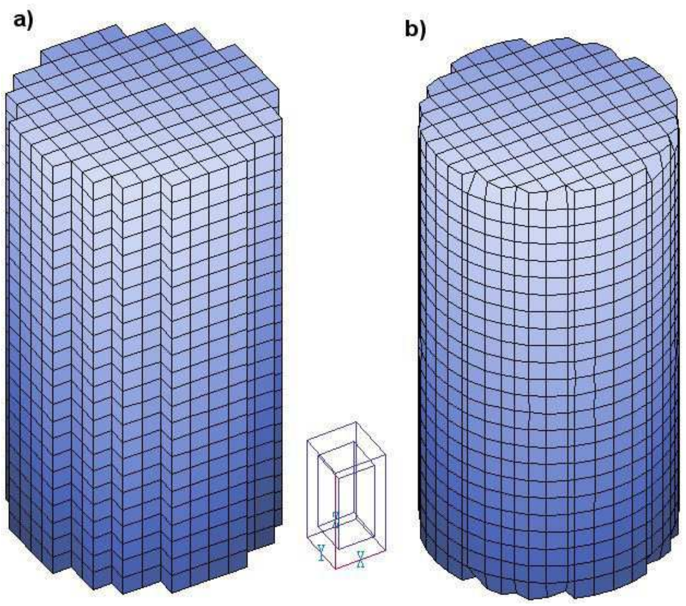


Figure 62: Representations of a cylindrical part. *a*) State after volume fitting. *b*) State after volume and surface fitting.

```
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
```

To generate the figure, 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.

16.5 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 interpolations in field solutions 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:

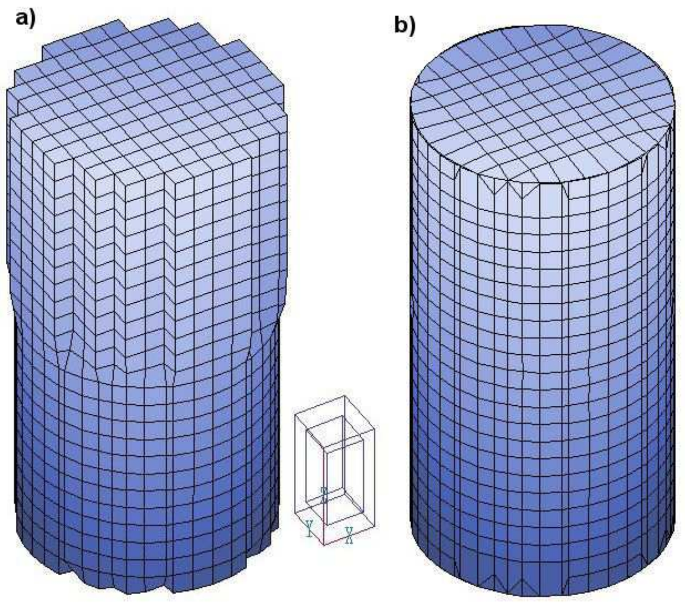


Figure 63: Fitting operations on a cylinder. *a*) The bottom half of the cylinder is surrounded by elements with $RegNo = 1$ and the top half by elements with $RegNo = 2$. Only the surface bounding $RegNo = 1$ has been fitted. *b*) Appearance of the cylinder with both surface and edge fitting.

SURFACE REGION RegNo
SURFACE PART PartNo

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 $\mathbf{x}$ represents the initial position of the node, then the new position $\mathbf{x}'$ 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 $\epsilon_s = 0.90$. A value of $\epsilon_s = 1.00$ corresponds to a perfect fit. This choice increases the possibility of element distortion and is seldom necessary. You can set individual values for the surface fitting tolerance with the following command form:

SURFACE REGION RegNo EpsiS
SURFACE PART PartNo 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 the tolerance $\epsilon_s = 1.00$ to fit nodes connected to facets on the border between elements with $PartNo = 8$ and $RegNo =$

Table 8: Surface and edge fitting accuracy (Cylinder with radius 3.0 and height 12.0)

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

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 62b shows the effect of surface fitting. In the example, the bottom half of the cylinder is surrounded by elements with $RegNo = 1$ and the top half by elements with $RegNo = 2$. Removing the comment symbols from 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. 63a. Only the boundary nodes in the bottom half of the solution space have been displaced. The ability to isolate portions of part boundaries that border specific regions is a powerful feature of **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. 63a and 63b). This effect is of little concern in electromagnetic or thermal field solutions because field values and thermal flux are undefined at surface discontinuities. With regard to the quality of the physical solution, there is nothing to be gained by adding sharp corners to objects in the mesh. On the other hand, there are a few instances (such as edge-emission of an electron emission surface) where the boundaries must be modeled exactly. Therefore, we have included 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. 63b by changing the two fitting statements in the example to

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

Table 8 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%.

IMPORTANT WARNING, BUT PEOPLE DO IT ANYWAY

Adding an *Edge* directive to every *Surface* command in an assembly to make the plots look better is the surest path to hours of frustration. Use edge fitting only where absolutely necessary.

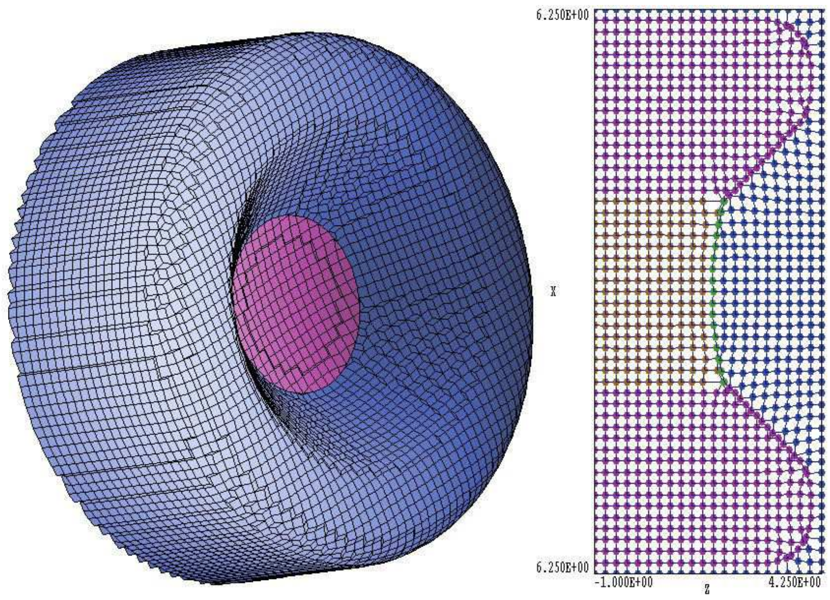


Figure 64: Example of coating operation – high-current cathode assembly. Three-dimensional view and slice through the mid-plane showing node identities. Note the coated region of nodes marked green.

With careful preanalysis of the simulation geometry and good choices of element sizes in the foundation mesh, **MetaMesh** runs automatically with few problems. 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.

16.6 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 (Sect. 3.5). The *Coat* command has the following format.

COAT REGNO REGNEW
COAT(5) = 7
The quantities *RegNo* and *RegNew* are integers. **MetaMesh** takes the following actions in

Table 9: Script extract illustrating the coating operation

```
Part 2
  Type Turning
* Focus electrode
     L    2.00E+00  2.00E+00  3.70E+00  3.70E+00  S
     A    3.70E+00  3.70E+00  4.00E+00  4.41E+00  3.00E+00  4.41E+00 S
     L    4.00E+00  4.41E+00  4.00E+00  5.00E+00 S
     A    4.00E+00  5.00E+00  3.00E+00  6.00E+00  3.00E+00  5.00E+00 S
     L    3.00E+00  6.00E+00 -1.00E+00  6.00E+00 S
     L   -1.00E+00  6.00E+00 -1.00E+00  2.00E+00
     L   -1.00E+00  2.00E+00  2.00E+00  2.00E+00
  End
  Region 2
  Surface Region 1 1.00
End
Part 3
  Type Turning
* Cathode support
     L   -1.00E+00  0.00E+00  1.71E+00  0.00E+00
     A    1.71E+00  1.77E-07  2.00E+00  2.00E+00  9.00E+00  0.00E+00 SE
     L    2.00E+00  2.00E+00 -1.00E+00  2.00E+00 SE
     L   -1.00E+00  2.00E+00 -1.00E+00  0.00E+00 SE
  End
  Region 3
  Surface Region 1 Edge 1.00
  Surface Region 2 Edge 1.00
  Coat 1 4
End
```

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 spherical-section surface.Table 9 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 64 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.

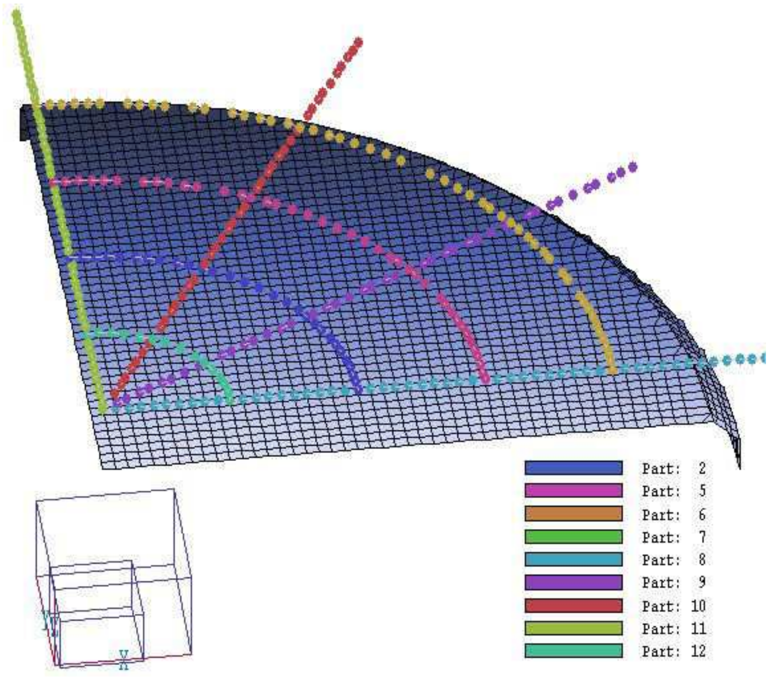


Figure 65: Cathode control grid created with the open part types *Line* and *Circle*.

16.7 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 10. 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 8.2 describes open part types in detail. Here we shall summarize the general rules for processing them.

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 to shift node positions.

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*

Table 10: Open part types

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

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 border 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 spans multiple zones of the foundation mesh. Figure 65 shows an example, a control grid above the surface of a beveled cathode ($RegNo = 2$). The grid was created with the commands listed in Table 10.

16.8 Summary of part section commands

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

TYPE PartType
TYPE = CYLINDER
Specify the geometric class of the part. The string parameter gives the part type, one of the choices listed Chaps. 8. Chapter 18 describes the *Type* structure for extrusions, turnings, transitions and the neon model.

REGION RegNo
REGION = 4
Associate the part with a region of the solution volume. The quantity *RegNo* is an integer from 1 to 250 or a symbolic name defined by the *Regname* command. Several parts may be used to construct a single region.

NAME NameString
NAME = EXPOXYPOT

Table 11: Selected commands to create the grid of Fig.65

```
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 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
...
```

Set up a symbolic reference to the assigned part number. The string may have length from 1 to 20 characters. Standard delimiters may not be included. The underscore symbol is allowed.

FAB Param1 Param2 ...
FAB 6.00 8.00
Set the size and shape of the part to be fabricated on the workbench. 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.

ROTATE ThetaX ThetaY ThetaZ [ROrder]
ROTATE 0.0 90.0 45.0
ROTATE 0.0 90.0 45.0 ZY
Orient the part on the workbench before moving it to the assembly space. 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 $\theta_x \rightarrow \theta_y \rightarrow \theta_z$. 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^o about the z axis and then by 90^o about the *new* y axis. The directions of positive rotation angles follow the right hand rule.

SHIFT Xs Ys Zs
SHIFT 0.00 0.00 3.56
Position the part in the assembly space. Shifts are performed after rotations. The three real number parameters are displacements along the x, y and z axes respectively.

SURFACE REGION RegNo [Edge] [EpsiS]
SURFACE REGION 5 1.00
SURFACE REGION 7 Edge 0.80
SURFACE REGION AirVolume
Fit nodes attached to specified surface facets of the part to model values to construct a conformal mesh. 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 *RegNo* is the region number (1-250). You may also use a string synonym defined with by *Regname* command. The optional keyword *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. Lower the value if a large number of element distortions occur during the construction process. 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. In this case, **MetaMesh** reports an error.

SURFACE PART PartNo [Edge] [EpsiS]
SURFACE PART 6 0.85
SURFACE PART 7 EDGE 0.95
SURFACE PART ENDPLATE
Fit specified surfaces of the part to ideal values to construct a conformal mesh. The difference from the *Surface region* command is that the included facets lie on the boundary with a specified part.

COAT RegNo RegNew
COAT 2 6 COAT AIRVOLUME LOWERELECTRODE
Change the region number of nodes connected to facets that border on region number *RegNo* to *RegNew*. The parameters *RegNo* and *RegNew* are integers in the range 1-250. 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*.

16.9 Symbolic part and region numbers

The direct use of region and part numbers in the *Part* section commands *Region*, *Surface* and *Coat* is sufficient 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 symbolic region and part numbers.

To illustrate the association of names with region numbers, consider the following statements:

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

The first statement defines the string *TransformerOi* 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 standard delimiters. 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** scripts to document the identity of regions.

The following is an example of a region list in the `MLS` file:

```
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 represents the fifth part to appear in a 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 in subsequent sections. For example, the following two statements (in a latter *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.

17 MetaMesh script directives and block operations

Script directives are special commands that may be included *between Part* sections to modify the operation of **MetaMesh**. Their main use is to shift or to 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.

17.1 Syntax in the MetaMesh script

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 a time.

The following directives control the position and orientation of a block 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 workbench coordinate system after any local rotations and shifts. Global rotations are performed before global translations. By default, rotations are performed in the order $XRot \rightarrow YRot \rightarrow ZRot$. 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 $RotOrder = ZY$ 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 or if there is a missing *#EndRotate* command.

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:

$$\text{LocalRot} \rightarrow \text{LocalDisp} \rightarrow \text{GlobalRot} \rightarrow \text{GlobalDisp}.$$

#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.

17.2 Script directives in Geometer

MetaMesh script directives are recognized when you load an `MIN` file into **Geometer**. Program operation is governed by the following rules:

- **MetaMesh** script directives may not be added or edited in **Geometer**.
- In the **Geometer** display, parts included in the `#SHIFT/#ENDSHIFT` are shifted as a block. Shifts of individual parts may be edited, but global shifts cannot be changed in **Geometer**. The shift script directives will be included in an `MIN` file created from **Geometer**.
- In the **Geometer** display, parts included in the `#ROTATE/#ENDROTATE` are rotated as a block. Rotations of individual parts may be edited in **Geometer**, but global rotations cannot be changed. The rotation directives will be included in an `MIN` file created from **Geometer**.
- All parts within the `#SKIP/#ENDSKIP` construct are ignored on input. The parts will not be included if you save an `MIN` file from **Geometer**.
- The `#NOFIT/#ENDNOFIT` directives do not affect the **Geometer** display. The rotation directives will be included in an `MIN` file created from **Geometer**.

17.3 Block operations example

The example `BLOCKDEMO` shows how to rotate and position several instances of a standard assembly. The full script is listed below. Figure 66 shows the resulting mesh.

```
* File: BlockDemo.MIN
GLOBAL
  XMesh
     -5.00  5.00  0.10
  End
  YMesh
     -5.00  5.00  0.10
  End
  ZMesh
      0.00  5.00  0.10
  End
  RegName(1): Air
```

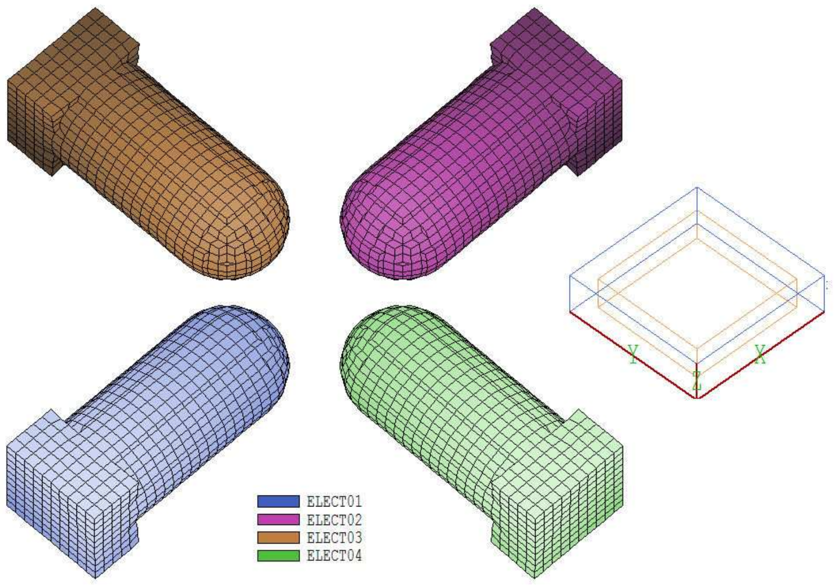


Figure 66: Macro-assembly created by rotating and positioning four assemblies using block operations.

```
  RegName(2): Elect01
  RegName(3): Elect02
  RegName(4): Elect03
  RegName(5): Elect04
END

PART
  Region: Air
  Name: Vacuum
  Type: Box
  Fab: 10.0 10.0 10.0
END

#ROTATE 0.0 90.0 0.0
#SHIFT -3.0 0.0 2.5
* Assembly 01
PART
  Region: Elect01
  Name: BlockBase01
  Type: Box
  Fab:   1.00  1.00  0.50
  Shift:   0.00  0.00  0.25
  Surface Region Air
END
PART
  Region: Elect01
  Name: BlockExtrusion01
  Type: Turning
```

```
      L    0.50  0.00  2.50  0.00
      A    2.50  0.00  2.00  0.50  2.00  0.00  S
      L    2.00  0.50  0.50  0.50  S
      L    0.50  0.50  0.50  0.00
  End
  Surface Region Air
END
#ENDSHIFT
#ENDROTATE

#ROTATE 0.0 -90.0 0.0
#SHIFT  3.0 0.0 2.5
* Assembly 02
PART
  Region: Elect02
  Name: BlockBase02
  Type: Box
  Fab:   1.00  1.00  0.50
  Shift:   0.00  0.00  0.25
  Surface Region Air
END
PART
  Region: Elect02
  Name: BlockExtrusion02
  Type: Turning
      L    0.50  0.00  2.50  0.00
      A    2.50  0.00  2.00  0.50  2.00  0.00  S
      L    2.00  0.50  0.50  0.50  S
      L    0.50  0.50  0.50  0.00
  End
  Surface Region Air
END
#ENDSHIFT
#ENDROTATE

#ROTATE 90.0 0.0 0.0
#SHIFT  0.0 3.0 2.5
* Assembly 03
PART
  Region: Elect03
  Name: BlockBase03
  Type: Box
  Fab:   1.00  1.00  0.50
  Shift:   0.00  0.00  0.25
  Surface Region Air
END
PART
  Region: Elect03
  Name: BlockExtrusion03
  Type: Turning
      L    0.50  0.00  2.50  0.00
      A    2.50  0.00  2.00  0.50  2.00  0.00  S
      L    2.00  0.50  0.50  0.50  S
      L    0.50  0.50  0.50  0.00
  End
  Surface Region Air
END
#ENDSHIFT
#ENDROTATE

#ROTATE -90.0 0.0 0.0
#SHIFT  0.0 -3.0 2.5
* Assembly 04
PART
  Region: Elect04
  Name: BlockBase04
  Type: Box
  Fab:   1.00  1.00  0.50
  Shift:   0.00  0.00  0.25
  Surface Region Air
END
PART
  Region: Elect04
  Name: BlockExtrusion04
  Type: Turning
      L    0.50  0.00  2.50  0.00
      A    2.50  0.00  2.00  0.50  2.00  0.00  S
      L    2.00  0.50  0.50  0.50  S
      L    0.50  0.50  0.50  0.00
  End
  Surface Region Air
END
#ENDSHIFT
#ENDROTATE

ENDFILE
```

18 Advanced shapes in MetaMesh scripts

Chapters 9, 10 and 12 discussed concepts for extrusions, turnings, STL objects, transitions and the neon model in relation to **Geometer**. This chapter show how parts based on the models are represented in the **MetaMesh** script.

18.1 STL objects

The STL type is the most versatile model available in **MetaMesh**. The associated part can assume almost any shape that can be defined with a three-dimensional CAD program. It is easy to specify a part based on an STL file in the **MetaMesh** script. The conversion process is fast and highly reliable if you follow the guidelines listed at the end of this section. Figure 67 shows an example. The STL file for the convoluted shape contained information on 6400 facets. The total mesh generation time was 86 seconds.

TYPE STL FPrefix [Fit, NoFit]
TYPE = STL, Linkage (Fit)
The keyword *STL* designates that the surface of the solid part is defined by an STL file. The file has the name `FPREFIX.STL` and must be available in the current directory. The file may be in either text or binary format. In the optional string parameter, the choice of *Fit* or *NoFit* determines whether the program attempts to conformalize the hexahedron mesh after volume fitting on the foundation mesh. It is important to note that even if the line contains the *Fit* directive, there will be no surface fitting unless the part section contains at least one *Surface Region* or *Surface Part* command (Sects.3.5 and 16.5). Fitting is performed only on the facets in contact with the regions or parts specified in the *Surface* commands. **MetaMesh** does not perform edge fitting on STL parts. The program ignores *Edge* directives and/or the fitting tolerance in *Surface* commands.

FAB FitToler [CutOff]
FAB 0.80 0.60
The first fabrication parameter (*FitToler*) for an STL part is the global surface fitting tolerance. It may have a value from 0.0 (no fitting) to 1.0 (perfect fitting). The default value $FitToler = 0.80$ gives good results with a minimum number of distorted elements. The second parameter functions only when the STL analysis method is set to *Proximity* (Sect. 15.3). Its function is somewhat complex. The proximity algorithm to determine whether an element lies within the STL surface relies on finding the closest facet. The process may fail when an object has sharp edges and the facet set includes acute triangles with large differences in size. The symptom is the presence of orphan elements that were incorrectly identified as inside or outside the shape. In this case, it is helpful to exclude facets from the distance comparison if they are inclined at a steep angle when viewed from the test point. **MetaMesh** computes the cosine of the angle between the facet normal vector and a vector from the point to the facet. The program excludes facets when the cosine is less than $CutOff$. The quantity may have a value between

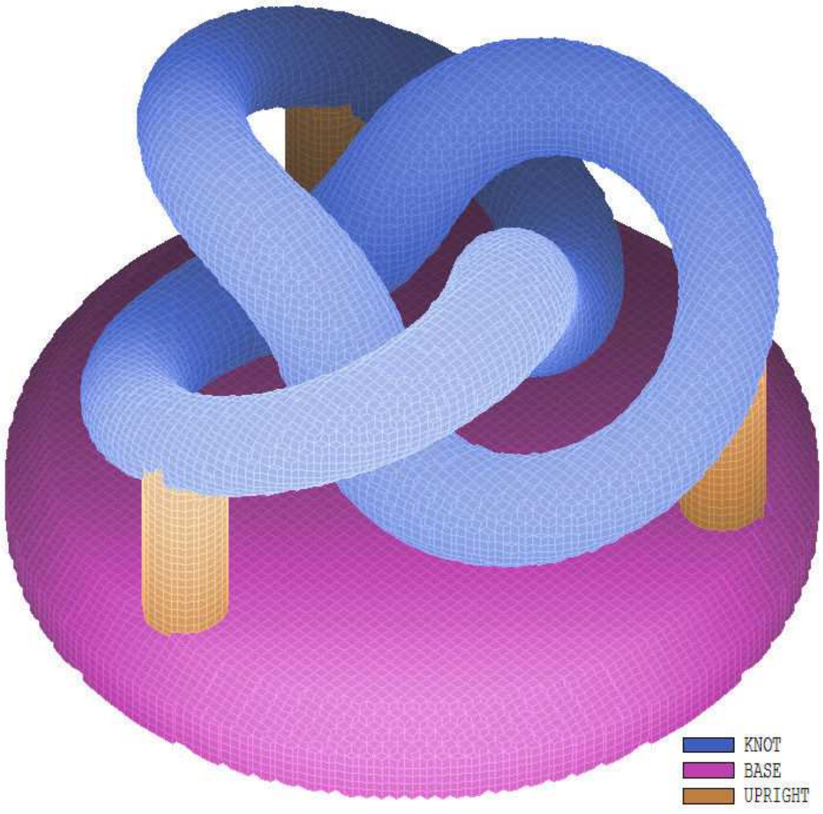


Figure 67: Conformal hexahedron mesh with an STL part (blue). The assembly also contains a turning (base, red) and three cylinders (uprights, orange).

0.0 and 1.0. – the default is 0.30. As a rule of thumb, increase the value of $CutOff$ if you observe orphan elements for an object with sharp edges and lower the value for an object with a smooth surface.

STLMETHOD [LINE, PROXIMITY]
STLMETHOD = PROXIMITY
Use this command to over-ride the `STL` analysis method set by the *STLMethod* command in the *Global* section (Sect. 15.3). The command applies the listed method only to the current part.

The transfer of three-dimensional information between programs that have different functions and different philosophies can be extremely frustrating. The **MetaMesh** process based on `STL` files is robust and easy to implement if you follow some common-sense precautions:

- When you create the STL file, try to use the minimum number of facets necessary to give a good representation of the surface.
- Break highly complex objects into several parts and export them as individual STL files.
- If you can control STL export properties in your CAD program, try to maximize the fraction of equilateral triangles and minimize the number of acute triangles.
- Remove details from the solid before exporting the STL file if they will not play a significant tole in the field solution. As shown in Fig. 68, small holes, fillets and other features may account for the majority of facets in the model.
- There is no guarantee that all STL files you encounter will be valid. Some may have holes or internal facets. It's a good practice to inspect files with a viewer before trying to construct a mesh. Note that the routines in **MetaMesh** are forgiving and often may give acceptable results for imperfect files.

Here is a general rule to decide if you have a good or bad STL file for conversion in **MetaMesh**. Load the file into a viewer and inspect it in the wireframe mode. If you can't figure out what the shape looks like using the extraordinary human powers of pattern recognition, imagine the challenge it will present to the code.

18.2 Defining extrusions and turnings

A common factor linking the extrusion, turning, transition and neon models is that the *Type* command is not a single line but a data structure. For an extrusion, the structure has the form:

```
TYPE EXTRUSION [X,Y,Z]
  Vector 1
  Vector 2
  Vector 3
    ...
  Vector N
END
```

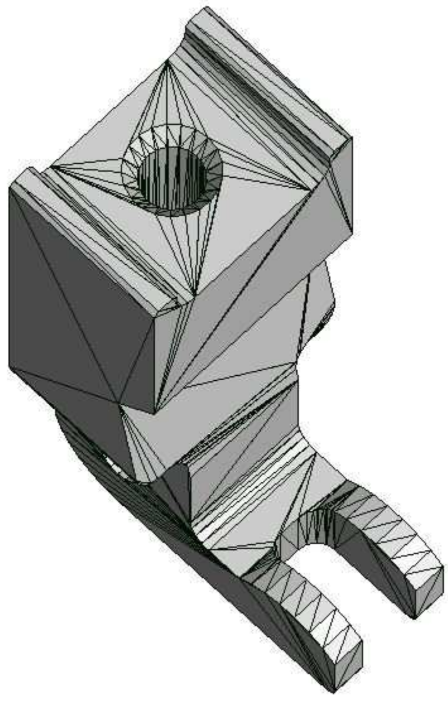


Figure 68: Typical STL object produced by a CAD program. Note that the hole and countersink require a large number of facets.

The set of vectors (that we call an *outline*) encloses a two-dimensional shape. By default, the extrusion extends in z and outline (used as the cross-section of the extrusion) lies in the x-y plane of the workbench. Add the optional string parameter Y or z to create extrusions that extend in x or y. In these cases, the coordinates are interpreted as y-z or z-x. Outlines may be created with a text editor or the graphical outline editor of **Geometer**. A line vector has the format:

```
L  XStart  YStart  XEnd  YEnd [SE]
```

and an arc vector has format

```
A  XStart  YStart  XEnd  YEnd  XCenter  YCenter [SE]
```

The extrusion *Type* structure may contain up to 50 vectors. During processing, **MetaMesh** analyzes outlines to confirm that 1) the vectors are contiguous, 2) the outline encloses a two-dimensional region and 3) and arc parameters are consistent. The program halts and issues an error report for an invalid outline. The following illustrates a valid structure:

```
TYPE EXTRUSION X
  L  -0.5  0.0  0.5  0.0  S
  A   0.5  0.0  0.0  0.5  0.0  0.0  S
  A   0.0  0.5 -0.5  0.0  0.0  0.0  S
END
```

The *Fab* command for an extrusion has a single parameter, the height H along the extrusion direction in the workbench frame. The *Rotate* and *Shift* commands have the same effect on extrusions, turnings, transitions and the neon model as they do on basic parts.

A turning is a figure of revolution about the z axis of the workbench space. It has an arbitrary cross-section in the r-z plane defined by an outline. The *Type* command has the format.

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

In this case the vector lines have the components:

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

Two parameters appear in the *Fab* command: *AngMin* and *AngMax*. They give the minimum and maximum azimuthal extent in the x-y plane. Enter quantitites in degrees. A value of 0.0^o corresponds to x axis of the workbench. The default values are $AngMin = 0.0^o$ and $AngMax = 360.0^o$ (a full turning).

18.3 Transitions

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. The cross-sections 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
```

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 workbench space) and the second vector is the mapping to the upper plane.

The *Fab* command has a single parameter gives the distance D between the two planes. The planes are located at positions $z = -D/2$ and $z = +D/2$ on the workbench. 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.52.

```
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 was used to generate the coordinates. The additional string parameter S indicates that surface fitting will occur on the associated facets.

18.4 Conformal fitting of turnings, extrusions and transitions

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

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 side facet 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. (Note that the SE option should be used infrequently, only if the edge in question is essential to the accuracy of the physical solutiion.) The following are important points to note about fitting extrusions, turnings and transitions:

- Fitting is not performed on a side surface if the strings S or SE are not included at the end of the vector data line, even if the *Part* section contains a *Surface* command. Conversely, there is no fitting unless the *Part* section contains *Surface* commands, even if the S designation appears on vector data lines.

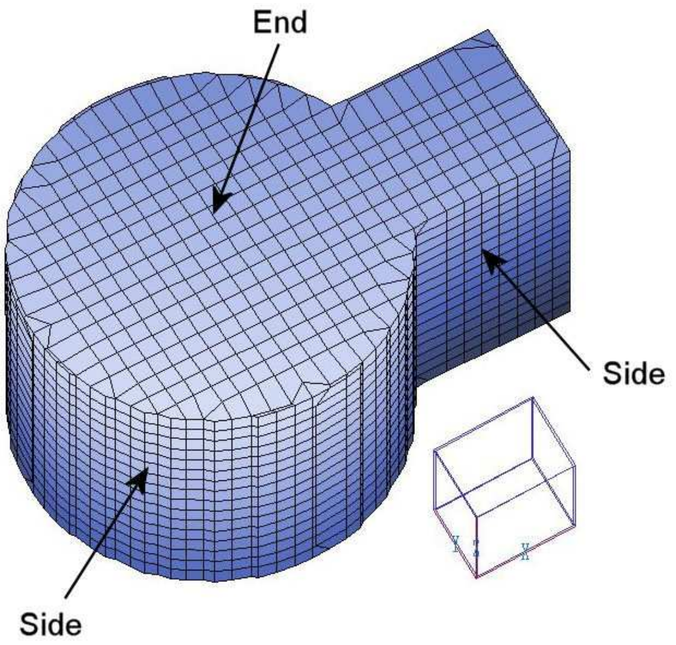


Figure 69: Extrusion with selective surface and edge fitting.

- The end faces of turnings (that cover less than 360.0°) and extrusions are fitted by default when there is a *Surface* command in the *Part* section, independent of the *S* or *SE* markings on the vector data lines.
- Edge fitting is not implemented on transitions.
- 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.

As an illustration, the following part section defines the extrusion shown in Fig. 69. 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.

```
PART
  Type Extrusion
     L    5.291E+00 -1.000E+00  7.500E+00 -1.000E+00 SE
     L    7.500E+00 -1.000E+00  7.500E+00  1.000E+00 SE
     L    7.500E+00  1.000E+00  5.291E+00  1.000E+00 SE
     A    5.291E+00  1.000E+00  3.000E+00  2.500E+00  3.0E+00  0.0E+00 S
     A    3.000E+00  2.500E+00  5.000E-01  6.083E-08  3.0E+00  0.0E+00 S
     A    5.000E-01  6.083E-08  3.000E+00 -2.500E+00  3.0E+00  0.0E+00 S
     A    3.000E+00 -2.500E+00  5.291E+00 -1.000E+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
```

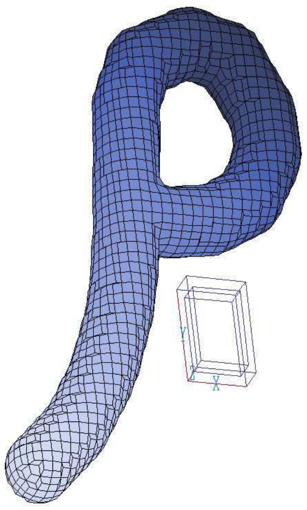


Figure 70: Construction of the letter ρ with the neon object.

By default **MetaMesh** assumes that extrusions, turnings and transitions exist in free space as in Fig. 69. To achieve the best representation, the code attempts to move nodes to nearby surfaces on the ends as well as on the sides. Many applications call for the construction of objects by butting extrusions, turnings and transitions together to form a continuous structure. In this case, the ends are not exposed and identified nodes should always be moved to the sides. You can signal this condition be adding the keyword *Sidefit* to the *Type* command for extrusions, turnings and transitions.

TYPE EXTRUSION SIDEFIT
TYPE TURNING SIDEFIT

In response to this command, **MetaMesh** moves nodes on all identified fitting facets to the nearest *side* of the object. Note that severe mesh distortions may occur if the *Sidefit* option is used when end facets are exposed.

18.5 Neon model

The neon model creates a tube of given radius that follows an arbitrary path in three-dimensional space. 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 workbench 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 500. You can define up to 16 neon parts in a script. The radius of the tube R is set by the first parameter in the *Fab* command.

Note that surface fitting of a neon object may become unpredictable for a coiled object if the surfaces associated with parallel line segments are close together. Surface fitting usually works if a neon object crosses itself, as in Fig.70. Because the objects have no edges, the *Edge* fitting option is ignored. You can position neon objects with the *Rotate* and *Shift* commands.

18.6 Extrusions and turning versus basic parts

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

- Always use the basic model if the object can be represented by a single part.
- Use a single turning or extrusion in preference to a combination of basic shapes. The mesh represnetation will be smoother at the transition points.

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

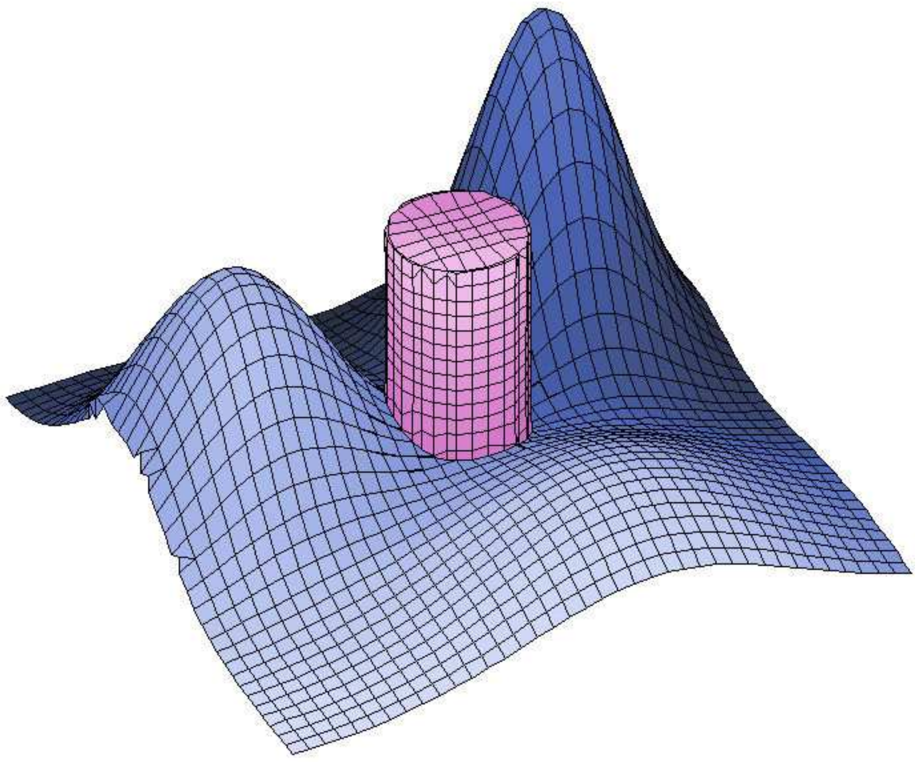


Figure 71: Mesh generated for the MOUNTAINS example.

19 Modeling mathematically-generated surfaces

Some applications call for complex three-dimensional surfaces that may be difficult (or even impossible) to construct using the mechanically-oriented 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 (Fig. 71).

A *Part* section in a **MetaMesh** script for a three-dimensional surface has the following components:

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

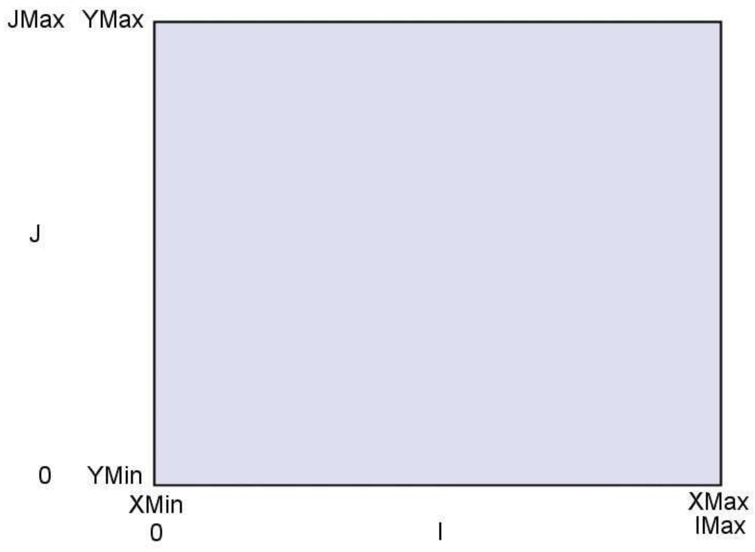


Figure 72: Conventions for tabulating values in the elevation file

The first two commands (*Type* and *Fab*) are required, the others are optional. The quantity *DataName* 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 Z_{min} gives the position of the lower surface of the solid. The solid extends from Z_{min} to $z(x, y)$ along the z axis of the workbench. 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 $RotX = 180^o$ 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 the 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 $\epsilon_s = 1.00$ in *Surface* commands.

A script may contain up to five *Surf3D* parts. **MetaMesh** issues an error message if any portion of a defined surface is less than Z_{min}. 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. 71).

The elevation file is in free-form text format. The data are tabulated over a rectangular mesh with the conventions shown in Fig. 72. 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 $0 \leq J \leq J_{max}0$. Elevation values $z(I, J)$ are tabulated at positions $[x(I), y(J)]$ where:

$$\Delta x = (X_{max} - X_{min})/I_{max}, \quad \Delta y = (Y_{max} - Y_{min})/J_{max},$$
$$X(I) = X_{min} + I\Delta x, \quad Y(J) = Y_{min} + J\Delta y.$$

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 I_{max} and J_{max} are integers. All other quantities are real numbers in any valid format. The following rules apply:

- Numbers in the header lines may be separated by any of the valid **AMaze** delimiters (space, comma, equal sign, tab, left paren, right paren).
- Comment lines beginning with '*" (asterisk) may 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 you can modify 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. 71. All files are included in the **MetaMesh** example library.

Model type: SURF3D
Fabrication parameters: 1) Z_{min} (lower bound)
Comments: The solid extends from Z_{min} to $z(x,y)$ on the workbench. The elevation values $z(x,y)$ are specified in a data file. The solid covers a rectangular region in the x-y plane. The data file header gives the limits of the rectangle ($X_{min}, Y_{min}, X_{max}, Y_{max}$).

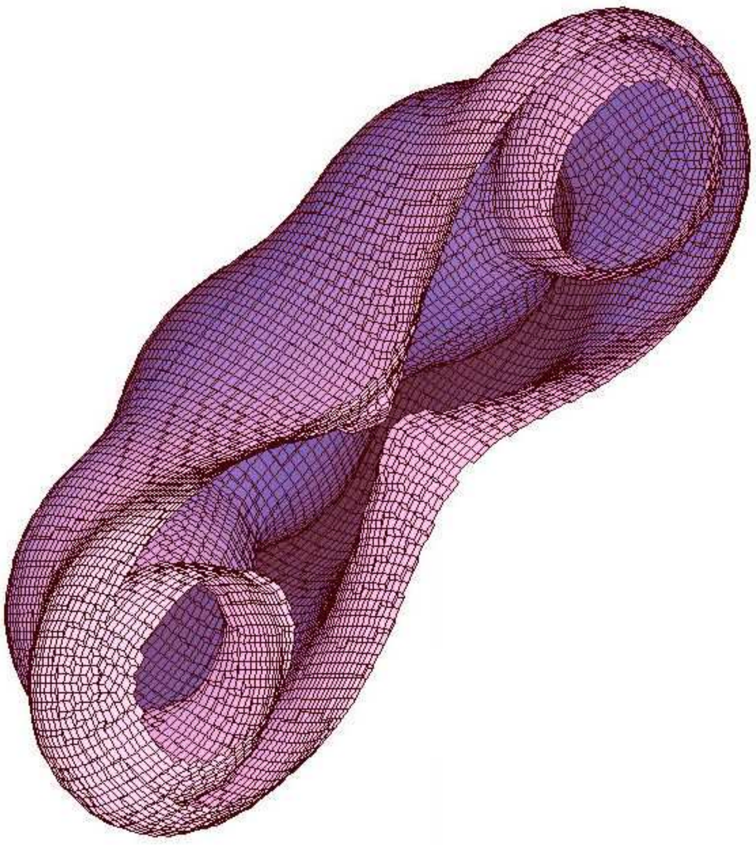


Figure 73: Cutaway view of a structured conformal mesh of surfaces of electrostatic potential for eight equally-spaced point charges in the x-y plane at a radius $R = 2.0$. The potential of one charge is given by $\phi = 2.0/r$. The region outer surface is $\phi = 8.0$ and the inner surface is $\phi = 9.0$.

20 Creating meshes from images

20.1 Three-dimensional images

MetaMesh has powerful features to construct three-dimensional conformal meshes directly from image data. In a general sense, a *image* consists of a set of values $H(x_i, y_i, z_i)$ defined at nodes of a regular mesh in Cartesian space. The definition encompasses a wide variety of data types:

- Voxel information in medical files generated by MRI, ultrasound or CAT scan devices.
- Calculated values of a three-dimensional mathematical functions.
- Data on physical quantities like pressure profiles determined from measurements or simulations.
- Composites of stacked two dimensional images in standard graphics formats like BMP.

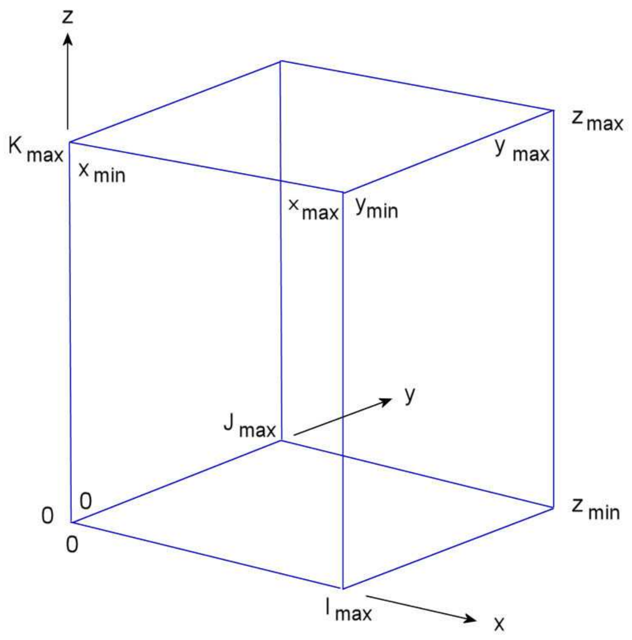


Figure 74: Conventions for three-dimensional image meshes

Metamesh can build meshes such that the boundaries of regions lie on surfaces $H(x, y, z) = A$. With this feature, you can quickly and efficiently represent objects with highly complex surfaces like that of Fig. 73. Image regions may be combined with regions constructed from the standard mechanical parts discussed in previous sections.

This section discusses conventions for representing three-dimensional images. The following section describes the format of the **MetaMesh** image file. Section 20.3 covers script commands to load and to analyze image files and summarizes volume-fitting methods to assign elements of the foundation mesh to regions. Section 20.4 discusses high-accuracy surface fitting methods that can be applied when $H(x, y, z)$ is a smooth, continuous function.

Figure 74 shows conventions for the regular mesh of **MetaMesh** images. We use a right-handed coordinate system where the index I gives the position along x, J along y and K along z. The mesh covers the spatial region defined by the limiting quantities $x_{min}, y_{min}, z_{min}, x_{max}, y_{max}$ and z_{max}. The indices have the range:

$$0 \leq I \leq I_{max}, \quad 0 \leq J \leq J_{max}, \quad 0 \leq K \leq K_{max}.$$

The image quantity $H(x_I, y_J, z_K)$ is recorded at node points of the mesh:

$$x_I = x_{min} + I\Delta x,$$
$$y_J = y_{min} + J\Delta y,$$
$$z_K = z_{min} + K\Delta z.$$

The mesh intervals are given by:

$$\Delta x = \frac{x_{max} - x_{min}}{I_{max}}, \quad \Delta y = \frac{y_{max} - y_{min}}{J_{max}}, \quad \Delta z = \frac{z_{max} - z_{min}}{K_{max}}.$$

20.2 MetaMesh image file format

There is a diversity of three-dimensional image formats, many of them proprietary, unnecessarily complex or poorly documented. It would impossible to create a code that could read all formats. Instead, we have chosen an simple standard format for the **MetaMesh** image file. With a knowledge of a source format, it is easy to translate to **MetaMesh**.

MetaMesh images are text files with a name of the form `FileName.MMI` (for **M**eta**M**esh **I**mage). The file consists of a two-line header followed by N data lines, where

$$N = (I_{max} + 1)(J_{max} + 1)(K_{max} + 1).$$

The following sample shows a portion of an image file created by **AtoM**:

```
* MetaMesh file created from an ANALYZE file with AtoM
* Copyright, Field Precision 2005
* VoxUnits: mm
   180   216   180
   -90.500  -108.500   -90.500    90.500   108.500    90.500
  0.052
  0.056
  0.068
  0.092
  0.112
  0.111
  ...
```

The following rules apply to the image file

- Any number of documenting comment lines (beginning with an asterisk) may appear at (and only at) the beginning of the file.
- The first header data line contains the quantities I_{max}, J_{max} and K_{max} in any valid integer format with spaces or commas as delimiters.
- The second header data line contains the quantities x_{min}, y_{min}, z_{min}, x_{max}, y_{max} and z_{max} in any valid real-number format.
- The remaining data lines contain $H(I, J, K)$ in any valid real-number format.

The following code extract from **AtoM** illustrates how to read or to write the image data lines:

```
DO K=0,KMax
  DO J=0,JMax
    DO I=0,IMax
      RTemp = REAL(IV(I,J,K))
      WRITE (OutFile,5350) RTemp
    END DO
  END DO
END DO
```

The resulting order is:

```
H(0,0,0)
H(1,0,0)
...
H(IMax,0,0)
H(0,1,0)
H(1,1,0)
...
H(0,JMax,0)
H(1,JMax,0)
...
H(IMax,JMax,0)
H(IMax,JMax,1)
...
H(IMax,JMax,KMax)
```

20.3 MetaMesh script commands for images

Image processing operations in **MetaMesh** are controlled by the following command structure that appears in a *Part* section:

```
TYPE IMAGE FilePrefix [ABS, REL]
  HLow(1)  HHigh(1)     RegNo(1)
  HLow(2)  HHigh(2)     RegNo(2)
  ...
  HLow(NInt)  HHigh(NInt)  RegNo(NInt)
END
```

In the *Type* command, the quantity *FilePrefix* is the prefix of an image file available in the working directory. The full file name is `FileName.MMI`. The script may include up to five image-type parts mixed with standard parts. The program handles images much like other parts. You can include rotations, shifts and a name in the **PART** section. Up to 120 data lines may be included between the *Type* and *End* commands to define intervals. Each data line must contain two real numbers and an integer. The quantities *HLow* and *HHigh* define limits. During volume fitting, an element of the foundation mesh with centroid at (x, y, z) will be assigned to region $RegNo(N)$ if

$$H_{Low}(N) \leq H(x, y, z) \leq H_{High}(N).$$

The volume-fitting process creates a set of regular elements with the same part number but with different region numbers, depending on number of intervals defined.

The optional string parameters *Abs* and *Rel* in the *Type* command determine how the interval values are interpreted. For the default *Abs* option, values of *HLow* and *HHigh* are compared directly with those in the image file. For example, with an MRI image containing unsigned character data, *HLow* and *HHigh* should be in the range 0.0 to 255.0. Under the *Rel* option, **MetaMesh** computes the minimum and maximum data values in the image file (H_{min} and H_{max}) and assigns an element to a region if

$$H_{Low}(N) \leq \frac{H(x,y,z) - H_{min}}{H_{max} - H_{min}} \leq H_{High}(N).$$

In this case, *HLow* and *HHigh* should be in the range 0.0 to 1.0.

20.4 Conformal meshes for images with continuous values

The volume-fitting process assigns elements of the foundation mesh to regions according to the values in the image file. The result is a regular mesh of box elements. The boundaries of regions have a stair-step appearance and do not conform exactly to surfaces of $H(x,y,z) = H_{Low}$ and $H(x,y,z) = H_{High}$. **MetaMesh** has a high-accuracy fitting algorithm to correct surfaces that can be applied when the data are smooth and continuous. The procedure relies on the fact that ∇H is normal to ideal region boundaries, which are surfaces of constant $H(x,y,z)$. The program employs the following operations to correct the node positions of a stairstep mesh:

- Collect the set of element facets that constitute the boundaries of each region.
- Divide the facets into two sets corresponding to H_{High} and H_{Low}.
- For each node at position $\mathbf{X}_0$ connected to a facet at H_{high}, find the value of the image function and its gradient at the current node position, H_0 and ∇H_0.
- Calculate the displacement vector

$$\delta\mathbf{x_0} = \frac{\nabla H_0}{|\nabla H_0|^2}\,(H_{High} - H_0).$$

- Correct the node position according to $\mathbf{X}_1 = \mathbf{X}_0 + \alpha\ \delta\mathbf{x_0}$, where $\alpha < 1.0$.
- Perform the same operations on the H_{Low} nodes.
- Continue corrections through several iterations to allow the nodes to converge to the ideal surface.

Figures 73 and 75*b* show examples of high-accuracy surface fitting.

Fitting is controlled by the following command:

FAB NCycle
FAB = 4

The single parameter in the *Fab* command gives the number of cycles for node position correction. A value of zero (the default) suppresses conformalization. Larger values give a closer approximation, but may lead to increased numbers of distorted elements.

To conclude, we shall discuss how *Image* type parts interact with the standard mechanical parts discussed in previous chapters. The following rules apply:

- The overwrite principle is in effect. In processing an image, **MetaMesh** reassigns elements in shared volumes with any parts that appear earlier in the script including other images.
- Image operations require a large amount of temporary memory, so volume assignment and surface corrections are performed immediately in sequence. One implication is that volume fitting for subsequent standard parts may take place on a modified foundation mesh.
- Surface fitting of standard parts occurs after any image processing operations. Consequently, the surfaces of standard parts take precedence over those of images.

The `RFQ` example illustrates a useful application of the **MetaMesh** image capabilities. Given an expression for a valid solution to the Poisson equation, we can generate the corresponding electrostatic field by constructing a set of electrodes that lie on surfaces of constant potential. The equation for the instantaneous potential in an RFQ (radio-frequency quadrupole) accelerator is:

$$\phi(x,y,z) = \phi_0\left[\frac{x^2}{a^2}\right]\left[1+\epsilon\sin\left(\frac{2\pi z}{D}\right)\right] - \phi_0\left[\frac{y^2}{a^2}\right]\left[1-\epsilon\sin\left(\frac{2\pi z}{D}\right)\right] + \phi_0\epsilon\left(\frac{D^2}{2\pi^2 a^2}\right)\sin\left(\frac{2\pi z}{D}\right),$$

where the normalizing potential is related to the average pole tip field E_0 by

$$\phi_0 = E_0 a/2.$$

The Perl script `rfq.pl` creates a data file `RFQ.MMI` of potential values from the equation with the following parameters: $x_{min} = y_{min} = -1.3$, $x_{max} = y_{max} = 1.3$, , $z_{min} = -3.0$, $z_{max} = 3.0$, $\Delta x = \Delta y = \Delta z = 0.10$, $D = 2.0$, $a = 0.5$ and $\epsilon = 0.08$. The **MetaMesh** input file rfq.min contains the following *Part* sections:

```
Part
  Type Box
  Region 1
  Fab  5.00  5.00   10.00
End
Part 2
  Type Image RFQ.MMI
     0.50  20.00  2
   -20.00  -0.50  3
  End
  Fab  8
End
```

Elements with potential values greater than $\phi = +0.5$ V are assigned to region 2 (positive electrodes), while elements with $\phi \leq -0.5$ V constitute region 3 (negative electrodes). All

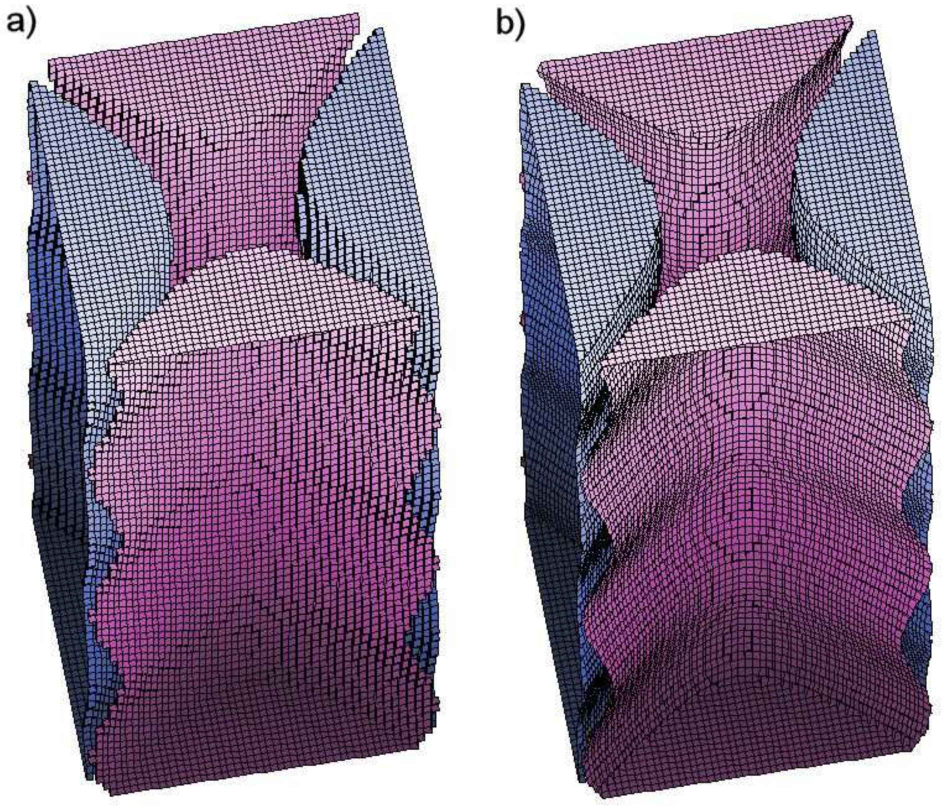


Figure 75: Electrode shapes determined from equipotential surfaces for a radio-frequency quadrupole accelerator. *a*) Surface fitting suppressed. *b*) Surface fitting applied.

other elements have the property of vacuum (region 1). Figure 75 shows the resulting electrode shapes. The mesh on the left-hand side involved only volume fitting. At the given size of elements in the foundation mesh, the resolution of the small undulations that create longitudinal accelerating components of electric field is poor. In contrast, the surfaces in the fitted mesh on the right-hand side follows theoretical contours almost exactly.

21 Importing parts represented by unstructured tetahedron meshes

This chapter describes methods to create a part from a three-dimensional object represented by an unstructured tetrahedron mesh (UTM). The part may be a single unit or consist of several disconnected segments. You can add multiple objects within a **MetaMesh** script and combine them with standard part models. The only constraint is that all elements associated with the object must have the same region number. The mesh translation capabilities of **MetaMesh** make it possible to import information from CAD programs, mesh generators or finite-element programs that create UTMs. For this purpose, we have defined a standard file format for unstructured tetrahedron meshes.

The following section reviews the format of the **MetaMesh** file for a tetrahedron mesh. Section 21.2 summarizes **MetaMesh** script commands to load and to transform UTM data. Section21.3 briefly describes the method employed in **MetaMesh** for UTM conversion.

21.1 Format of the UTM file

A wide variety of programs can create unstructured tetrahedron meshes for mechanical or mathematical objects. It would be impossible to include direct support for all programs in **MetaMesh**. Instead, we have included the capability to process parts with volumes and surfaces specified by a UTM in a standard text file format. It is relative easy to create programs or scripts that translate from any documented format to the **MetaMesh** standard.

A **MetaMesh** file for an unstructured tetrahedron mesh has a name of the form `FPrefix.UTM`. The file is in text format with spaces used as delimiters. **MetaMesh** employs a free-form parser that can handle multiple spaces, indentations and real numbers in any valid format (such as 5.245, 6.78E+01, 6, ...). Table 12 shows the file organization. You may include any number of descriptive header lines at the beginning of the file as long as first word of any line is not *Element* or *Node*. To find the starting position, **Metamesh** searches for the line whose first entry is the string *Element*. The second entry in the line ($NElem$) is an integer that equals the number of tetrahedron elements in the mesh. The program then reads $NElem$ data lines. Each line contains four integer numbers, the numbers of nodes in the following node list that correspond to the vertices of the tetrahedrons. The program next searches for the line beginning with the string *Node* and reads the integer number $NNode$. Finally, **MetaMesh** reads the $NNode$ data lines of the node list. Each line contains three real numbers giving the spatial coordinates of the node, (x, y, z). The program makes several checks of syntax and also confirms that the number of nodes equals the maximum node number referenced in the element definitions. **MetaMesh** checks the integrity of the mesh by ensuring that the calculated volume of each element is a positive number.

It is important that the nodes of an element appear in the rotational order shown in Fig. 76. Consider the facet defined by nodes 1, 2 and 3. If you move the fingers of your right hand from $1 \Rightarrow 2 \Rightarrow 3$, the thumb should point in the direction of 4. **MetaMesh** will issue an error message if the nodes do not follow the right-hand rule.

Table 12: MetaMesh standard unstructured tetrahedron mesh file format

```
(Optional header lines)
ELEMENT  NElem
   N11  N12  N13  N14
   N21  N22  N23  N24
   N31  N32  N33  N34
   N41  N42  N43  N44
(Optional lines)
NODE  NNode
   X1   Y1   Z1
   X2   Y2   Z2
   X3   Y3   Z3
   X4   Y4   Z4
    ...
```

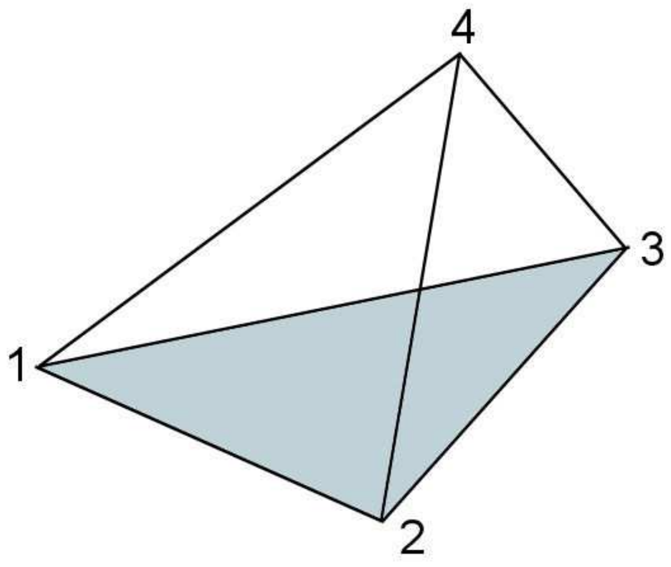


Figure 76: Rotation order of tetrahedron nodes for the standard **MetaMesh** UTM file

The *Help* menu in the main **MetaMesh** menu contains a useful tool to check the syntax of UTM files. In response to the command *UTM file parameters*, the load dialog lists entries with names of the form `FPREFIX.UTM`. The routine checks the consistency of the file and reports the number of nodes and elements and the spatial limits of nodes.

21.2 Script commands for UTM import

With a few exceptions, a UTM object behaves in the same way as other geometric models in **MetaMesh** (cylinders, boxes, extrusions, ...). The main difference is that the surface is more complex and takes correspondingly longer to analyze. You may apply local rotations and shifts and the part will be affected by global rotations and shifts defined by script directives. In theory, you could define up to 250 UTM parts, although in practice more than few would be unwieldy. A part section for a UTM object may contain the following components:

```
PART
  TYPE UTM FPrefix [Fit, NoFit]
  FAB SurfToler
  NAME PartName
  REGION RegNo
  ROTATE  Rx Ry Rz [RotOrder]
  SHIFT Sx Sy Sz
  COAT RegBound RegNew
END
```

The following rules apply to the part commands

- In the *Type* command, the model type is *UTM*. The string parameter is the prefix of the data file (`FPrefix.UTM`). Both files must be available in the working directory. **MetaMesh** has several error traps to determine if the file format is valid.
- The optional string parameter *Fit* (the default) or *Nofit* in the *Type* command controls conformalization of the part surface. **MetaMesh** uses the same general procedure to represent UTM objects as it uses for other models. The first step is volume fitting. Here the program assigns hexahedron elements of the foundation mesh to the object if its center-of-mass is inside an element of the object on the tetrahedron mesh (Fig. 77*a*). If the *Fit* option (or no entry) appears in the *Type* command, the program collects element facets on the hexahedron mesh that are on the surface of the object. **MetaMesh** then shifts nodes to positions that lie on nearby surface facets in the tetrahedron mesh (Figs. 77*b* and 77*c*). If the *Nofit* option appears, the object has the stairstep appearance of Fig. 77*a*. Note that volumetric assignment consumes most of the conversion time, so there is little incentive to using the *Nofit* option except for debugging.
- Elements defined by the *UTM* model over-write elements of previously-defined parts in the shared volume.
- The commands *Name, Region, Rotate, Shift* and *Coat* operate in the same way they do for other **MetaMesh** part models.

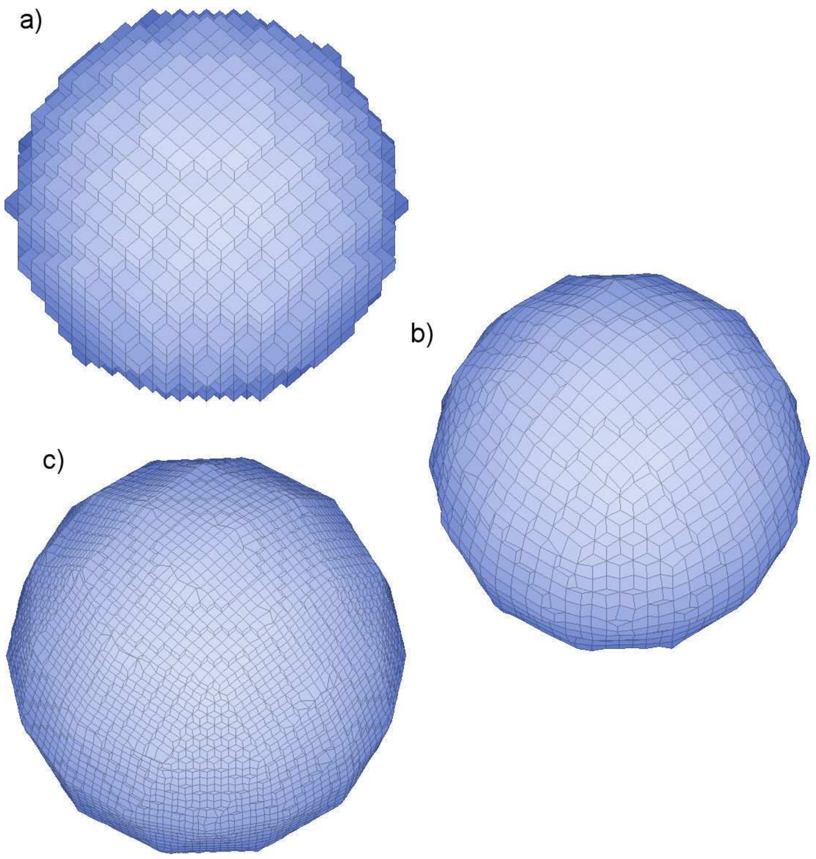


Figure 77: Hexahedron meshes created from a tetrahedron mesh of a sphere of radius 0.5 at coarse resolution. *a*) Hexahedron element size 0.05 with no surface fitting. *b*) Element size 0.05 with surface fitting. *c*) Element size 0.025 with surface fitting

- The *Surface* command is ignored. There is a small difference in the logic of surface fitting for *UTM* parts compared to simple models. Storage of a tetrahedron mesh requires a large amount of memory, so **MetaMesh** loads only one UTM at a time. Surface fitting takes place immediately after the object is loaded. For simple parts, surface fitting is deferred until all volumes have been assigned.
- The *Fab* command contains the single entry *SurfToler*, the tolerance for surface fitting. A value of 1.00 gives perfect fitting but may increase the number of distorted elements. The default value of 0.95 generally gives good results. Lower the fitting tolerance if there are severe mesh distortions.

21.3 Mesh conversion method

This section gives a brief description of methods employed in **MetaMesh** to convert unstructured tetrahedron meshes to conformal hexahedron meshes. The intention is to give you some guidelines for picking good resolution values that will minimize processing time. As described in Sect. 21.1. `UTM` files contain a set of node positions at the vertices of tetrahedrons and lists of the particular nodes that constitute each element. The inherent property of the unstructured mesh is that the elements and nodes do not occur in a predictable order in the lists. That is why the conversion process in **MetaMesh** may occupy considerable time. To find if a spatial point is inside one of the elements of the UTM, the program may have to search the entire list. This fact emphasizes the advantage of the structured meshes used in the **AMaze** programs – it most cases it takes more time to extract information from the UTM than to carry out a field solution on the resulting structured hexahedron mesh.

The **MetaMesh** conversion process consists of the following steps:

1. Open the file and extract information on nodes and elements. At the same time, determine the maximum and minimum spatial extents of the object in the solution space.

2. Make a list of surface facets in the tetrahedron mesh by checking elements where a combination of three nodes appears in only one element of the list.

3. Carry out the volume assignment of elements in the foundation mesh. Calculate the center-of-mass of each hexahedron. If the point is inside the extents of the object, check the list of tetrahedron elements. If the point is inside an element, assign the associated *RegNo* and *PartNo* to the element and surrounding nodes in the hexahedron mesh. To reduce the run time, **MetaMesh** employs a optimal search technique that takes advantage of pseudo-structure in the UTM element list.

4. Check to see which facets are on the surface of the part in the hexahedron mesh and collect all nodes connected to the facets.

5. For each surface node, find the closest facet in the list of tetrahedron surface facets. Find the closest point inside or on the edge of the triangular facet and project the hexahedron node to the point using *SurfToler*.

It is worthwhile to clarify the goal of the conversion procedure. Clearly it is impossible to make an exact replica of an unstructured tetrahedron mesh on a structured hexahedron mesh except in the limit of vanishingly small elements. Therefore, our goal should be to make an approximate representation that provides sufficient accuracy in the subsequent finite-element calculation. A conversion with extremely high accuracy may not be necessary and will significantly increase the times required for mesh processing and the finite-element solution. Figure 77 illustrates some concerns. We constructed a sphere of radius $R = 0.5$ and purposely picked a coarse resolution for the tetrahedron mesh. The result is a solid with a variety of flat surface facets. The conversion with no surface fitting (Fig. 77*a*) is clearly inadequate. Calculated electric fields or thermal flux would vary widely over the discontinuous surface. Figure 77*b* shows the result with surface fitting for a hexahedron element width of 0.05. This is a substantial improvement and may be sufficient to give good accuracy in the finite-element calculation. Nonetheless, there are small bumps on some of the facet edges. If our intention is to reproduce accurately the faceted structure, we would use smaller hexahedron elements as in Fig. 77*c*. On the other hand, this would be a wasted effort if our goal was to represent a sphere. In this case, we would use smaller tetrahedron elements to approximate a smooth surface. The element size of 0.05 would probably be sufficient to approximate such a surface. Some quantitative values for the example of Fig 77 will give a sense of the accuracy of the conversion process. An ideal sphere of radius $R = 0.5$ has volume $V = 0.5236$, while the faceted structure created has $V = 0.4763$. With element size 0.05 the volume of the resulting shape on the hexahedron mesh is $V = 0.4747$, a difference of only 0.34%. With a reduced element size of 0.025, the difference drops to 0.04%.

22 MetaMesh - loading, editing and processing scripts

MetaMesh runs interactively in a window when you call the program with no command-line parameters. When **MetaMesh** starts only the *File* and *Help* menus are active. Other menus become active when data are loaded and processing takes place. We shall begin by discussing the commands of the file menu.

22.1 File menu commands

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 `RUNNAME.MIN`. Changing directories in the dialog will also change the working directory of the program. In this case, output files will be written to the new directory.

LOAD MDF FILE

Load a previously-processed mesh file in the format described in Chap. 25. You can inspect the mesh, but it is not possible to make changes. Because of the over-write 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 a mesh file saved in text format).

SAVE MESH

Save the currently-processed mesh in the format described in Chap. 25. The output file has a name of the form `RUNNAME.MDF`, where *RunName* is the prefix of the currently loaded script. The suffix `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 editor to view or to modify the current input script. This command is active only if a script has been loaded. If an error occurred during script processing, the cursor will be positioned on the error line when the editor is opened.

EDIT MLS FILE

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

EDIT FILE

Work on any text file with the editor. Changing directories in the dialog does not change the working directory of the program.

22.2 Process 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 Chapter 3. Processing terminates 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 recorded in the text listing file `RUNNAME.MLS`.

MetaMesh performs several checks to ensure that the completed mesh is valid and records the results in the listing file, `RUNNNAME.MLS`. An example is shown in Table 13. Optional volume integrals are taken over all elements and organized by region number. Three-dimensional integrations are performed over each hexahedron using the normal-coordinate method. 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 $\mathbf{u}_3$ (edge unit vectors of an element at a node arranged in the positive 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, 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 **AMaze** field solution program may exhibit numerical instability. There are several ways to correct the condition, listed in order of the severity of the problem:

- Check that the logic of *Surface* commands is correct. Make sure that the directives are not telling the code to fit internal facets of a part to its outer boundary.
- Check that the element size in the problem area is small enough to resolve objects.
- Reduce the fitting tolerance (ϵ_s) for the affected surface.
- Eliminate edge fitting for the affected surface.
- Selectively deactivate surface fitting to locate the source of the problem.

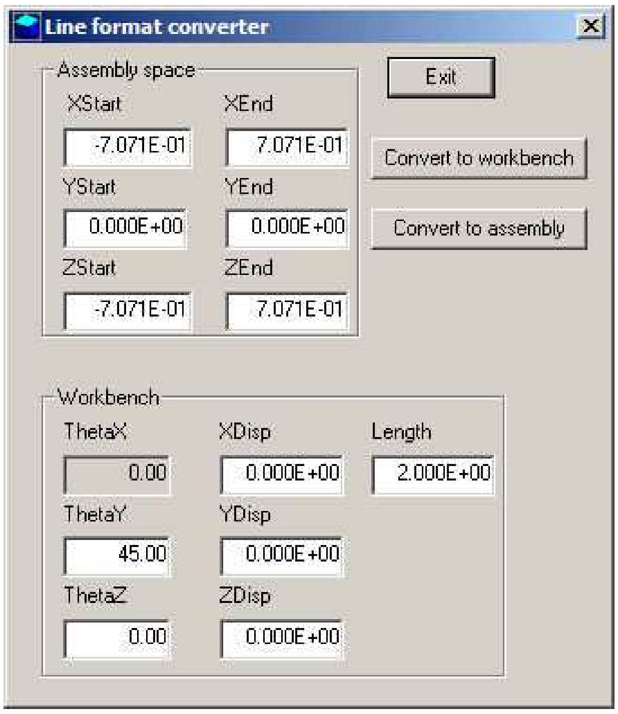
22.3 Help menu commands

METAMESH MANUAL

The command loads this manual in your default PDF reader. This command will not operate unless the file `METAMESH.PDF` is in the same directory as the executable program `metamesh.exe`.

Table 13: Example of diagnostic messages in the **MetaMesh** listing file.

```
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
```



Assembly space

XStart	XEnd
-7.071E-01	7.071E-01
YStart	YEnd
0.000E+00	0.000E+00
ZStart	ZEnd
-7.071E-01	7.071E-01

Workbench

ThetaX	XDisp	Length
0.00	0.000E+00	2.000E+00
ThetaY	YDisp	
45.00	0.000E+00	
ThetaZ	ZDisp	
0.00	0.000E+00	

Figure 78: Line format conversion dialog.

LINE CONVERTER

The *Line Converter* is a utility to help in the preparation of scripts. In response to the command, the program displays the dialog of Fig. 78. The quantities in the top group are the absolute coordinates in the solution space of the starting and ending points of a line. When you click the button *Convert to workbench*, the program fills in values for the bottom group. The quantity *Length* is the line length, the single parameter is 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 angles to use 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 the 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.

22.4 Command line or batch file operation

You can also launch **MetaMesh** as a background task from the command prompt. In this mode, the program can run under control of a DOS batch file or the Field Precision **GCon** utility. The advantage is that you can set up your computer to perform an extended series of operations autonomously. **MetaMesh** runs in the background 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.

A DOS 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 minimize problems, we advise running batch scripts from **GCon** rather than from the command prompt.

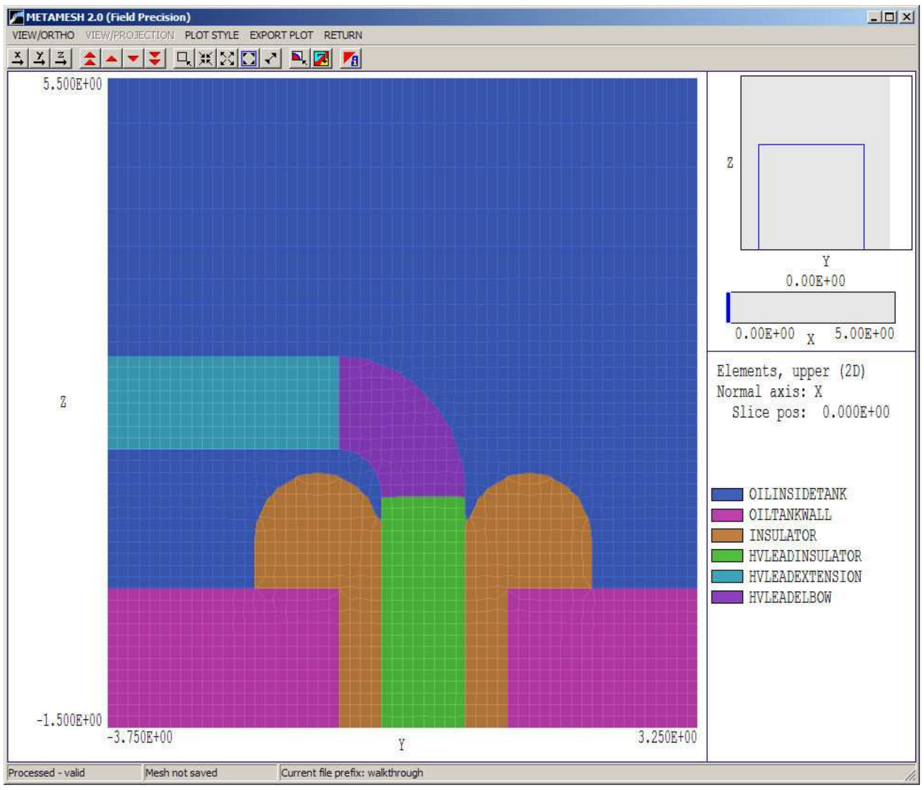


Figure 79: Screen display, 2D plots. Example of *Mesh/upper elements* plot.

23 MetaMesh interactive mode - 2D plot menu

23.1 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. 79.

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_0 contains the set of nodes (I_0, J, K), where $0 \leq J \leq J_{max}$ and $0 \leq K \leq K_{max}$. 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 the x axis.

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. 79). 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, at times the view may not be 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.

MESH/UPPER ELEMENTS
MESH/LOWER ELEMENTS

A plot of element facets in the logical plane with color-coding by elements (Fig. 79). Because the node planes lie on boundaries between elements, we can choose colors corresponding to the part or region number of the elements either above or below the plane. The upper element plot shows element identities above the logical plane, where the term *above* indicates the direction of higher values of the normal axis index.

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 80 shows an example. The program displays color-coded information on elements above or below the logical plane.

23.2 Adjusting the view

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 commands depend on the plot group. Tp begin, 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 move between logical planes.

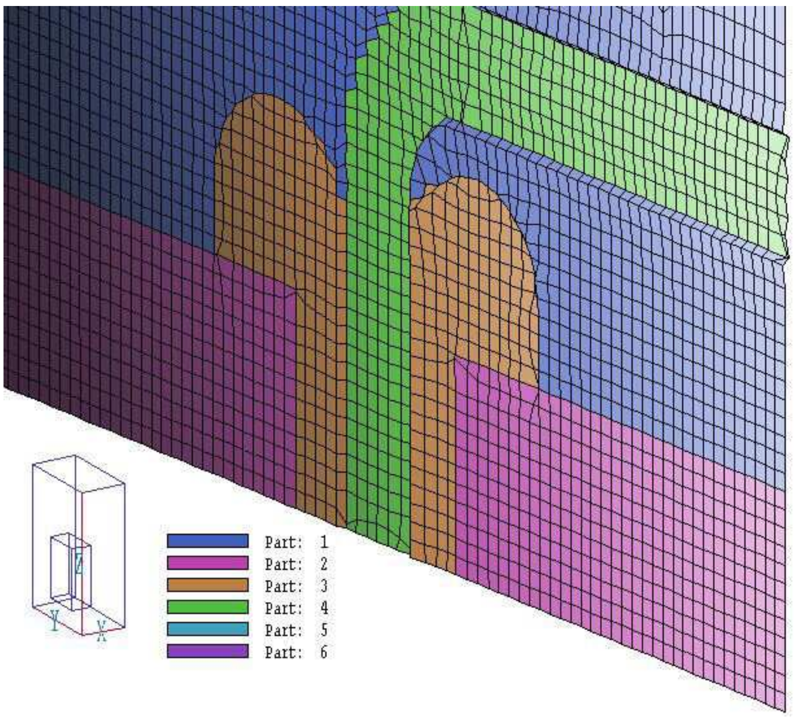


Figure 80: Example of a *Projection/upper elements* plot

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 displayed 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. Note that plots are constructed at the approximate midpoints of logical planes along the normal axis (*i.e.*, the middle of a layer of elements). Therefore, values displayed in response to slider movements will change discontinuously.

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 *Esc* 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 the *View/Ortho* menu are used to set plot features or to request element information

GRID DISPLAY

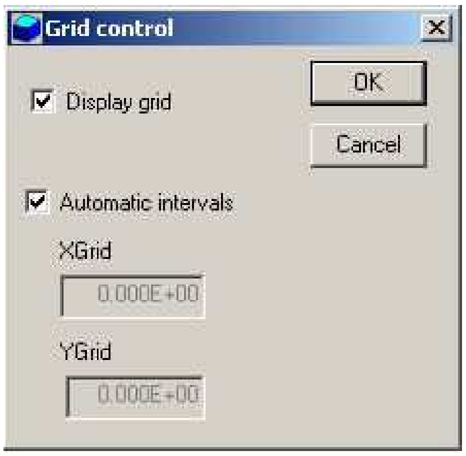
The program displays the dialog shown in Fig. 81 to control the display of the plot grid. Under the *Automatic* option, **MetaMesh** picks convenient intervals for the grid display. Alternatively, you can set intervals manually. The grid intervals are listed in the information window.

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


Grid control

☑ Display grid

☑ Automatic intervals

XGrid: 0.000E+00

YGrid: 0.000E+00

Figure 81: Grid control dialog

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 when the *Mesh/upper elements* or *Mesh/lower elements* plot type is active. 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.

The *View/Projection* menu is active when the plot type is either *Projection/upper element* or *Projection/lower element*. The menu contains the previously-discussed set of commands to set the logical plane. In addition, you can use the mouse to change your viewpoint of the three-dimensional plane. See Section 24.2 for a complete description of mouse controls for interactive 3D views.

23.3 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 Windows *Start Menu* to make changes in the default before starting **MetaMesh**. With the appropriate print drivers, you can create PDF files or send the information to a network printer.

SAVE PLOT FILE

Use this command to create a graphics file of the current plot in either Windows Bitmap (BMP) or Portable Network Graphics (PNG) formats. In the dialog, specify the format, the size in pixels and the file prefix. The graphics file is created in the current directory.

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.

23.4 Saving and loading views

The creation of plots for presentations may involve some effort. With the following two commands, you can save all the current view parameters and immediately restore the plot.

SAVE NAMED VIEW

Save the view parameters for the current plot. Quantities such as the slice axis, slice position and zoom limits are saved for two dimensional plots. Parameters for three-dimensional plots include the viewpoint position, displayed regions and cut planes. The information is stored in a text file in the current directory with a name of the form `FPREFIX.FPV`.

LOAD NAMED VIEW

Load a view file and refresh the plot. Note that you must be in the 2D plot menu to recall a 2D plot view. Similarly, 3D plot views may only be loaded to the 3D plot menu.

The file contains the complete set of plot parameters. This excerpt illustrates the format:

```
Program: MetaMesh
2D/3D: 2D
DisplayBy: Regions
Outline: On
NSlice:     40
SliceAxis: YAxis
PlotType: LogElemUp
XPMin: -1.500000E+00
XPMax:  4.250000E+00
...
```

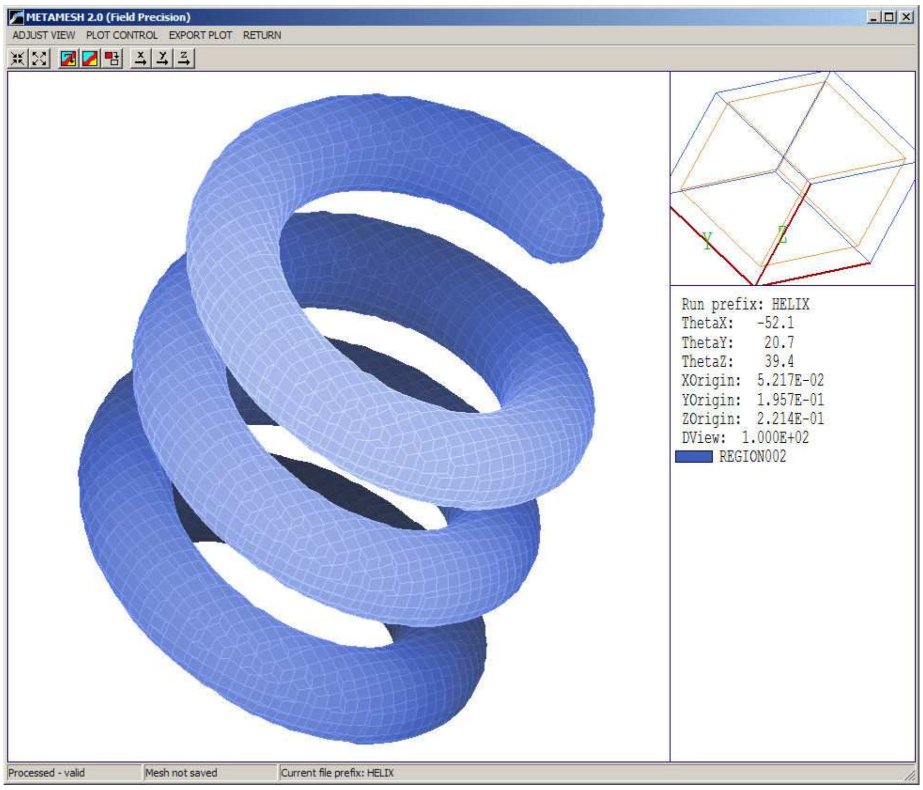
If a specific solution file is loaded, the plot will be restored exactly. The saved view feature in **MetaMesh** has two useful features if a different meshes are loaded:

- Dynamic adaptation to different solutions.
- Option for user control of the view parameter set.

Regarding the first feature, there are situations where you want to create consistent views of a set of solutions with different geometries, maintaining a similar appearance. Some plot properties (like the viewpoint rotation matrix) are applicable to any solution, but others (like region cut planes or slice plot limits) depend on the geometry. **MetaMesh** checks each plot parameter for validity. If a parameter is outside the allowed range for the currently-loaded solution, the program computes an alternative. The goal is to preserve as many features of the view as possible.

You can modify view files with an editor. The order of entries is not rigid. On input, **MetaMesh** uses a free-form parser. If a parameter is missing, the program simply makes no change from the value current in the program. The implication is that you can modify a saved view to include only elements essential to your application. For example, you could compare a series of assemblies with different sizes, maintaining an orthographic 3D view from the same point in Cartesian space. In this case, the view file would contain only the entries:

```
DView:  1.000000E+37
R11:  8.660253E-01
R12: -5.000002E-01
R13:  0.000000E+00
R21:  1.669031E-01
R22:  2.890846E-01
R23:  9.426408E-01
R31: -4.713208E-01
R32: -8.163510E-01
R33:  3.338061E-01
```



```
Run prefix: HELIX
ThetaX:   -52.1
ThetaY:    20.7
ThetaZ:    39.4
XOrigin:  5.217E-02
YOrigin:  1.957E-01
ZOrigin:  2.214E-01
DView:  1.000E+02
▬ REGION002
```

```
Processed - valid    Mesh not saved    Current file prefix: HELIX
```

Figure 82: Screen display, 3D plots.

24 MetaMesh interactive mode - 3D plot menu

24.1 Surface plot methods

If you click on *Plot 3D* after mesh processing, **MetaMesh** displays the toolbar and menus illustrated in Fig. 82. 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.

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 with $PartNo = 8$ that border on elements with $PartNo \neq 8$. Similarly, the surface of region 3 consists of the set of facets of elements that have $RegNo = 3$ that border on elements with $RegNo \neq 3$. 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 the most distant facets. 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 of the plot, the 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 scan order and calculated distances, facets on a boundary may be plotted with colors associated with either adjacent part or region.

You may 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 65 illustrates the 3D plot with open parts. **Notes**: 1) open parts are not included in the hidden surface algorithm, so they are visible behind filled part surfaces, 2) open parts are displayed only when surfaces are plotted by part number.

24.2 Controlling the 3D view

Several interactive features are available When **MetaMesh** displays three-dimensional views. The philosophy and control set is almost identical to that of **Geometer**(Chap. 4). You use the mouse to change the viewpoint, as though you were moving around the assembly.

As shown in Fig. 82, the screen is divided into three regions:

- The main plot area is on the left-hand side.
- The information area on at the bottom-right displays text information on the plot.
- The orientation area at the top-right shows the current view. The red lines represent the Cartesian axes and the orange box is the approximate field of view.

The mouse controls the view when the cursor is inside the plot area. As a reminder, the cursor style changes from an arrow to a directional symbol. The plot area is divided into the five active regions shown in Fig. 25. To move closer to the objects, move the cursor into Region A (center of the plot), and briefly hold down the left mouse button. The orientation window shows the effect – the orange box shrinks with the narrowing view. The main plot is updated when you release the mouse button. To move the viewpoint outward, hold down the right mouse button with the cursor in Region A. Table 14 summarizes how the view changes with mouse actions in the different regions. Note that the normal mouse actions are active when the cursor is outside the plot area.

Wandering about in three-dimensional space may be confusing. The following menu commands are helpful to get reorientated:

+X VIEW
+Y VIEW
+Z VIEW
Move the viewpoint to a location on the $+x$, $+y$ or $+z$ axis.

Table 14: Changing the viewpoint of three-dimensional plots with the mouse

Region	**Button**	**Action**
A (center)	Left	Move toward the objects
	Right	Move away from the objects
B (right)	Left	Walk around the objects to the right
	Right	Shift position to the right
C (up)	Left	Walk around the objects toward the top
	Right	Shift position upwards
D (left)	Left	Walk around the objects to the left
	Right	Shift position to the left
E (down)	Left	Walk around the objects toward the bottom
	Right	Shift position downwards

RESTORE PLOT DEFAULTS

Set all plot parameters (*e.g.,* magnification, shifts, rotations) to the standard view that appears when you enter the 3D plot menu.

24.3 3D plot control menu commands

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

PLOT STYLE

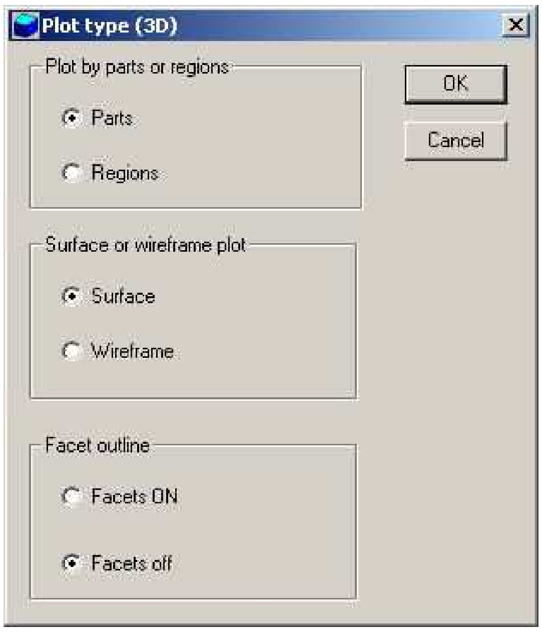
This command calls the dialog shown in Fig. 83. The radio buttons labeled *Parts* and *Regions* determine whether the program will pick facets and apply color-coding according to the part number or the region number of the elements. The buttons *Surface* and *Wireframe* determine the plot style. Figure 82 shows the surface plot style, while Fig. 84 shows a wireframe plot. The wireframe plot regenerates quickly but it may be confusing for complex assemblies. A useful strategy 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.

REGION/PART DISPLAY

The command displays the dialog of Fig. 85. The example illustrates one reason to include *Partname* and *Regname* commands in the script. The descriptive names are displayed in the dialog. The information may 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.23.2. You can adjust the perspective of the plot by



Plot type (3D)

Plot by parts or regions

[✓ Parts] Regions

Surface or wireframe plot

[✓ Surface] Wireframe

Facet outline

Facets ON [✓ Facets off]

Figure 83: Dialog to set the 3D plot style.

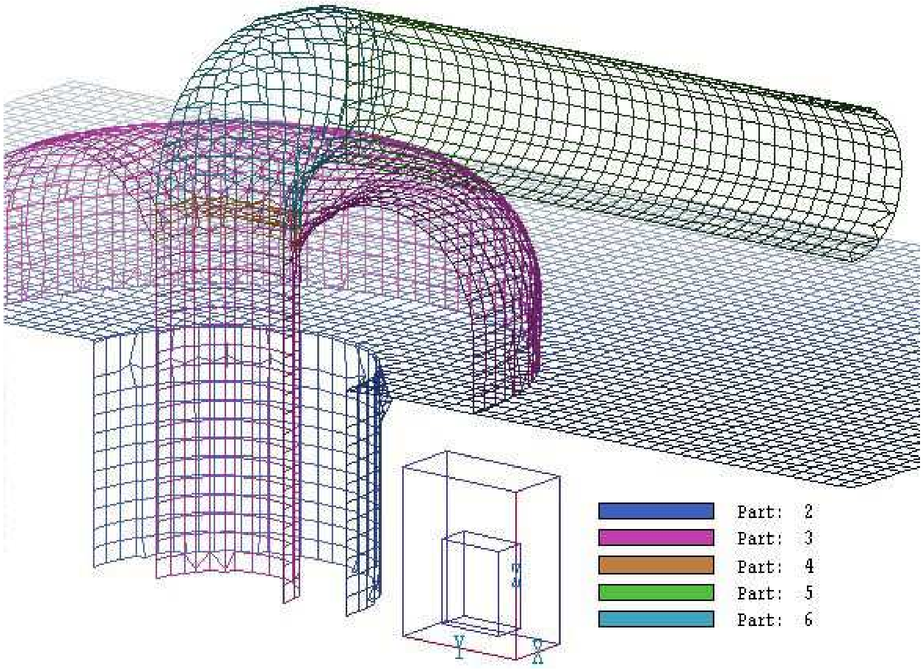
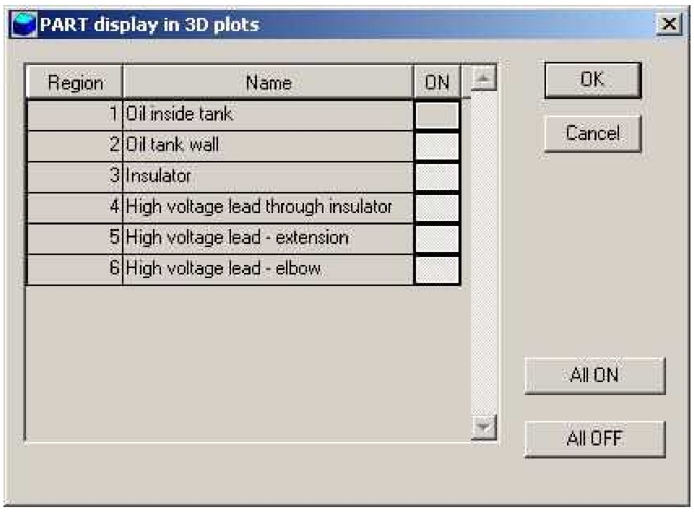


Figure 84: Example of a Wireframe plot.



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 85: Dialog to choose which parts or regions are displayed.

changing values of the parameter *DView*. A good choice is to set *DView* somewhat larger than 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 limit the display areas along the Cartesian axes. **MetaMesh** does not display facets that lie outside the limits. With this feature you can create cutaway views. When a mesh is processed, the default is to set the cut plane limits equal to the dimensions of the solution volume.

25 Formats of the MetaMesh output file

In the structured, conformal mesh of **MetaMesh**, 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,...).

MetaMesh output files have names of the form `FPREFIX.MDF`, where `MDF` signifies **MetaMesh** Data File. The string `FPREFIX` may contain up to 50 characters and should include no spaces. The **MetaMesh** binary output file has a simple and compact format. The following code extract comprises the entire output algorithm:

```
WRITE (OutMDF) 'BINARY'
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
DO NR=1,NRegs
  WRITE (OutMDF) Reg(NR).Name
END DO
```

The binary sequential file starts with the string `BINARY` (6 bytes) and the mesh size parameters *IMax*, *JMax* and *KMax* (4-byte integers). This is followed by $(I_{max}+1)(J_{max}+1)(K_{max}+1)$ data sequences containing the following quantities:

- The spatial coordinates of the node (x, y, z) (8 byte real)
- The region number of the node ($RegNo$) (4 byte integer)
- The region number of the associated upper element ($RegUp$) (4 byte integer)

The order of bytes for integers starts from the LSB (least significant bit) to the MSB (most significant bit). This order is sometimes referred to as *little endian*. Real numbers are recorded in IEEE T_floating format (little endian).. To conclude the file, **MetaMesh** records `NReg` region names (20 byte strings).

MetaMesh can also generate output files in text format for easy transfer of information to your own programs. The following excerpt illustrates the text file format:

```
TEXT
IMax:      20
JMax:      20
KMax:      40

    I       J       K          x             y             z       RegNo RegUp
==============================================================================
       0       0       0 -5.000000E+00 -5.000000E+00 -1.000000E+01     1     1
       1       0       0 -4.500000E+00 -5.000000E+00 -1.000000E+01     1     1
       2       0       0 -4.000000E+00 -5.000000E+00 -1.000000E+01     1     1
       3       0       0 -3.500000E+00 -5.000000E+00 -1.000000E+01     1     1
       4       0       0 -3.000000E+00 -5.000000E+00 -1.000000E+01     1     1
   ...

Region names
==========================
    1: Outer boundary
    2: Dielectric
    3: Inner electrode
    4: Outer electrode
```

Data entries on lines are separated by spaces. The first line is the string `TEXT`. In the next three lines, the second entry gives I_{max}, J_{max} and K_{max}. A total of N_{node} data lines follow after three title lines. Each data line contains the node indices (three integers in I8 format), the node coordinates (three real numbers in E14.6 format), the node region number (I6) and the element region number (I6). After a blank and a title line, the data lines give the region number (I6) and corresponding region name (20 byte string).

26 Importing parts from SolidWorks

Chapter 10 and Sect. 18.1 discussed the general topic of information transfer from solid modeling programs to **MetaMesh** via STL (stereo lithography) files. The chapter gives specific information on data export from the popular **SolidWorks** program[5]. There are several areas of interest:

- Optimizing the solid model for the best results in **MetaMesh**.
- Setting STL parameters in **SolidWorks**.
- Exporting from the **SolidWorks** assembly space to position multiple parts.

There is a close conceptual correspondence between the STL format, **SolidWorks** and **MetaMesh**. In the two programs, *Parts* are the basic building blocks. A *Part* is a contiguous solid body with a single surface. In a topological sense, the term *single surface* means that any two points on the surface may be connected by a path that lies on the surface. Within this definition, the surface of a *Part* may be quite complex, including thru holes and convolutions. The STL format defines a *Part* through a set of connected triangular facets that constitute a closed surface. A valid STL file may contain only a single *Part* (*i.e.*, one contiguous surface).

SolidWorks has two main working environments:

- *Part space*: fabricate and modify a single part in a local coordinate system.
- *Assembly space*: shift and orient multiple *Parts* to create a system with complex objects.

These environments mirror the *WorkBench* and *Mesh* spaces of **MetaMesh** (Chap. 16). The utility of a solid modeler for **MetaMesh** input depends on whether STL files may be created from information in both the *Part* and *Assembly* spaces.

- When an STL object is exported from the *Part* space, it describes the surface of the object but does gives information about the absolute position and orientation in an assembly. In this case, it is necessary to position the object in the mesh space using the `Shift` and `Rotate` commands of **MetaMesh**. This task may involve extended calculations and/or trial-and-error.
- If the *Assembly* and mesh spaces share the same coordinate system, export from the *Assembly* space gives absolute facet coordinates. In principle, *Parts* exported from the *Assembly* space may be directly entered in the **MetaMesh** script with no shifts or rotations. The advantage is that **SolidWorks** has sophisticated tools for placing objects relative to the coordinate system and to each other.

In export from the *Part* space, **SolidWorks** has several options to control the quality of the surface representation and generates a preview of the facets. It is possible to suppress unneeded details (*e.g.*, bolt holes, fillets,) so that they will not be included in the STL model. In export

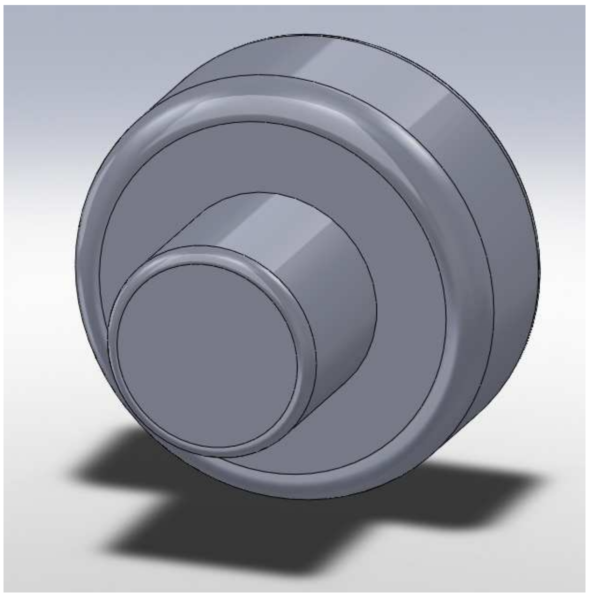


Figure 86: Test part viewed in the *Part* fabrication space of **SolidWorks**.

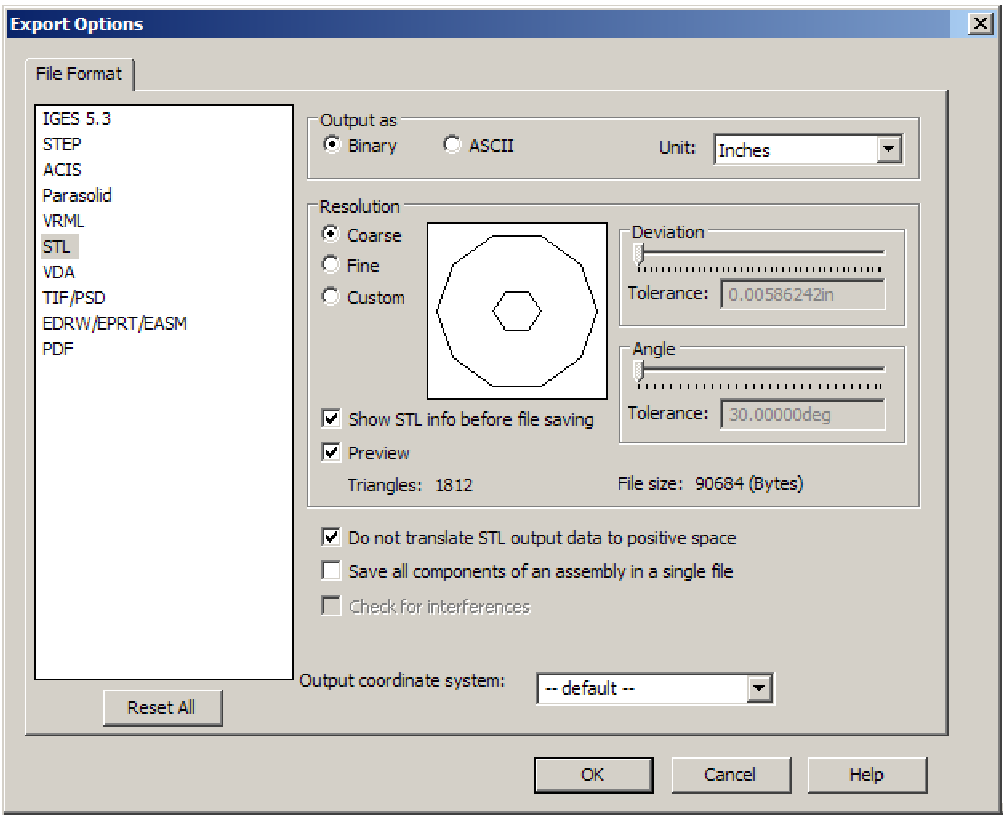
from the *Assembly* space, entire parts may be excluded. **SolidWorks** creates an individual STL file for each active part with the correct absolute coordinates.

To begin, consider export from the *Part* space of **SolidWorks** (Fig. 86). Choose *File/Save as* and pick *STL* in the file type menu. An *Options* button becomes active. Clicking on it brings up the dialog of Fig. 87 with several choices. You can save files in either *Binary* or *ASCII* format – **Geometer** and **MetaMesh** recognize both. Unless you want to introduce a conversion, be sure to set the *Units* to those in force during the construction of the part. To check or change the fabrication units, choose *Tools/Options/Document properties/Units*. Regarding *Resolution*, there is no reason to create STL facets that are smaller than the mesh element size. Initially, pick the *Coarse* option. There are two considerations if you pick the *Custom* option:

- A small value of the *Deviation* (compared to the element size) results in a large number of surface facets, slowing **MetaMesh** and increasing the chances of error.
- A small value of the *Angle* results in acute triangular facets. In this case, it may be necessary to raise the *CutOff* parameter discussed in Sect. 18.1 of the **MetaMesh** manual to obtain a valid mesh.

The option *Show STL file before saving* is not critical, but the *Preview* option may be valuable if you are adjusting the *Deviation* and *Angle*. When the box is checked, the program shows the view of Fig. 88. Be sure to check the box *Do not translate STL output data to positive space*. With this setting, **SolidWorks** will record the actual coordinates in the *Part* or *Assembly* spaces. Uncheck the box *Save all components of an assembly in a single file*. If this option were active, the program could combine two disconnected objects into one STL file. Such a file is invalid under the STL format definition and would be rejected by **MetaMesh** and most other STL software.

[5]We appreciate the use of materials and technical assistance from Dassault Systemes SolidWorks Corporation.



Export Options

File Format

IGES 5.3
STEP
ACIS
Parasolid
VRML
STL
VDA
TIF/PSD
EDRW/EPRT/EASM
PDF

Output as: [✓ Binary] ASCII — Unit: Inches

Resolution: [✓ Coarse] Fine Custom

Deviation — Tolerance: 0.00586242in

Angle — Tolerance: 30.00000deg

[✓] Show STL info before file saving

[✓] Preview

Triangles: 1812 — File size: 90684 (Bytes)

[✓] Do not translate STL output data to positive space

[ ] Save all components of an assembly in a single file

[ ] Check for interferences

Output coordinate system: -- default --

Figure 87: **SolidWorks** dialog to set STL file options.

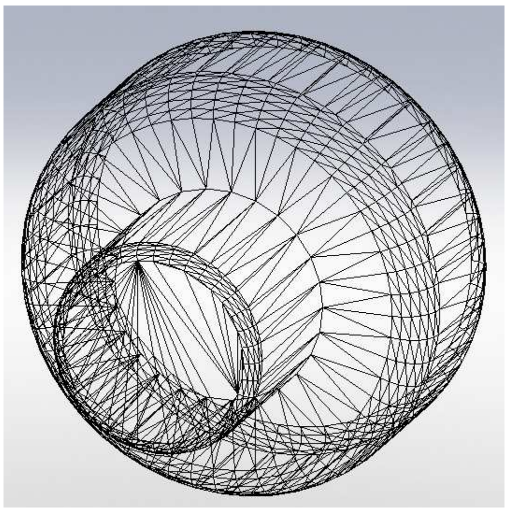


Figure 88: View of the test part when the *Preview* box is active.

It is useful to say a few words about what type of *Part* makes a good STL object for **MetaMesh**. One virtue of the part of Fig. 86 is that it does not have sharp edges. Sharp edges force **MetaMesh** to make difficult decisions about locating nodes. Furthermore, numerically-calculated electric and magnetic are generally inaccurate near geometric discontinuities. You should avoid sharp edges in both high-voltage systems and simulations of high-voltage systems. On the other hand, note that the fillets increase the number of facets required to represent the part (Fig. 88). Fillets, bolt holes and other small features increase the work of mesh generation and should not be included if they do not affect the field solution. Before exporting an STL file from the *Part* space, decide if features can be eliminated. Make selections from the features tree on the left and use *Edit/Suppress* to deactivate them. After you save the STL file, you can use *Edit/Unsuppress* to restore the features.

To conclude, there are several considerations for exporting STL files from the *Assembly* space:

- Work with a a copy of the assembly file and delete or suppress any parts or features that are not required for the field model.

- As with export from the *Part* space, choose *File/Save as* and pick *STL* as the file type. Be sure that *Do not translate STL output data to positive space* is checked and that *Save all components of an assembly in a single file* is unchecked. Note that the *Preview* option does not function in the *Assembly* mode.

- **SolidWorks** will create multiple STL files, one for each unsuppressed part. The nodes coordinates gives the absolute positions of facets in the assembly.

- Individual *Parts* may intersect (*i.e.*, share a common volume). The over-write principle in **MetaMesh** will sort everything out as long as *Parts* appear in the correct order in the **MetaMesh** script.

- **SolidWorks** creates default STL file names for individual *Parts*, a concatenation of the assembly name, the part name and the number of times the part is used in the assembly. These names generally contain multiple spaces and will cause problems in **Geometer** and **MetaMesh**. Edit the names of the STL files, removing spaces before loading them.

As an example, the bottom of Fig. 89 shows two instances of the part of Fig. 86 placed in a **SolidWorks** assembly. An export produces two STL files. These files are used in the definition of STL parts in **MetaMesh** with no shifts or rotations, resulting in the mesh shown at the top.

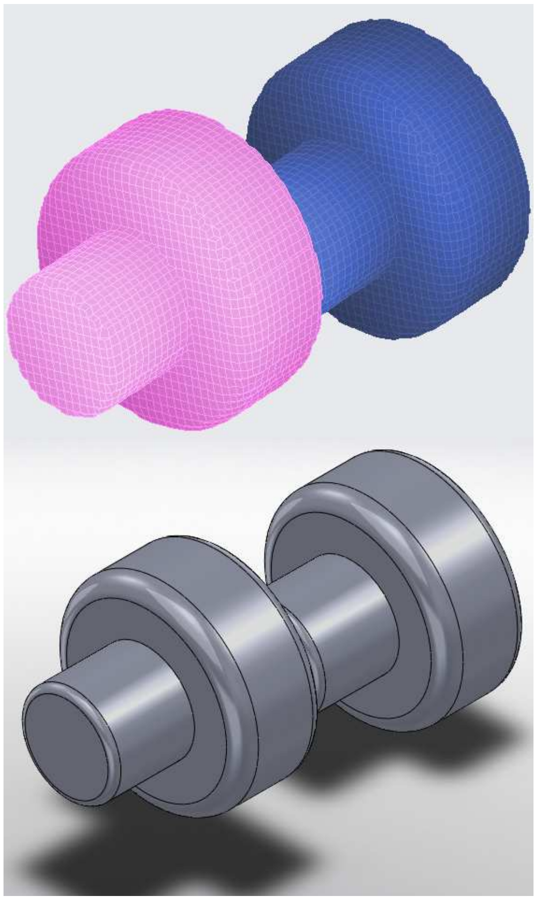


Figure 89: Representations of an assembly composed of two instances of the part of Fig. 86 in **SolidWorks** (bottom) and **MetaMesh** (top).

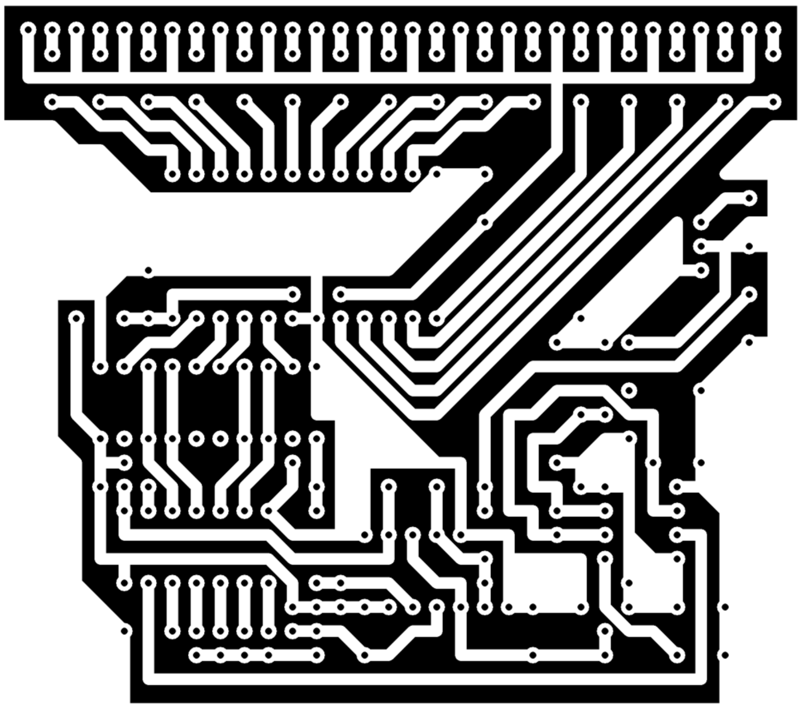


Figure 90: Printed circuit layout – conductors in white. File PC01.BMP.

27 MetaMesh - importing photographic images

Some three-dimensional objects have the form of a thin layer with relatively complex variations in the layer plane. One example is the printed circuit, as in Fig. 90. The image shows a layout of conductors (and possibly dielectrics) in a plane. A complete circuit may consist of several layers. Converting the conductors in Fig. 90 to STL objects or the native solid models of **MetaMesh** would be laborious. A simpler approach is to transfer the photographic bit pattern of directly to mesh elements in a layer of specified thickness. Parts of type PHOTO support this capability. They are useful whenever graphics files are available to define geometric variations in one or more slice planes. In addition to printed circuits, the feature is well-suited to medical images in the form of slices, such as MRI scans.

MetaMesh recognizes images in BMP, PNG and PCX format. The program analyzes pixels in the image in terms of *Lightness* or *Hue* values. Integer values of *Lightness* range from 0 (dark) to 100 (light). By convention, *Hue* is a cyclical quantity with units of degrees. Figure 91 shows the correspondence between spectral colors and units of *Hue*.

In the workbench space, a PHOTO type part is a box with dimension Δz_p in z and dimensions x_{pmin}, y_{pmin}, x_{pmax} and y_{pmax} in the x-y plane. The layer is centered at $z = 0.0$, covering the range $-\Delta z_p \leq z \leq \Delta z_p$. Like other parts, the orientation and position of a PHOTO part in the assembly space is determined by ROTATE and SHIFT commands. The function of the part is to assign enclosed elements of the foundation mesh to one or more regions according to values in the photographic image. The correspondence between *Lightness* or *Hue* and region number is defined by the PHOTOINTERVAL command. Processing of a PHOTO part involves the following

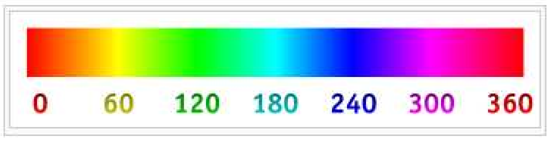


Figure 91: Values of *Hue* in degrees.

steps:

- **MetaMesh** loops through all elements of the foundation mesh, calculating the centroid coordinates of each element.
- The centroid coordinates in the assembly space are converted to workbench coordinates, depending on `SHIFT` and `ROTATE` parameters for the part.
- If the point is inside the `PHOTO` layer in the workbench space, the program projects the x-y coordinates to the pixel space of the image. If the *Lightness* or *Hue* value of the nearest pixel is within one of the intervals defined by the `PHOTOINTERVAL` command, **MetaMesh** assigns the corresponding region number to the element.

The following section discusses the **MetaMesh** script commands for `PHOTO` type parts. Section 27.2 reviews an application based on *Lightness* analysis, while Sect. {ssect:photohue covers an example based on *Hue*.

27.1 MetaMesh script commands for PHOTO parts

It is necessary to define intervals if any `PHOTO` type parts appear in the **MetaMesh** script. The following command structure appears in the `GLOBAL` section:

```
PHOTOINTERVAL [HUE,LIGHTNESS]
   ILow(1)  IHi(1)  RegNo(1)
   ILow(2)  IHi(2)  RegNo(2)
      ...
   ILow(NInt)  IHi(NInt)  RegNo(NInt)
END
```

The single string parameter (`Hue` or `Lightness`) determines how the image will be analyzed. The structure may include up to 250 data lines. Each data defines an interval and contains three integer numbers:

- The lower limit of the interval, *ILow*. For a *Lightness* analysis, the range is 0 to 100. For *Hue*, the range is 0 to 360.
- The upper limit of the interval, *IHigh*.
- The region number associated with the interval, *RegNo*.

Generally, *IHigh* should be larger than *ILow*. The exception is *Hue* values near 360°. **MetaMesh** recognizes the wraparound. For example, the choice $ILow = 350$ and $IHigh = 10$ defines a valid interval of 20° near the color red. The quantity *RegNo* must a number – the structure does not recognize symbolic region numbers set up by the `REGNAME` command.

PHOTOINTERVAL [Hue, Lightness]
PHOTOINTERVAL = Hue

This command signals the start of a photographic interval structure. The string parameter determines the analysis type for *PHOTO* parts. If the parameter does not appear, the default is *Lightness*. The command is followed by from 1 to 250 data lines consisting of three integer numbers. Each line contains the lower limit of the interval, the upper limit and the region number associated with the interval. The structure terminates with the `END` command.

A `PHOTO` type part is designated by the following form of the `TYPE` command in a `PART` structure:

TYPE PHOTO FileName XPMin YPMin XPMax YPMax
TYPE PHOTO MotorAssembly.BMP -20.0 -10.0 55.0 50.0

The string parameter is the full name of an image file. The file must be available in the working directory. The allowed types are `BMP`, `PNG` and `PCX`. The four real-number parameters define a rectangle in the *x*-*y* plane of the workbench space into which the rectangular image is mapped.MetaMesh commands!Type

The following characteristics apply to `PHOTO` type parts:

- The rectangle defined by x_{pmin}, y_{pmin}, x_{pmax} and y_{pmax} need not have the same proportions as the image.
- Mapping to the image is independent of the pixel dimensions. Nonetheless, the representation may be improved by picking resolution of the foundation mesh so that elements contain an integer number of pixels along the x and y dimensions.
- A single parameter in the `FAB` command defines the thickness of the layer, z_p. A `PHOTO` part may have any thickness. Region properties are uniform in the z direction of the workbench space. For thin `PHOTO` parts, the foundation mesh should contain a layer of elements with thickness z_p.
- The `NAME`, `ROTATE` and `SHIFT` commands have the standard functions common to all part types.
- The `REGION` command is ignored. Region assignments are defined in the `PHOTOINTERVAL` command. The `COAT` command is also ignored.
- Conformal fitting is not applied to `PHOTO` parts. The `SURFACE` command is ignored.

27.2 Analyzing an image according to lightness values

The monochrome image of Fig.90 is ideal for analysis via *Lightness* values – white pixels represent conductors and black pixels correspond to a dielectric. The example file `PC_DEMO.MIN` includes the following structure in the `GLOBAL` section:

```
PhotoInterval Lightness
   0   50  2
  50  100  3
End
```

Dark pixels will be assigned to Region 2 (dielectric) and light pixels to Region 3 (conductor). There are two `PART` structures in the script. The first creates a solution box to contain the structure (Region 1, air). The second `PART` structure has the form:

```
Part 2
  Type Photo PC01.BMP 0.00  0.00  50.00  50.00
  Name PC
  Fab 1.00
End
```

In response to the `TYPE` command, the image of Fig. 90 is loaded and mapped to a square space in the x-y plane with 50.0 mm sides. The layer has thickness 1.0 mm in z. The foundation mesh is defined with uniform element thickness of 0.5 mm, so the `PHOTO` object is two elements thick. Figure 92 shows the completed mesh in a slice normal to z.

Firstly, note that the element resolution in the mapping space (200 × 200) is considerably lower than the bitmap resolution of the original image (991 × 871). Therefore, the mesh representation is approximate. Secondly, in a *Lightness* analysis, avoid resizing images with photo editing software. The original image in Fig. 90 has either white (`RGB = FFFFFF`) or black (`RGB = 000000`) pixels. Depending on the algorithm, photo software may include gray elements in a reduced image for the best appearance.

27.3 Analyzing an image according to hue values

Analysis by *Hue* is generally more challenging. It may be necessary to adjust images with editing software like PhotoShop or Paintshop Pro before they can be used with **MetaMesh**. Figure 93 shows an acceptable image with well-defined regions of different hues. The *Hue* of the white background in the original image is undefined. Therefore, white pixels were converted to the specific hue, green. Note that green has minimal overlap with other colors in the image.

The question is how to pick the best *Hue* intervals. You can find the distribution of pixels according to *Hue* with image editing software. Also, **MetaMesh** records a distribution in terms of *Lightness* or *Hue* whenever a `PHOTO` image is loaded. Figure 94 shows a plot of the listing data. Blue and green regions are easily differentiated, but there is overlap between the red and orange regions. The following intervals were used in the example `HEART_DEMO.MIN`:

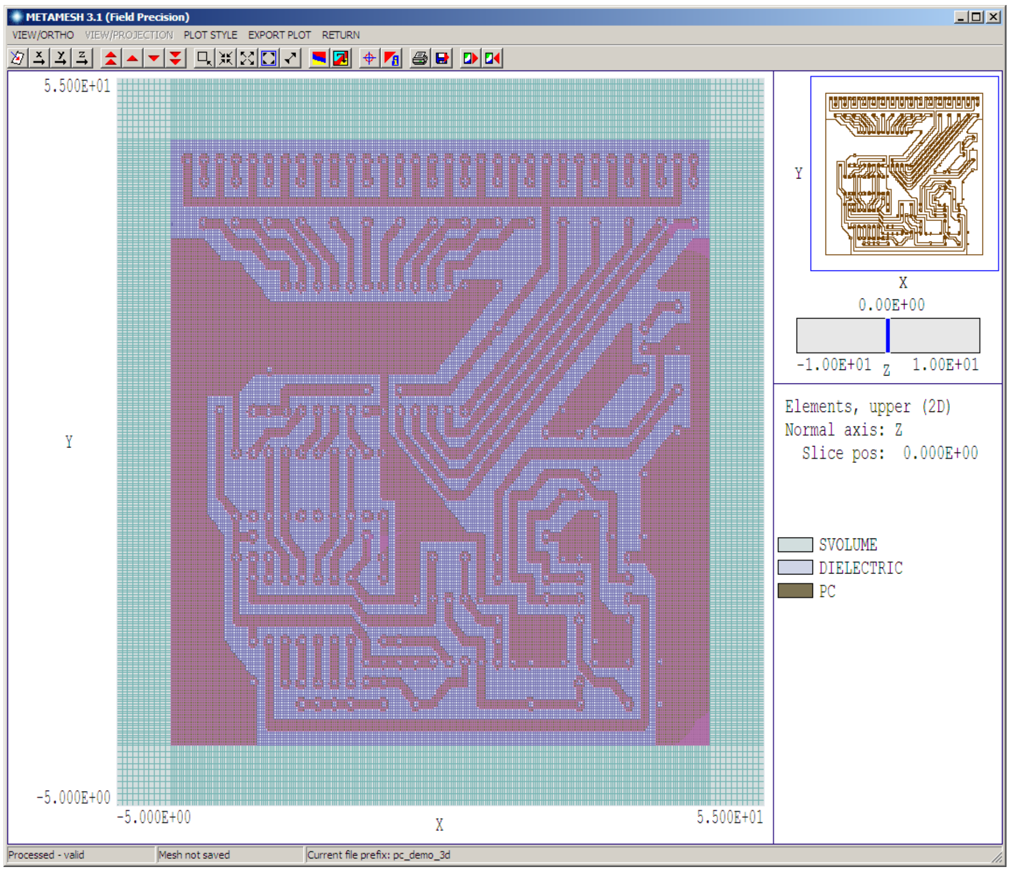


Figure 92: Completed mesh for example `PC_DEMO`. View in a plane normal to z of upper elements at $z = 0.0$ mm

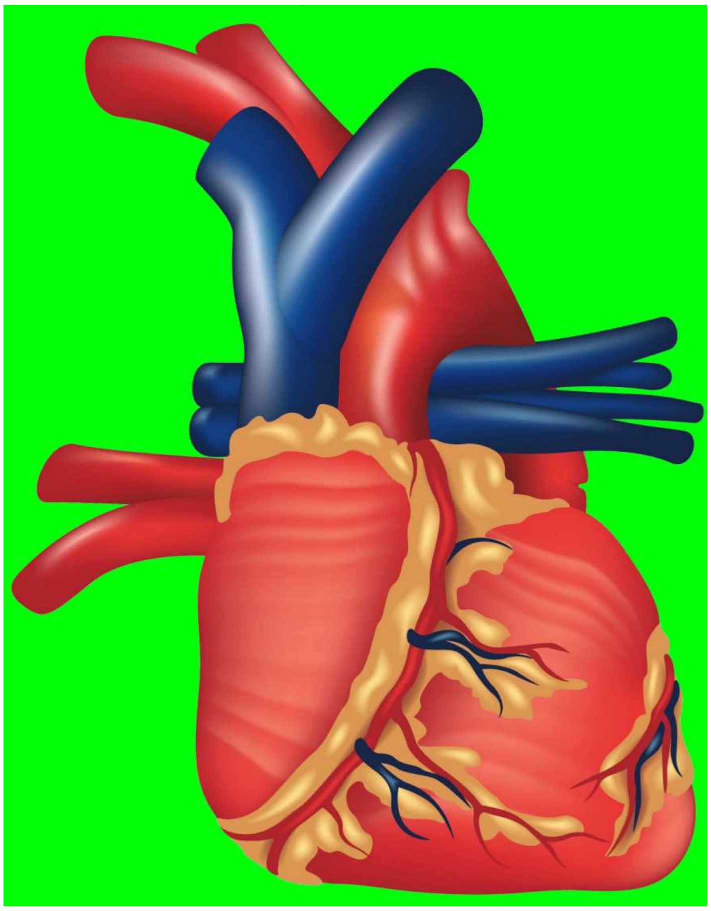


Figure 93: Test image of a human heart

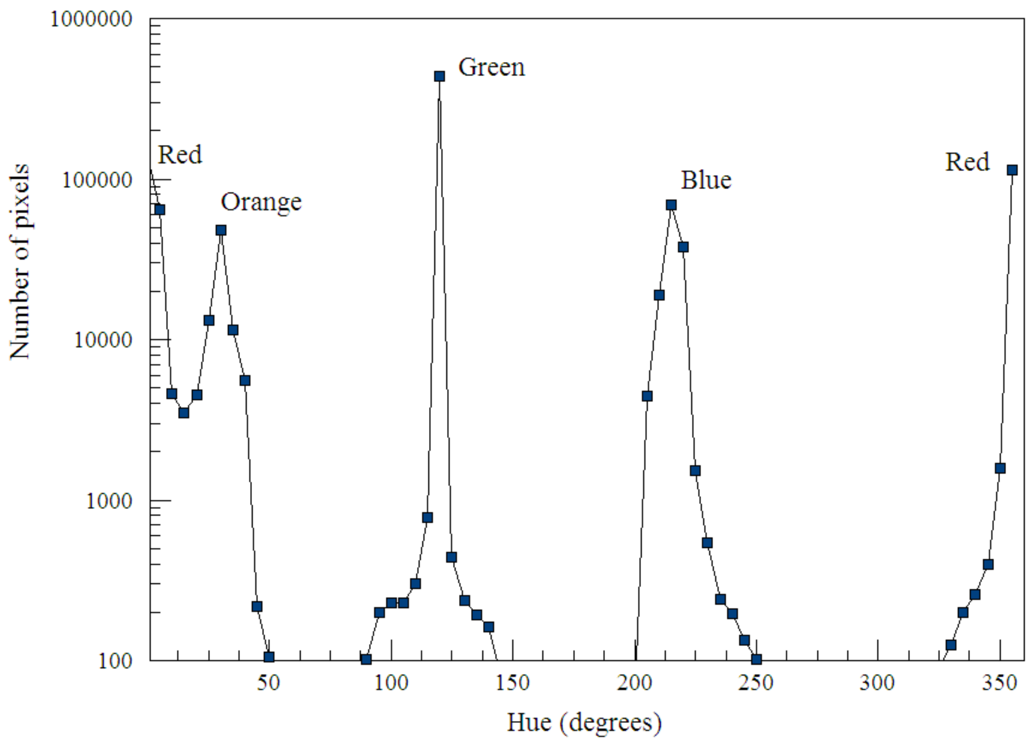


Figure 94: Number of pixels in the image of Fig. 93 as a function of *Hue* value.

```
*  RegName 2 Orange
*  RegName 3 Blue
*  RegName 4 Red
  PhotoInterval Hue
      15   80   2
     150  299   3
     300   15   4
  End
```

The intervals cover the entire *Hue* range except for the portion covered by the green background to ensure that all enclosed elements are assigned. Figure 95 shows the result. Note that some elements on the edges of Region 4 were assigned to Region 2, reflecting the overlap between red and orange (Fig. 94). There are two reasons for incorrect *Hue* assignment on some of the edges:

- Anti-aliasing on the object edge introduces variations.
- The original image was in `JPEG` format. The Fourier compression method gives fuzzy pixel values on sharp edges.

To conclude, the utility of the photographic method depends on whether there is a clear connection between image properties and the physical properties of the system (represented by different regions). In the case of the printed-circuit image (Fig. 90), there is a clear correspondence between *Lightness* and physical properties. The utility of the heart image is questionable. First, tissues with different *Hues* differ by function rather than physical properties. Second,

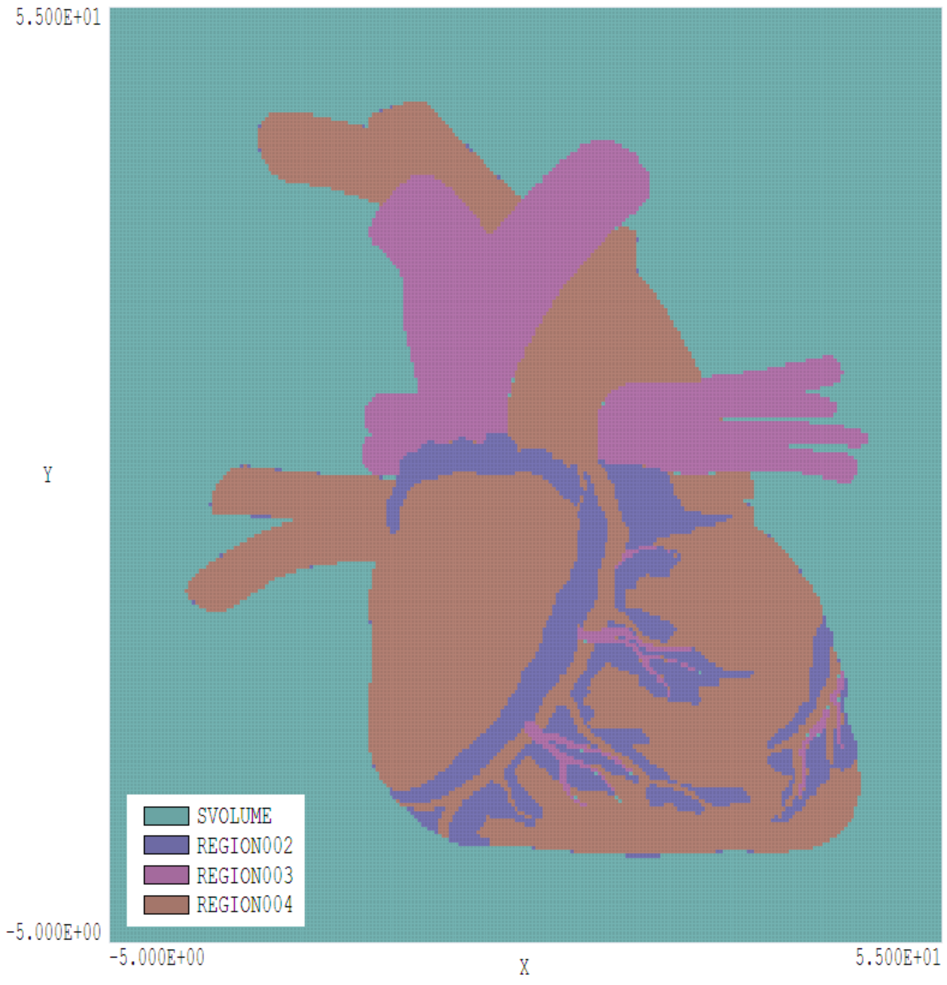


Figure 95: View of the mesh generated from Fig. 94.

there may be no purpose to projecting a three-dimensional rendition to a single plane. Better choices include MRI slices with clearly differentiated regions (like bone) or colored slices of a mechanical assembly prepared with a CAD program.

Index