Mesh 5.0

Universal Mesh Generator

Field Precision

PO Box 13595
Albuquerque, New Mexico 87192 U.S.A.
Telephone: 505-220-3975
FAX: 505-294-0222
E Mail: techinfo@fieldp.com
Internet: http://www.fieldp.com

Mesh 5.0 Manual - Contents

Chapter 1. Introduction

1.1. Program function

Mesh is the core of all **TriComp** applications. The program generates conformal triangular meshes for any two-dimensional system geometry. Figure 1.1 shows an example of such a mesh. The goal is to divide the system into a number of small pieces or *elements*. In this limit, the governing equation for the physical problem (such as Poisson's equation for electrostatics) reduces to a large set of coupled linear equations that can be solved easily on a computer. The underlying assumption is that physical properties of materials (such as the dielectric constant) are uniform over the volume of an element. Therefore, the boundaries of elements should correspond as closely as possible to the boundaries of physical objects. For example, the red and brown elements in Fig. 1.1 constitute electrodes while the green and light-blue elements represent dielectric supports. The dark blue elements represent vacuum. Usually mesh generation constitutes the main effort in a finite-element field solution. Once you understand **Mesh**, you can use any of the **TriComp** solution programs with little extra effort.

The philosophy of the **TriComp** programs is to automate processes but still to give you complete control over the solution. All input and output files are in ASCII format so you can check them with an editor. The task is much easier if you have a full-featured editor rather than a utility like Notepad. Therefore we have included the freeware package SynEdit for simultaneous editing of multiple files. All **TriComp** programs also have built-in editors. Simulations for different solution types follow the same basic steps. The first task is to prepare a **Mesh** control script that describes the geometry of your application. The easiest way to create a script is to use the built-in drawing editor of **Mesh**. You can create a drawing from scratch or import information from DXF files generated by CAD programs. You can also write or modify scripts directly using a text editor. With the script information, **Mesh** creates a file of geometric specifications that can be used by a solution program to create a file of field values. Each package includes a post-processor designed for the specific physical solution. The post-processors have extensive capabilities to create plots and to perform analyses. Table 1 summarizes the program names and functions for the **TriComp EStat** package.

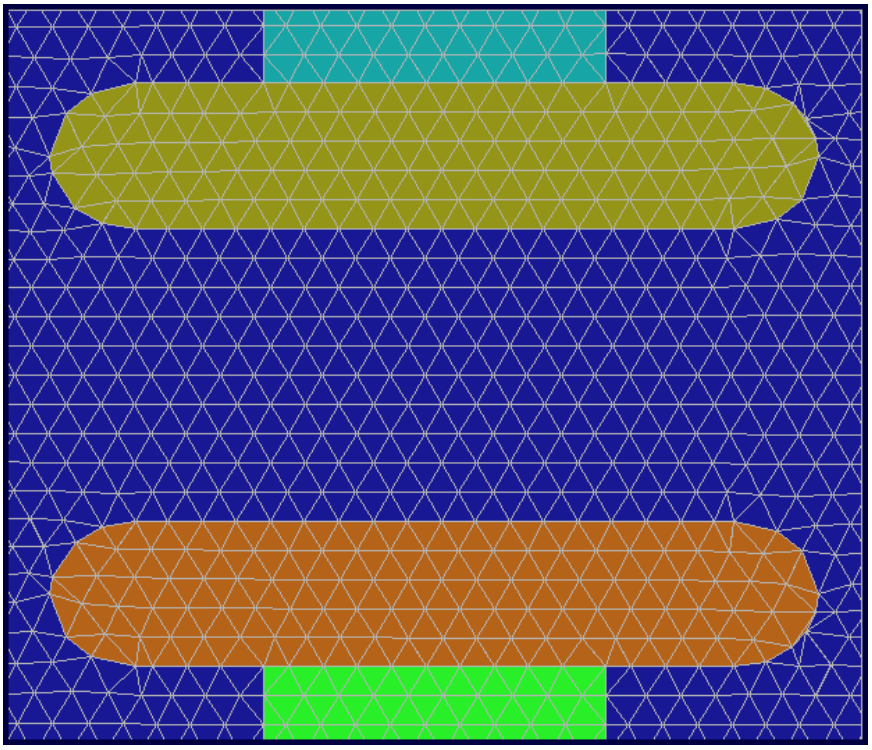

Figure 1.1. Example of a conformal mesh

Table 1. Programs of the TriComp EStat package	
Program name	**Function**
TC.EXE	Utility to launch individual **TriComp** programs and to organize data
MESH.EXE	Conformal mesh generator with drawing editor
ESTAT.EXE	Solution program generates values of electrostatic potential on the mesh
VESTAT.EXE	Post-processor to analyze **EStat** solutions
FPBATCH.EXE	Utility for automatic control of complex sequences of program operations

This manual covers operation of the **Mesh** program. There are also manuals for individual solution programs. Operational of the post-processors is covered in the solution program manuals. This manual is organized to help you start creating your own meshes quickly. Early chapters introduce basic techniques while advanced topics are covered in latter chapters. The remainder of this chapter discusses installation and testing of the program. We will step through an example and review the components of the mesh generation package. Chapter 2 addresses essential techniques and terminology for triangular meshes. Chapter 3 reviews the structure of the **Mesh** control script. A script is an ASCII file with a set of commands to control program operation and to define your system geometry. All **TriComp** programs run from scripts with a standard format. Chapter 4 describes operation of **Mesh** in the interactive mode. The program has advanced visualization capabilities that allow you to check mesh integrity and to make repairs if necessary. Chapter 5 describes the Drawing Editor and the preparation of DXF files with CAD programs. Chapter 6 covers advanced techniques to create meshes with *variable resolution*. In this case mesh elements are concentrated in critical regions for maximum accuracy and minimum run time. The chapter also covers ways to solve problems in mesh generation.

1.2. System requirements

TriComp 5.0 requires a Pentium-class computer with at least 64 MB of RAM and an SVGA graphics screen. The 32-bit applications run under Windows 95/98/ME/NT/2000/XP or Linux. The programs use dynamic memory allocation, so the number of elements is limited only by the installed memory. A computer with 128 MB of memory can handle over 1,000,000 elements. The programs of a complete solution package generally occupy less than 5 MB on a hard disk. At least 30 MB of free storage is recommended for temporary data files. **Mesh** and the **TriComp** postprocessors export hardcopy through the Windows print drivers to any installed device.

1.3. Installation

Field Precision programs are self-contained. They make no changes in the Windows registry and will not over-write essential driver files or dynamic link libraries. Therefore you can install and remove any of the **TriComp** programs without fear of changing your Windows setup. Field Precision programs will not affect the operation of software already on the system.

On a hard disk, make a directory (folder) \TRICOMP and subdirectories \TRICOMP\EXAMPLES, \TRICOMP\MANUALS and \TRICOMP\BUFFER. The directory \TRICOMP\EXAMPLES is a convenient place to store examples supplied with **Mesh** and different solution programs. The directory \TRICOMP\MANUAL is a good place to keep PDF instruction manuals, and \TRICOMP\BUFFER is used for temporary storage of input and output data files.

Copy the executable files MESH.EXE and TC.EXE from the distribution CD to \TRICOMP. Also copy the online instruction manual MESH50.HTML to this directory. Please copy and read the file TERMS.TXT which describes license requirements for using **TriComp**.

Copy the file DTEST.MIN to /TRICOMP/BUFFER. This file is a sample input that we will use for the walkthrough test. Finally, copy the files MESH50.PDF (this manual) and TC50.PDF to \TRICOMP\MANUAL.

You can run any of the **TriComp** programs from the **TC** program launcher. The discussions in this manual assume you are using this method. To begin, set up a Windows shortcut to **TC.** For example, run Windows Explorer and go the directory \TRICOMP. Drag the logo for TC.EXE to the desktop or *Start Menu* directory to make a shortcut. Run **TC** and check the *Program* and *Data* directories listed by the program. For the standard setup, the *Program* directory should be \TRICOMP and the data directory should be \TRICOMP\BUFFER. If this is not the case, click on the *Change Program Directory* or *Change Data Directory* command. Set the correct location in the dialog. The settings will be recorded and reloaded next time you use **TC.**

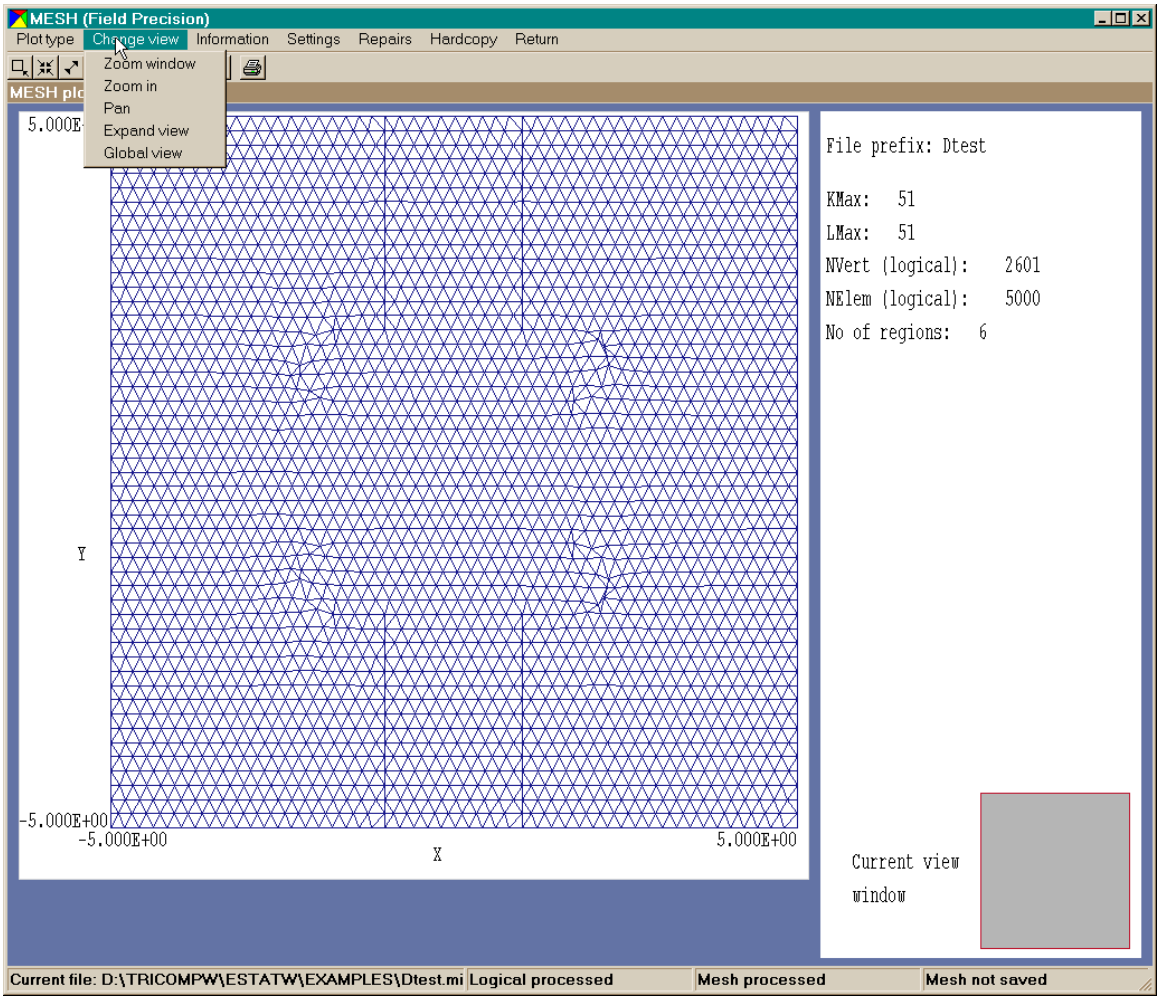

Figure 1.2. Mesh in the interactive mode, screen shot

1.4. Walkthrough example

After installation, run a test to verify that the programs are installed correctly. Run **TC** and launch the **Mesh** program. (Note that you can also run `MESH.EXE` directly without using **TC**.) From the *File menu*, choose *Load script (MIN)*. If necessary, change to the directory `\TRICOMP\BUFFER`. The dialog box should show one available file, `DTEST.MIN`. Pick the file and click on *OK*.

Command files are ASCII scripts with a simple language to define any two-dimensional geometry. All script files have the suffix `MIN` (**M**esh **IN**put). You can create and modify the files directly with an ASCII editor, or you can use the Drawing Editor to generate input graphically. You can inspect the content of `DTEST.MIN` by picking

Edit current script (MIN) command on the *File* menu. The command opens a full-featured Windows text editor. Exit the editor to return to **Mesh**.

The main function of **Mesh** is to convert the geometric information in the script file to a conformal triangular mesh that closely follows material boundaries. From the *Process menu*, pick *Process mesh*. The program opens a text window and displays information on the analysis. When the job is completed, the program displays the message *Mesh generation successful*. Press any key or click the right mouse button to continue. During processing, **Mesh** records the screen information in the listing file `DTEST.MLS` (**M**esh **L**i**S**t), so don't worry about missing anything. You can inspect this file with the *Edit current listing (MLS)* command of the *File menu*.

Next, click on *Plot-repair* menu. You should see the display of Fig. 1.2. At this point, you can experiment with commands to change the plot style and limits. Chapter 4 gives a detailed description of the program capabilities and Chapter 5 describes the drawing editor. Note that the *Default printer* command in the *Export plot* sends a copy of the current plot to the *default* Windows printer.

Chapter 2. Mesh generation concepts

2.1. Mesh basics

Field solutions by finite-element methods are based on the division of the volume of interest into small pieces called *elements*. For two-dimensional solutions, the most useful element shape is the triangle. The collection of triangle areas and vertices (or *nodes*) is called the *computational mesh*. Mesh generation in **TriComp** consists of two operations: 1) filling a region of space with ordered elements of appropriate size and 2) shifting the vertices of triangles (*nodes*) so that the element faces lie on material boundaries. We refer to the first operation as generating a *foundation mesh* and to the second as fitting physical boundaries to create a *conformal mesh*.

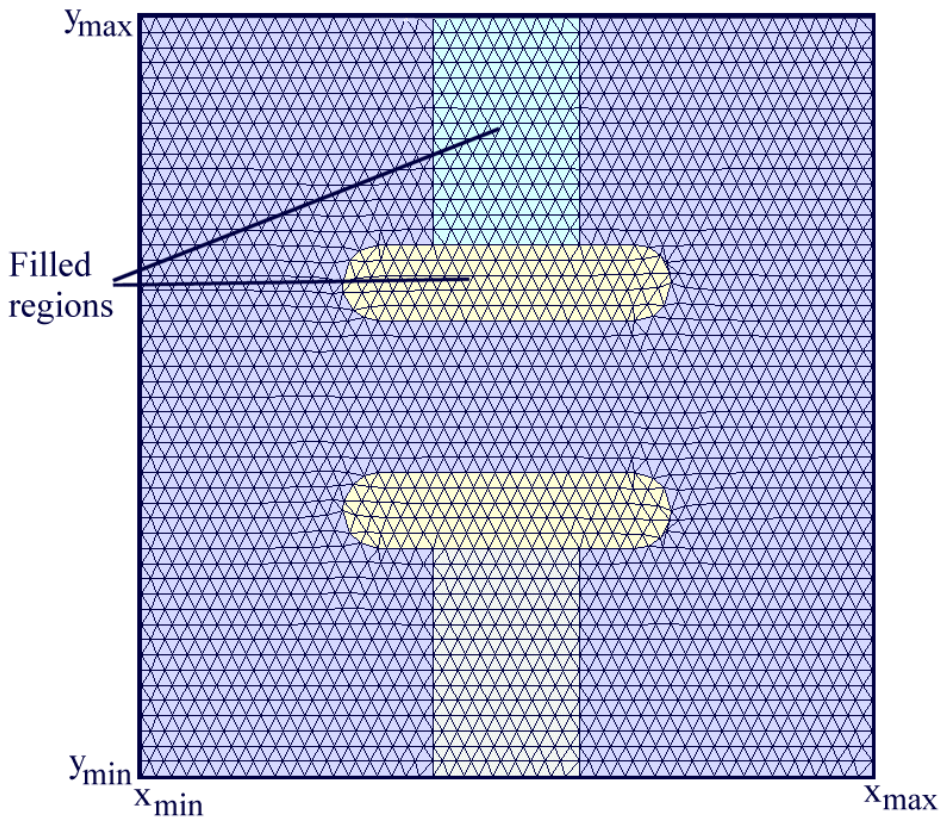

Figure 2.1. Computational mesh showing elements, regions, and solution volume limits.

Figure 2.1 defines terms used in mesh generation. It shows an example (`DTest`) of a mesh for an electrostatic problem where two long plate electrodes are suspended in a grounded box by dielectric supports. Nodes and elements are assigned to different *regions* of the problem: the vacuum space, the electrodes, the dielectric supports, and the grounded boundary. Note that the node locations have been chosen so that several of them lie on region boundaries. Therefore, the sides of triangles follow the boundaries and it is clear which triangle corresponds to which material. In electrostatic solutions, the finite-element method yields values for the potential at the nodes under the assumption that material characteristics are uniform over the triangle areas. The function of **Mesh** is to take geometric information about your system and to arrange nodes so that the triangles closely fit the object shapes. It is easy to deal with simple symmetric systems – the challenge is to devise a generator that can reliably handle arbitrary geometries.

Mesh is controlled by an *input script*, a text file similar to a BASIC program. You set program switches and specify the geometry by issuing commands and supplying parameters. You can also create files graphically using the Drawing Editor or a CAD program. Even if you use the graphic approach for most of your work, there are three reasons to understand the structure of the input script:

- The discussion gives insight into effective mesh generation techniques.
- You can edit the command file directly to make small changes or to check for possible drawing errors.
- You can modify the file to activate advanced features like variable mesh resolution

2.2. Foundation mesh

Mesh starts by creating a foundation mesh of approximately uniform triangles that fill a rectangular area large enough to contain the solution. Figure 2.2 shows an example. The active solution area may or may not fill the entire rectangle. Because the foundation mesh is logically structured, we know the indices of the nodes and how to identify the neighboring nodes and elements. **Mesh** analyzes the vectors of the boundary of each region you define and moves selected nodes so that they lie on given points, lines or arcs. The node displacements must be small enough so that all sets of three connected points in the foundation mesh form valid triangles. The mesh generation procedure may fail if the foundation mesh is poorly suited to the object boundaries. Therefore, it is necessary to pick good dimensions for the triangles and sometimes to introduce variations in triangle size (*variable resolution*). For a complete discussion of the mathematical basis for **TriComp** mesh generation, see S. Humphries, Jr., **Field Solutions on Computers** (CRC Press, Boca Raton, 1997), Sects. 5.2 and 5.3.

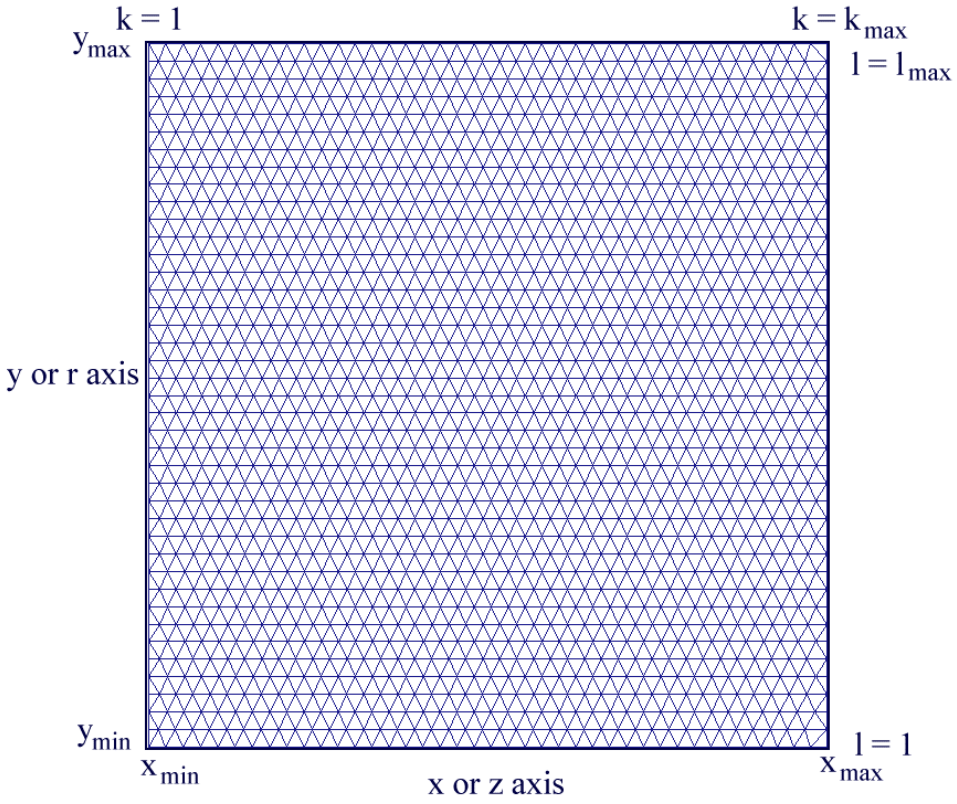

Figure 2.2. Foundation mesh parameters

2.3. Coordinate limits

The foundation mesh rectangle extends from x_{min} to x_{max} along the horizontal axis and from y_{min} to y_{max} along the vertical axis (Fig.2.2). The vertex indices are in the range $1 \le k \le k_{max}$ along *x* and $1 \le l \le l_{max}$ along *y*. Because extra points must be added to implement boundary conditions, the total number of points in the foundation mesh is $i_{tot} = (k_{max}+2)(l_{max}+2) - 1$.

All **TriComp** solution programs can find solutions in cylindrical coordinates for systems with azimuthal symmetry. In this case, the horizontal axis represents the *z* direction and the vertical axis represents *r*.

2.4. Regions

Regions are areas of the solution space where nodes and possibly elements have common properties (Fig. 2.1). For example, in electrostatic problems regions can represent electrodes, dielectrics, space-charge clouds, constant potential boundaries, or symmetry boundaries. In thermal transport problems, regions may be fixed temperature bodies, heat sources, or volumes of different thermal conductivity, mass and heat capacity. **Mesh** assigns an integer number from 1 to 127 to each node and element of a region. Numbers are assigned in the order that the regions appear in the script file. These numbers are associated with specific physical properties only in the solution programs. Therefore, **Mesh** output files are compatible with all **TriComp** solution programs.

2.5. Filled and open regions

There are two types of regions in **Mesh**: *filled* and *open*. The boundary of an open region may be any collection of points, lines and arcs. In this case, **Mesh** assigns the current region number in the processing sequence to nodes along the region boundaries and does not renumber any elements. For example, in electrostatic calculations open regions often represent symmetry or fixed potential boundaries. An open region consisting of a set of unconnected points can be used to simulate a fixed-potential grid. Lines and arcs of open regions may be connected or unconnected at their end points. The one limitation is that the vectors of a single region cannot cross each other.

Filled regions represent solid bodies like electrodes or dielectrics. Here, line or arc vectors define a closed curve in *x-y* or *r-z* space. After processing the boundary, **Mesh** assigns the current region number to all elements and nodes enclosed by the boundary.

To process a point, **Mesh** moves the nearest *unclamped* node to a location on the region boundary and sets the node region number. It then *clamps* the vertex position so that it is not moved by subsequent operations. For lines and arcs, **Mesh** firsts displaces and clamps nodes at the start and end points. The program then walks from the start to the end along a path that moves between logically-connected points. In the process, the program displaces and clamps points that are closest to the vector. This process gives a series of triangular elements with sides aligned along the boundary vector.

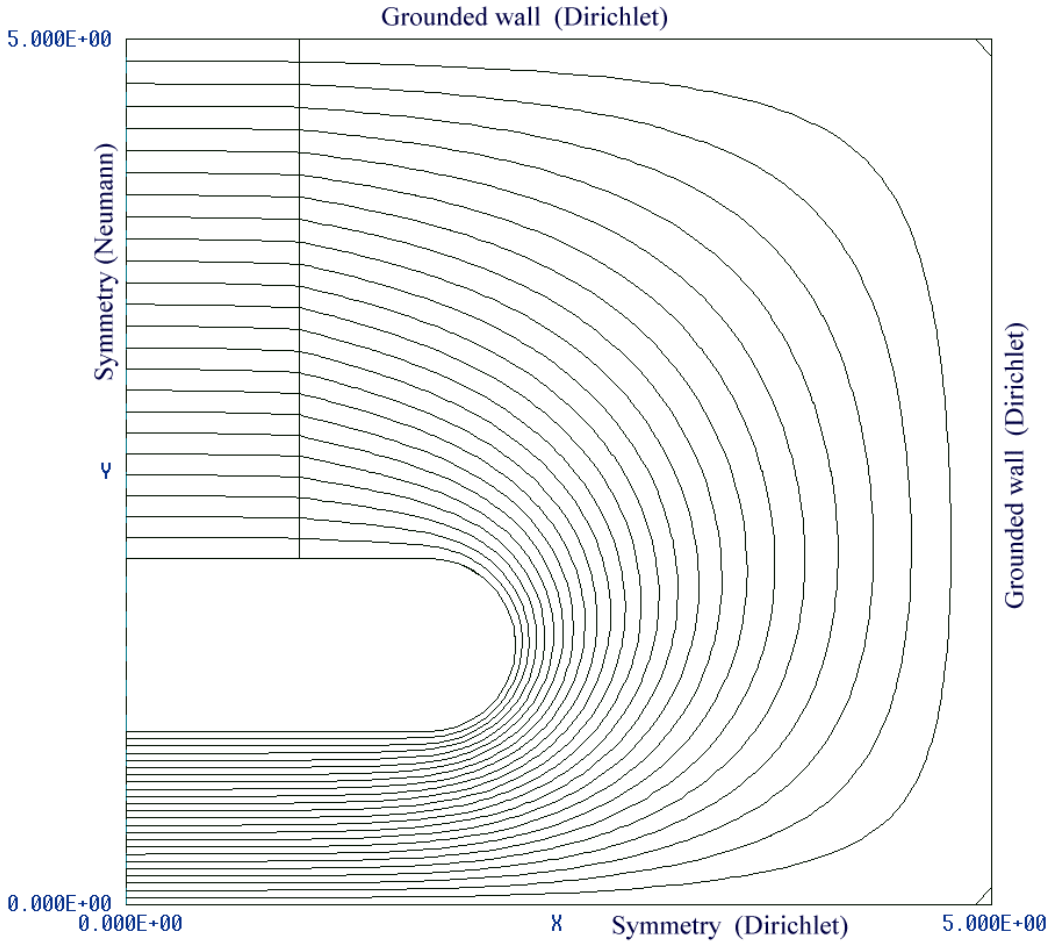

Figure 2.3. Symmetry conditions, electrostatic model of one quadrant of a system

2.6. Dirichlet and Neumann boundary conditions

One function of an open region is to set a condition along a solution boundary. There are two classes of boundaries for field solutions: *Dirichlet* and *Neumann*. The Dirichlet condition states that the quantity to be evaluated has a specified value along a surface. Previous versions of **TriComp** supported only the special case where the quantity has a constant value. Version 5.0 has new features to support spatial variations along Dirichlet boundaries. In electrostatic solutions, the equipotential surface of a metal electrode is a Dirichlet boundary. In thermal transport, a highly conductive body has uniform temperature.

The generalized Neumann condition implies that the normal derivative of the calculated quantity has a specified value along a surface. The **TriComp** programs utilize a special case of the Neumann condition

where the derivative equals zero. In electrostatics, this means that the electric field is parallel to the boundary. The condition is often used to enable treatment of half of a symmetric system (Fig. 2.3). In thermal transport a Neumann boundary represents a perfect insulator (zero heat flux through the surface). One of the useful features of the finite-element method is that the specialized Neumann condition applies along any unspecified external boundary. Therefore, the Neumann condition can be implemented on slanted and curved boundaries. **Because Neumann boundaries occur naturally, you need only define Dirichlet sections on the periphery of the solution volume.**

2.7. Order of regions

The order in which regions appear in the script file is important because the currently-processed region over-writes any previously-defined region numbers for overlapping nodes or elements. For example, suppose we wanted to calculate electrostatic fields around a circular wire with an insulating jacket. The procedure is first to define a circular filled region to represent the insulating dielectric. This is followed by a region that defines a smaller circle inside the first to represent the constant-potential wire cross-section. All elements inside the small circle will assume the second region number. The nodes on the wire boundary will have the fixed potential condition because the wire region was entered last.

The over-write process requires that regions with special boundary conditions should be entered in the script last so that nodes on the boundary assume the correct region number. For example, suppose we have a solution where an electrode and dielectric share a common surface. The electrode region must appear *after* the dielectric so that nodes on the common surface retain the number of the constant-potential region. If you observe a solution that appears to have non-physical field lines, the likely cause is incorrect region order in the mesh input file.

2.8. Region definition example

The example of Figure 2.4 clarifies how to set up regions in **Mesh**. In the electrostatic solution program, the top and bottom electrodes are set to +1500 V and -1500 V respectively. To begin, note that the

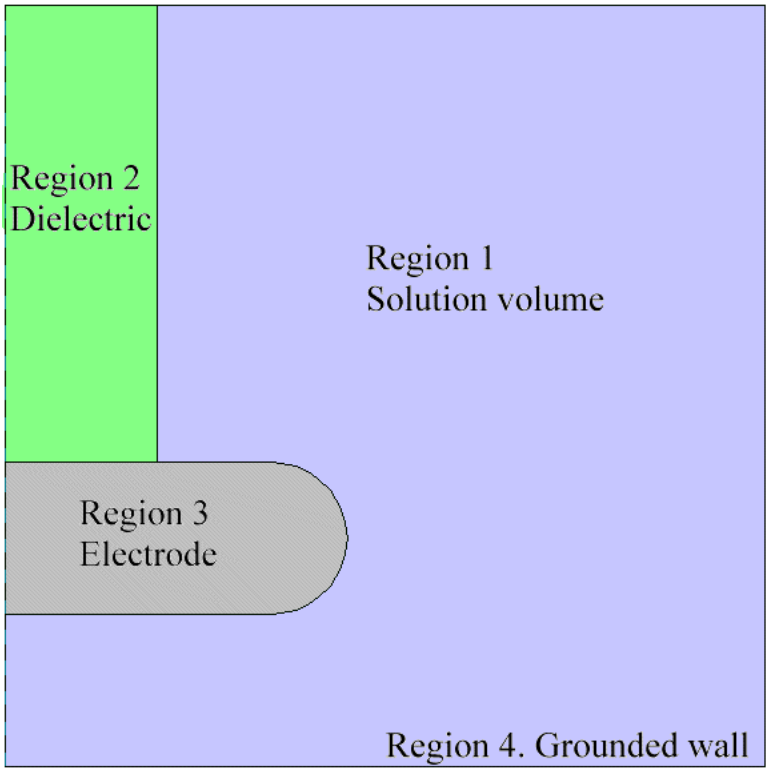

Figure 2.4*a*.

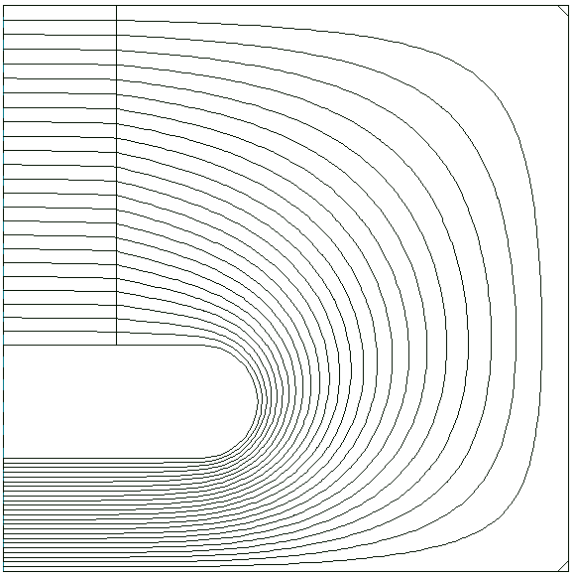

Figure 2.4*b*

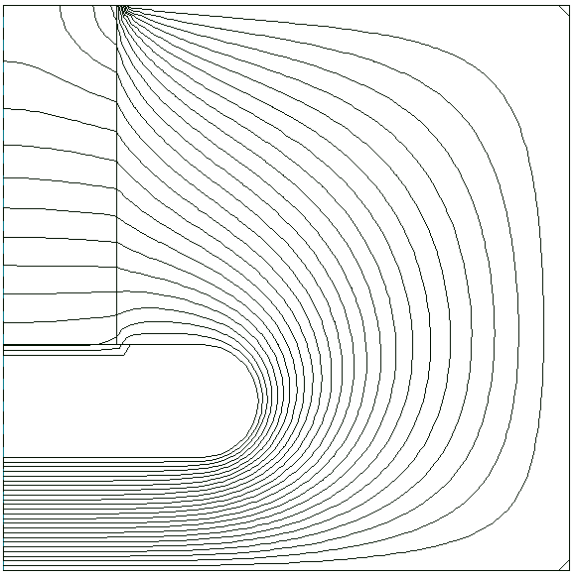

Figure 2.4*c*

problem has two symmetry planes. The potential at the position (*x*,-*y*) must be the negative of that at (*x*,*y*). Therefore, the potential must satisfy $\phi = 0.0$ in the plane $y = 0$ (Dirichlet condition). Similarly, the potential at (-*x*,*y*) equals that at (*x*,*y*). Therefore, the condition in the plane $x = 0$ is that $\partial\phi/\partial x = 0$ (specialized Neumann condition). Therefore we can solve for the potential in the first quadrant only using Dirichlet and Neumann conditions on the *x* and *y* axes and then infer the potential in the other quadrants from the symmetry conditions. Figure 2.4*a* shows the modified geometry with boundary conditions and region numbers.

The general procedure is first to enter all regions that do not have fixed boundary conditions (*i.e.*, dielectrics) and then to enter Dirichlet boundaries and fixed potential regions. In this way, important boundaries are not accidently over-written. The first step is to set elements in the foundation mesh that represent the vacuum region with $\epsilon_r = 1.0$. Region 1 is a filled region that occupies the entire foundation mesh rectangle. The second step is to add the dielectric support as Region 2. This is a closed region inside Region 1. **Mesh** shifts boundary vertices that have not already been fixed and writes the number 2 on the nodes and enclosed elements. Region 3 is the fixed potential electrode. The vertices on the shared boundary with the dielectric are changed to fixed potential points. The unspecified points along the left-hand boundary automatically satisfy the specialized Neumann condition. The final step is to define an open region (Region 4) to set the Dirichlet condition ($\phi = 0$) along the bottom, right and top boundaries. Figure 2.4*b* shows an equipotential plot of the correct solution, while Figure 2.4*c* shows the non-physical solution when the dielectric support region is accidently placed at the end of the command script.

Chapter 3. Input script

3.1. Script file structure

The **Mesh** command file is an ASCII script that you can create and modify with a text editor. You can also generate the file graphically with the Drawing Editor (Chapter 5). The file must have a name of the form `FPrefix.MIN`. The string `FPrefix` is the run name, a descriptive title from 1 to 20 characters in length. The run name identifies all associated data files for mesh generation, solution and analysis. The input script has the following structure:

```
Global
 (Global commands)
End

Region [RegName1]
 (Boundary vectors)
End

Region [RegName2]
 (Boundary vectors)
End

...

Region [RegNameN]
 (Boundary vectors)
End

EndFile
```

There is one `Global` section with commands that affect the entire solution and there are up to 127 `Region` sections with geometric information on each physical object in the problem. The `Global` section must appear first in the file. Its function is to set program parameters and characteristics of the foundation mesh. The `End` command marks the end of global input. The `Global` commands can be entered in any order. **Mesh** reads commands until encountering the `End` command and then carries out the operations. The `Region` command marks the beginning of region boundary information that continues to the `End` command. Finally, `EndFile` signifies that all regions have been entered.

Mesh reads the commands with a free-form parser. A line consists of a command and one or more parameters separated by any number of delimiter characters. Valid delimiters are *Space, Comma, Tab, Colon, Left parenthesis, Right parenthesis* and *Equals sign*. You can use any of these characters to give your input files a distinctive style. You can also include indentations for readability. Commands and keywords can be entered in upper or lower case. **Mesh** ignores blank lines and comment lines. A comment line starts with the characters '*' (asterisk) followed by any set of characters. Use comment lines to document runs, recording the purpose of a run and a summary of the results. Real number parameters can be entered in any valid format. The following forms are allowed:

```
2.3456
2.63E12
-1.95E+02
5
```

The last number is interpreted as 5.0. You can also include any amount of text after the `EndFile` command. Here no asterisk is necessary. As an illustration, Table 2.1 shows the input file for the example of Figs. 2.4.

Table 2. Mesh input script example

```
* File DTQUAD.MIN
GLOBAL
   XMesh
     0.000000E+00   5.000000E+00   9.999999E-02
       End
   YMesh
     0.000000E+00   5.000000E+00   9.999999E-02
       End
   Graphics: Partial
END

REGION  FILL Vacuum
* Vacuum
  L  0.00000E+00  0.00000E+00  5.00000E+00  0.00000E+00
  L  5.00000E+00  0.00000E+00  5.00000E+00  5.00000E+00
  L  5.00000E+00  5.00000E+00  0.00000E+00  5.00000E+00
  L  0.00000E+00  5.00000E+00  0.00000E+00  0.00000E+00
  END

REGION  FILL Dielectric
* Dielectric
  L  0.00000E+00  5.00000E+00  1.00000E+00  5.00000E+00
  L  1.00000E+00  5.00000E+00  1.00000E+00  2.00000E+00
  L  1.00000E+00  2.00000E+00  0.00000E+00  2.00000E+00
  L  0.00000E+00  2.00000E+00  0.00000E+00  5.00000E+00
 END

REGION  FILL Electrode_Up
* Upper electrode
  L  0.00000E+00  2.00000E+00  1.75000E+00  2.00000E+00
  L  0.00000E+00  1.00000E+00  1.75000E+00  1.00000E+00
  A  1.75000E+00  1.00000E+00  2.25000E+00  1.50000E+00  1.75000E+00  1.50000E+00
  A  2.25000E+00  1.50000E+00  1.75000E+00  2.00000E+00  1.75000E+00  1.50000E+00
  L  0.00000E+00  1.00000E+00  0.00000E+00  2.00000E+00
 END

REGION Gnd_Bound
* Grounded boundary
     L  0.00000E+00  0.00000E+00  5.00000E+00  0.00000E+00
     L  5.00000E+00  0.00000E+00  5.00000E+00  5.00000E+00
     L  5.00000E+00  5.00000E+00  0.00000E+00  5.00000E+00
   END

ENDFILE
```

3.2. Basic foundation mesh definition

The *XMesh* and *YMesh* commands in the *Global* section specify the foundation mesh geometry. They define the limits of the solution rectangle and give the triangle size along each axis. The *XMesh* command signifies that a list of triangle properties will follow. The *End* command signals the end of the list. In this section we shall consider the simplest form with a single data line:

```
XMesh
 -6.00 10.00 0.25
End
```

The parameters in the data line are

```
Xmin  Xmax  Dx
```

for rectangular problems or

```
Zmin  Zmax  Dz
```

for a cylindrical geometry. The data in the example specify that the solution rectangle extends from -6.00 to 10.00 along the *x* axis with an approximate triangle base size of 0.25 units. For geometries of moderate complexity, pick *Dx* to give about 100 elements along the axis. The values can cover any range as long as $x_{max} > x_{min}$.

Similarly, the *YMesh* statement defines mesh properties along the *y* axis.

```
YMesh
 3.25  10.95 0.10
End
```

In this case the parameters are

```
Ymin  Ymax  Dy  (rectangular)
Rmin  Rmax  Dr  (cylindrical)
```

For cylindrical problems, values of $r_{min} < 0$ give non-physical results. The subsequent solution program will give an error message for negative values of *r*.

Spatial dimensions are flexible in **TriComp**. You can enter spatial quantities like x_{max} and y_{min} in any convenient set of consistent units. When you run a solution program, you can enter a conversion factor

that changes coordinates from **Mesh** into SI units (meters). When preparing **Mesh** input, pick units that keep the dimensions in the unity range for easy interpretation of the file. Examples are cm, microns, or miles.

3.3. Defining regions

After **Mesh** processes the *Global* commands and sets up a foundation mesh, it is ready for sequential processing of region sections that define the problem geometry . The *Region* command that marks the beginning of a region definition has two forms.

```
Region
Region Fill
```

You can also include a trailing region name of up to 24 characters to help document runs.

```
Region Fill UpperPlate
```

Note that the delimiters (including spaces) listed in Sect. 3.1 may not be included as part of the region name. Therefore, *Focus_electrode* is valid but *Focus electrode* is not. In the latter case, **Mesh** would assign the region name *Focus*.

Mesh assigns numbers sequentially to regions as they appear in the file. The nodes along the region boundaries are marked with the region number when they are shifted. The program creates a list of region numbers and names in the listing file in the form of comment lines. You can paste this information into the input script of the solution file to document the associated of physical properties with region numbers:

```
Assignment of region numbers
   Number of regions in the file:   6
   Region    Region
   number    name
   ================
*       1    VACUUM
*       2    ANODE
*       3    CATH_SUPPORT
*       4    FOCUS_ELEC
*       5    CHAMBER_WALL
*       6    CATHODE_SURF
```

The *Fill* keyword designates that the region should be filled. After processing the boundary vectors, sorting them, and checking that they define a closed curve, **Mesh** marks all internal nodes and elements with the current region number.

A complete region section may look like this

```
Region Fill Upper_Pad
 XShift: 2.5
 YShift: 0.2
* Upper pad, 1.0 microamp source
     A -5.0 30.0  0.0  35.0  0.0  30.0
     A  0.0 35.0  5.0  30.0  0.0  30.0
     A  5.0 30.0  0.0  25.0  0.0  30.0
     A  0.0 25.0 -5.0  30.0  0.0  30.0
End
```

It consists of 1) the *Region* command, 2) any number of comment lines, 3) the optional *XShift*, *YShift* and *Rotate* commands, 4) one or more arc, line or point vectors that define the boundary, and 5) an *End* statement. The following sections describe how to enter points, lines and arcs.

The *XShift* and *YShift* commands have the following syntax:

XShift xs

YShift ys

They cause a transformation of coordinates for all point, line and arc data of the region according to the formulas:

x(program) = x(file) + xs,

y(program) = y(file) + ys.

The commands are useful for geometries with repeated structures. You can simply copy and paste the vector data lines for one or more regions and use the *XShift* and *YShift* commands to position them. Note that **Mesh** does not provide automatic clipping. A region shifted with points outside the solution volume will generate an error message.

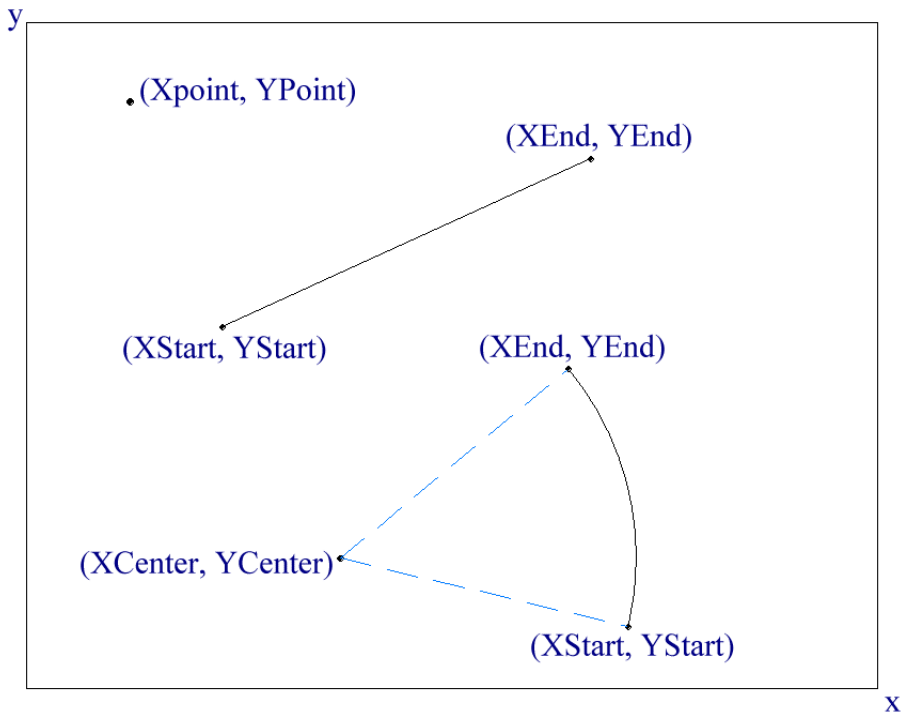

Figure 2.5. Parameters for points, lines and arcs.

The *Rotate* command has the form:

Rotate Ang [xc yc]

The quantity *Ang* is the rotation angle in degrees. A positive rotation is in the counter-clockwise direction. The optional parameters *xc* and *yc* define the center point for the rotation. The default values are *xc* = 0.0 and *yc* = 0.0. As with the shift commands, the *Rotate* command acts on all vectors in the region section and the program does not automatically implement clipping. Note that the rotation is performed *before* shift operations.

3.4. Points

A *point* data line moves and marks a single node. The line has the format

```
P  0.1678 10.45
```

The letter *P* designates the vector type. The two real-number parameters (*XPoint* and *YPoint*) are the (*x,y*) or (*z,r*) coordinates of the desired node location (Fig. 2.5). The point position must be inside the solution rectangle. Be sure to use the same distance units as those in the *XMesh* and *YMesh* statements. Because zero-dimension points are inherently disconnected, they are not allowed in the definition of a *Filled* region boundary. You can include up to 2000 individual points in a region. Points can be combined with lines and arcs in an *Open* region. In electrostatic solutions, arrays of points are useful to simulate grids.

3.5. Lines

The following command defines a straight line

```
L  0.571 0.986 1.756 -0.234
```

The four real-number parameters are

```
XStart YStart XEnd YEnd
```

the coordinates of the starting and ending points (Fig. 2.5). The line must fit inside the solution rectangle. If the solution program handles cylindrical problems, then the quantities are interpreted as

```
XStart ➔ ZStart
XEnd ➔ ZEnd
YStart ➔ RStart
YEnd ➔ REnd
```

3.6. Arcs

The statement

```
A  0.500 1.000 1.000 0.500 0.500 0.500
```

defines an arc of a circle. The six real-number values are

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

the coordinates of the starting, ending and center points. Figure 2.5 shows the definition of the points. **Mesh** gives an error message if the start and end points are outside the solution rectangle but does not check whether all points on the arc are valid.

For a given set of start-end-center points there are two possible arcs with positive and negative angles. **Mesh** always picks the arc that spans less than 180°. Use two or more entries to define arcs with spans in the range 180° to 360°.

3.7. Filled region boundaries

The components of *Open* regions need not be connected and can be entered in any order. In contrast, the boundary vectors of a *Filled* region must define a closed, continuous curve. The endpoints must meet to within the tolerance distance discussed in Chapter 6. You can enter the vectors in any order. **Mesh** has a powerful sorting feature that reorders vectors so that the start point of a vector matches the end point of the previous one. When the program detects the keyword *Fill*, it picks a starting point and then attempts to walk around the boundary, looking for matching vectors and reversing start and end points when necessary. The resulting set of vectors with corrected order is recorded in the listing file, `FName.MLS`. **Mesh** issues an error message if it does not return to the starting point of a filled region.

Chapter 4. Running Mesh in the interactive mode

4.1. File menu

There are two ways to run the program `MESH.EXE`: 1) as a non-graphical program under control of the **GCon** program or from the command prompt and 2) as in interactive graphics program. The first mode enables runs of **Mesh** and **TriComp** solution programs under batch file control. In this case you can set up your computer to perform

an extended series of operations autonomously. **Mesh** runs in the non-graphical mode if a file prefix is supplied when the program is called. For example, suppose you type

```
\TRICOMP\MESH \DATA\IFACE <Enter>
```

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

```
REM Variation of focus electrode postion
REM Processing FELEC01
START \TRICOMP\MESH \GUNDESIGN\FELEC01\FELEC01
START \TRICOMP\ESTAT \GUNDESIGN\FELEC01\FELEC01
REM Processing FELEC02
START\TRICOMP\MESH \GUNDESIGN\FELEC02\FELEC02
START\TRICOMP\ESTAT \GUNDESIGN\FELEC02\FELEC02
REM Processing FELEC03
START \TRICOMP\MESH  \GUNDESIGN\FELEC03\FELEC03
START \TRICOMP\ESTAT \GUNDESIGN\FELEC03\FELEC03
REM Job completed
```

The GCon manual covers operation of the program and preparation of batch files in detail. In the remainder of this chapter we shall discuss operation of **Mesh** in the interactive graphical mode. This mode is invoked whenever the program is called with no command line parameter. Figure 4.1 shows a screen shot of **Mesh**. When the program starts, only the *File, Drawing* and *Help* menus are active. Other menus become active when data is loaded and processing takes place. Chapter 5 covers the *Drawing* menu. We shall begin by discussing the commands of the file menu.

Load script (MIN)

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

Load mesh (MOU)

Load a previously-processed mesh file in the format described in Appendix A. You can inspect the mesh and make modifications.

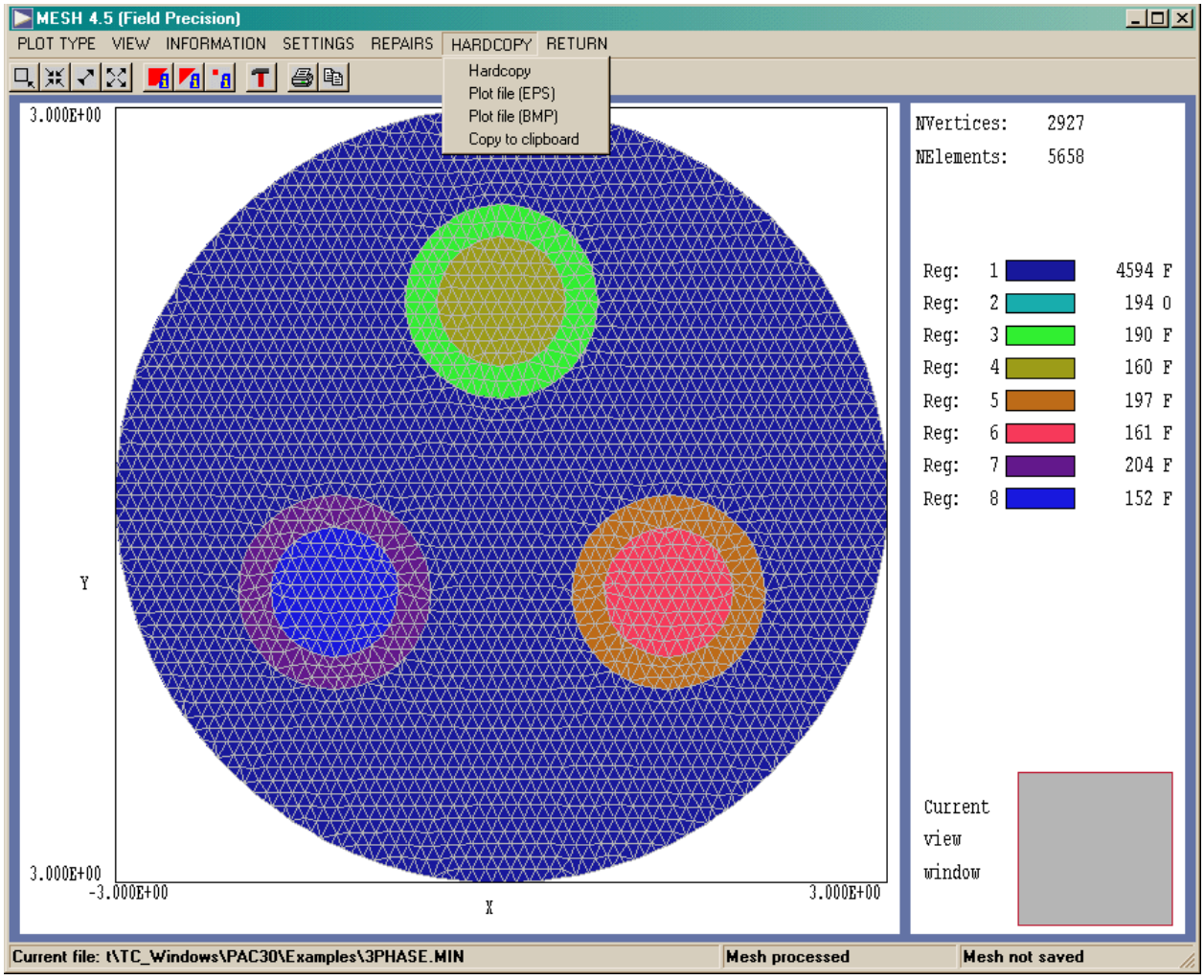

Figure 4.1. Screen shot of **Mesh**

Save mesh (MOU)

Save the current mesh in the format described in Appendix A. The output file has a name of the form `FPrefix.MOU`, where `FPrefix` is the prefix of the currently loaded input script file. This 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 current script (MIN)

Open a full-featured Windows editor to view or to modify the current input script (`FPrefix.MIN`). This command is active only if a script file has been loaded. If mesh processing has been

unsuccessful, the cursor of the editor is located at the line with the first error.

Edit current listing file (MLS)

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

Edit file

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

4.2. Process command

The *Process* command initiates the mesh generation procedure. The program analyzes the input script, generates a foundation mesh and fits region boundaries. Status information is displayed on a text screen and recorded in the current listing file (`FPrefix.MLS`). When processing is complete, press the right mouse button or any key to continue. The *Process* command is active only when a script file has been loaded.

The script file contains all information on the foundation mesh and region boundaries. A run usually consists of the following steps.

- Load the script file with the *Load script (MIN)* command
- Process the mesh with the *Process* command.
- Optionally, inspect or modify the mesh with the commands of the *Plot menu*.
- Save the mesh with the *Save mesh (MOU)* command.

The status bar reports the state of the program during these operations.

4.3. Help menu

You can view an HTML version of this manual (without figures) within the program by clicking on the *Mesh manual* command of the *Help menu*. The file `MESH50.HTML` must be in same directory as the executable program `MESH.EXE`. The command runs your default browser and loads the file. Alternatively, you can run Adobe Acrobat

Reader and keep the PDF version of this document in the background. You can activate the *Plot menu* when a mesh has been processed. In this menu, you can make screen or hardcopy plots of the full mesh or of a specified area. You can also correct mesh generation errors. This chapter and Sects. 4.5-4.8 describe the commands of the *Plot menu*.

4.4. Plot menu - plot types

The plot screen has two areas:

Main plot area
The plot area shows a scaled plot of the mesh with limits and an optional grid.

Legend area
The area to the right of the plot window constains parameters of the mesh, a legend for the color-coded regions, and an orientation box to show the limits and location of zoomed views. The following commands change the orientation box: *Zoom window, Zoom in, Pan, Expand* and *Global view.*

In the *Plot types* menu, you can choose between three different styles:

Mesh plot
Basic plot of element boundaries.

Region plot
Plot of elements color-coded by region number with the option to display element boundaries (Fig. 4.1). This plot is useful for checking whether numbers have been correctly assigned to *Filled* regions.

Node plot
Plot of nodes color-coded by region number. This plot is useful for checking whether region numbers have been correctly assigned to nodes, particularly along shared boundaries.

4.5. Plot menu - change view

The commands in this menu change the limits of screen and hardcopy plots. The commands are identical to those used in the **TriComp** post-processors.

Zoom window

Zoom in on 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 can enter coordinates for any point via the keyboard by pressing the *F1* key. Press the right mouse button or the *ESC* key to exit.

Zoom in

Narrow the view around the current center of the plot.

Pan

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

Expand

Expand the view around the current center of the plot.

Global view

Expand the view to show the full solution volume.

4.6. Plot menu - information

The commands of this menu are used to get information about entities in the mesh.

Node info

Move the mouse close to a node and click the left button. The identified node is highlighted and the following information appears in a dialog: coordinates (*x*,*y*), indices (*k*,*l*) and region number (*RegNo*). Click *OK* to continue.

Element info

Move the mouse into an element and click the left button. The identified element is highlighted and the following information appears in a dialog: center-of-mass coordinates (*x,y*), area and region number (*RegUp* or *RegDn*).

Region info

Move the mouse close to the boundary of a region and click the left button. The identified region is highlighted and the following information appears in a dialog: area, region number and status (*Filled* or *Open*),

4.7. Plot menu - settings

The commands of *Settings* menu change the appearance of the plot and the operation of the mouse. The menu includes the following commands:

Toggle snap

Mouse snap coordinates are a useful feature in the **TriComp** graphics programs. When the snap mode is active, the mouse returns the coordinate values closest to an integer multiple of the quantity *DSnap*. In other words, if *DSnap* = 0.5 and the mouse position is [5.4331,-2.6253], the returned coordinates are [5.5,-2.5]. Snap mode is useful for the *Zoom window* and *Pan* operations. The snap mode is turned off automatically when using any of the *Information* commands.

Set DSnap

Set the distance scale for the mouse snap mode.

Keyboard/mouse

By default, coordinate input in the *Plot menu* is through the mouse. You can temporarily switch to keyboard mode in any pointing operation by pressing the *F1* key. This command sets and resets a flag that changes the default coordinate method.

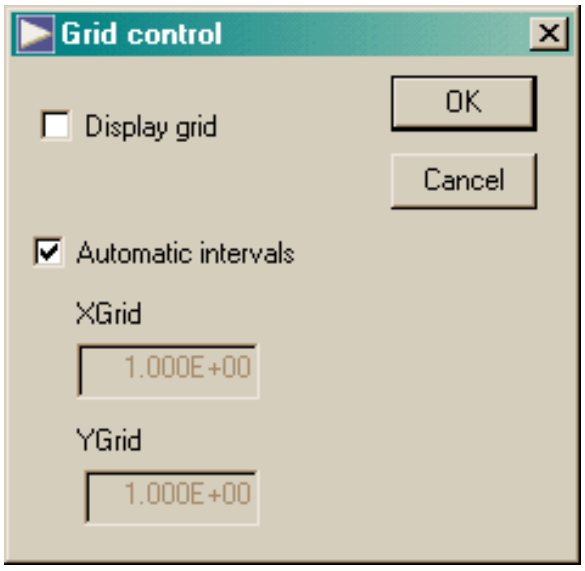

Figure 4.2. Grid control dialog

Grid control

This command initiates a dialog (Fig. 4.2.) to control the display of a reference grid in the three types of **Mesh** plots. Put a check in the *Display grid* box to activate the feature. By default, **Mesh** uses the *Automatic intervals* mode. In this mode the program chooses intervals so that the grid lines lie at convenient locations (*i.e.*, 0.02, 0.05. 0.10, ...). The grid spacings change automatically with zoom operations. The intervals are listed in the information area on the right-hand side of the screen. You can also enter intervals manually by removing the check in the *Automatic intervals* box and typing values in *XGrid* and *YGrid*. In this case, the intervals remain fixed under zoom operations.

Toggle element outline

Element boundaries in *Region* plots may coalesce for very fine meshes. This command turns element outlines *ON* and *OFF*.

4.8. Plot menu - repairs

There are three types of errors that may occur during mesh processing.

Type 1. Syntax errors, non-matching intervals in the *XMesh* and *YMesh* commands, open boundaries in *Filled* regions,...

Type 2. Inability to find a logical path between the start and end points of a line or arc vector.

Type 3. Inverted elements.

Type 4. Incorrect region number assignment to vertices or regions.

The first two types of errors are fatal and halt processing of the mesh. Errors of the first type can be corrected using the *Edit Script MIN* command. If a syntax error condition occurs, the editor starts with the cursor in the line where the error was detected. The second type of error occurs when the shape of elements in the foundation mesh is ill-suited to the lines or arcs that constitute one or more region boundaries. The program identifies the vector that was being processed when the problem occurred and writes suggested corrections in the listing file. These errors usually require changes in the script. Chapter 6 describes causes and cures for logical path errors.

The third or fourth types of errors can be corrected by changing the input script or by directly modifying entities in the processed mesh using the commands of the *Repairs* pulldown menu. The first option is preferable because the changes will be implemented in any regeneration of the mesh. As described in Chapter 6, inverted elements occur when the foundation mesh is inappropriate for one or more region boundary vectors. These errors occur rarely in Version 5.0 of **Mesh** because of the new dual-pass mesh fitting process. Region numbers may be incorrectly assigned for complex geometries where the endpoints of several vectors in different regions share the same *y* coordinate. Again, this error has been largely eliminated in Version 5.0.

The following commands are useful when there are a few stubborn errors. Be sure to save the mesh after correction and to preserve the `FPrefix.MOU` file for future runs. The first three commands correct Type 3 errors.

Relax bad nodes

Mesh displays an error message if there are inverted elements and marks the associated nodes in red on *Mesh* and *Region* plots. Set the plot style to *Region* and zoom in on the affected area. Then click on the *Relax bad nodes* command one or more times. The operation usually untangles the bad elements. You can then complete repairs with the *Move node* command.

Relax box

Occasionally, you may have to relax additional nodes surrounding bad elements to untangle the mesh. After selecting this command, use the mouse to create a box around the affected region by clicking on two corner points. The program identifies the enclosed nodes and relaxes their positions. After using the command one or more times, proceed to the *Move node* command to complete the repair.

Move node

With this command you can move an individual node of the mesh to correct a boundary. Move the cross-hair pattern near the target node and click the left button. **Mesh** highlights the node. Move the mouse to the desired new position and click the left button. Click the right button to abort the operation.

The following commands correct Type 4 errors.

Node RegNo (point)

You can spot nodes with incorrect region numbers using the *Node* plot style. This command changes the identity of an individual node. Use the mouse to identify the target node. A dialog appears with the present region number. Type in the new number and click *OK.*

Node RegNo (box)

This command is similar to the *Node RegNo (point)* command except that you outline a region of the mesh by clicking on the two corners of a box with the mouse. The change effects all nodes within the box.

Element RegNo (point)

Use this command to change the region number of a single element. Move the mouse pointer inside the element and click the left button.

Element RegNo (box)

This command is similar to the *Element RegNo (point)* command except that you define a region of the mesh by clicking on the two corners of a box with the mouse. The change effects all elements with center-of-mass inside the box.

4.9. Plot menu - hardcopy

You can export plots to hardcopy devices and files with the commands of the *Hardcopy* menu.

Default printer

This command sends a copy of the current plot to the *default* Windows printer. If you have several printers, use the *Settings/Printers* option on the *Start Menu* to make changes in the default before running **Mesh**. The command supports all installed Windows print drivers. The plot will be in color if you have a color printer with the appropriate driver.

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

This command initiates a graphics file of the current plot in either Encapsulated PostScript, Windows Bitmap or Portable Network Graphics formats. The program prompts for a file prefix. The graphics files are created in the current directory with names of the form `FPrefix.EPS`, `FPrefix.BMP`.or `FPrefix.PNG`.

Copy to clipboard

Copies the current plot to the Windows clipboard (in Windows Metafile format) where you can paste it into other applications.

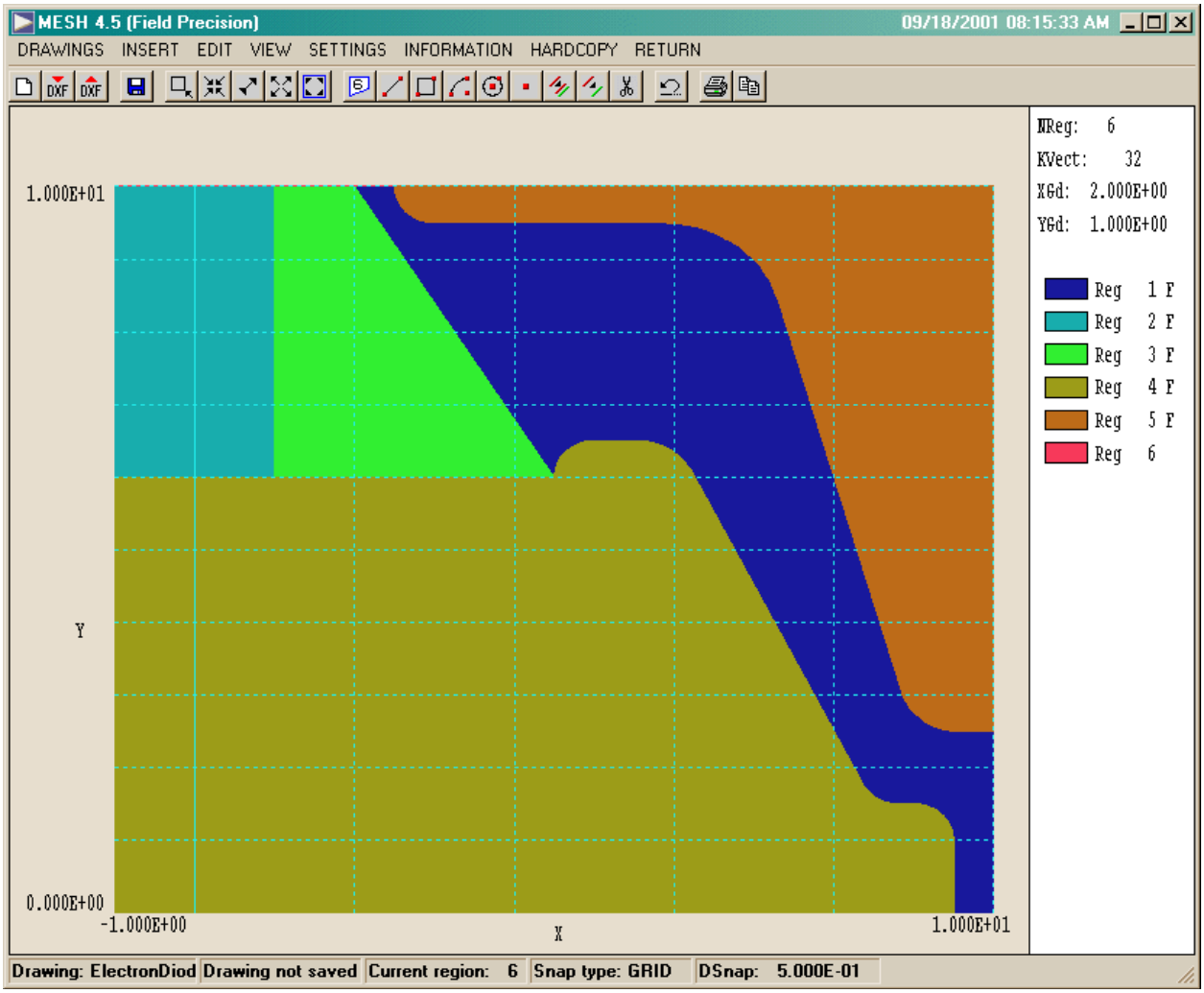

Figure 5.1. Drawing editor screen shot, display of filled regions

Chapter 5. Drawing editor

5.1. Functions of the drawing editor

Version 5.0 of **Mesh** contains a useful drawing editor with many of the functions of advanced CAD programs. The editor enables graphical preparation of boundary input specifications which can be converted directly to a **Mesh** input script. The drawing editor offers the convenience of graphical mesh specification while preserving the portability and transparency of scripts. The drawing editor gives you several capabilities:

■ You can create drawings from scratch and then automatically generate **Mesh** script files from the vector information.

■ You can export DXF drawing files that can be read by most CAD programs or sent to colleagues by E mail.

■ You can import existing drawings from TurboCAD, AutoCAD and other CAD programs into a special layout region. You can then quickly trace and organize entities for incorporation into a **Mesh** script.

■ You can prepare drawings with CAD programs following a few simple rules so that they can be directly translated to a **Mesh** script.

The following section describes a walkthrough example illustrating many of the features of the Drawing Editor. Sections 5.3 through 5.5 cover commands and capabilities of the Drawing Editor menu. Section 5.3 command discusses commands of the *Drawings* menu. Here you can start new drawings, import information from existing DXF files and create **Mesh** scripts (MIN). The commands of the *Insert* menu (Section 5.4) add entities (line, arc, rectangle,...) to new or existing drawings. Entities are organized by numbered layers. The layer number of an entity is used as the region number when it is written to a **Mesh** script. Therefore the terms *region* and *layer* are synonymous. The *Insert* menu also contains useful commands to trace or to move vectors from existing drawings to an ordered set of region boundaries. The *Edit* menu (Section 5.5) contains a comprehensive set of commands to modify drawing vectors (delete, copy, move, fillet, ...). You can also reorder regions so that they are written in the proper order to **Mesh** script files. For example, you can change the region number of a dielectric region so that it appears earlier in the script than a neighboring electrode region. Section 5.6 describes commands of the *Settings* and *Information* menus. Commands in the other menus (*View* and *Hardcopy*) are similar to those described in Sects. 4.5 and 4.9. Finally, Section 5.7 covers methods to prepare drawings in other CAD programs and to import them into **Mesh**.

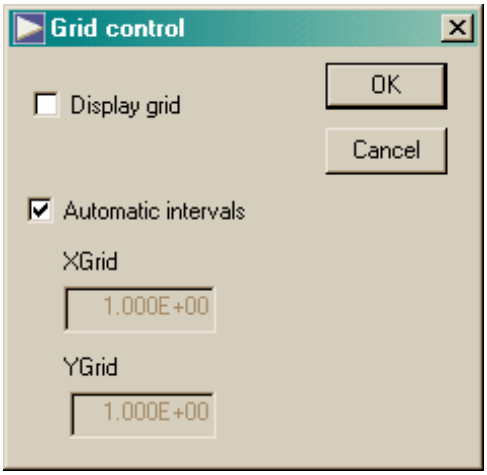

Figure 5.2. Grid control dialog

5.2. Walkthrough example

This section gives a step-by-step description of the preparation of a mesh to simulate a high-power electron-beam injector. Figure 5.1 shows the geometry, a cylindrical system with azimuthal symmetry about the *z*-axis (bottom). The ordinate is the radial distance from the axis. The area shown as *Region* 1 is vacuum and *Region* 2 is a purified-water dielectric at the end of a high-power coaxial transmission line. *Region* 3 is a vacuum insulator, *Region* 4 is the high-voltage center conductor, and *Regions* 5 and 6 correspond to the grounded vacuum chamber wall.

Run **Mesh** and go to the *Drawing* menu. Click on the *New drawing* tool or the *New drawing* command in the *Drawings* menu to bring up a dialog. Type `ElectronDiode` in the title box and set the following limits: z_{min} = -1.0, z_{max} = 10.0, r_{min} = 0.0 and r_{max} = 10.0. After you click *OK*, the program displays the drawing area with default grid intervals. The information area and status bar show that no vectors are currently defined and that the program is ready to insert vectors into *Region* 1. Furthermore, the mouse snap mode is set to *Grid* with a snap distance of 0.50. The latter figure means that whenever you pick a coordinate by pressing the left mouse button the program chooses the closest snap point, such as (-0.5, 1.5), (0.0, 5.0), ... We next set display grid intervals that will be convenient for the drawing. Click on *Grid control* in the *Settings* menu to display the dialog shown in Fig. 5.2. Uncheck the *Automatic intervals* box and change the *z* and *r* grid intervals to 0.5. After you click *OK*, the program displays a new grid with dashed blue lines (Fig. 5.3). Note that the line at *z* = 0.0 is solid.

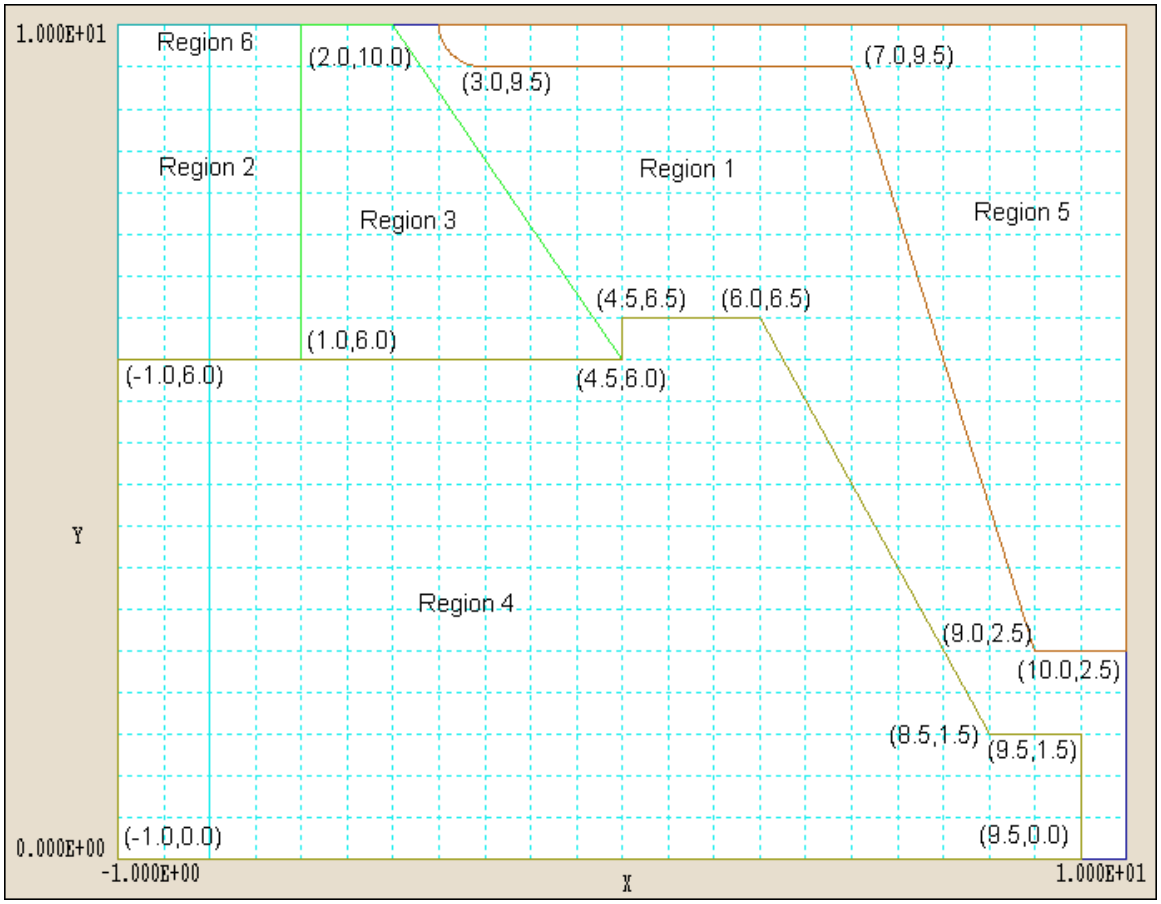

Figure 5.3. Rough drawing for the example *ElectronDiode* showing coordinates and region assignments

Region 1 (vacuum) will cover the full solution area. Later, we will over-write portions of the region to represent electrodes and dielectrics. Pick the *Rectangle* command in the *Insert* menu and move the mouse pointer into the plot area. Once inside the plot, the mouse becomes active and the cursor changes to a cross-hair pattern. The status bar at the bottom of the display shifts to coordinate-input mode. It shows brief instructions, the present location of the cursor and the snap status. Move the cursor close to the bottom-left corner and click the left button. Then move the cursor to the top-right corner and click the left button a second time. The information area now shows that the drawing contains four vectors. The set of vectors (plotted in dark blue) outlines the drawing region. The drawing may not be correct if the pointer was not close enough to snap to the corners. In this case, pick the *Undo last operation* command in the *Edit* menu or click the *Undo* tool.

To ensure proper assignment of boundary conditions in **EStat**, we want to define dielectric regions first and then to enter constant potential regions (electrodes). *Region* 2 represents purified water at the end of a low-impedance coaxial transmission line. In the *Insert* menu click on *Set current region* and enter the number 2. Use the *Rectangle* command to define the region illustrated in Fig. 5.3. The corner points are at *z-r* coordinates of (-1.0,6.0) and (1.0,10.0). *Region* 3 represents a vacuum insulator. Set the current region to 3 with the *Change current region* command. Click the *Line* tool or the *Line* command in the *Insert* menu. The program enters *repeat mode* where you can enter multiple lines. Note that the menus and tools are deactivated. Click the left mouse button to set a line start point at (1.0,6.0), then move to (4.5,6.0) and click the left button again. The program inserts the vector, plots it in green and awaits input for the next line. Click the left mouse button to set the start point at (4.5,6.0) and the end point at (2.0,10.0). To draw the top line, set the start point at (2.0,10.0). Before setting the end point, press the *F2* key. This brings up a dialog where you can change the snap mode for the next coordinate entry. Section 5.6 discusses snap modes in detail. Grid snap is sufficient for the present drawing, so click *CANCEL* to exit the dialog with no changes. Continue by setting the end point of the line at (1.0,10.0). Finally, draw a line from (1.0,10.0) to (1.0,6.0) to complete the outline of the insulator. Press *ESC* or click the right mouse button to exit repeat mode. (Note that the right mouse button is active only when the pointer is in the plot area.)

To complete the rough drawing, we need to enter three more regions: Region 4 (inner high-voltage electrode), Region 5 (grounded shaped vacuum chamber in the diode region) and Region 6 (grounded outer wall of the transmission line). Use the *Change current region* and *Line* commands to outline Region 4 (Fig. 5.3) with the following coordinates: (-1.0,0.0) ⇨ (9.5,0.0) ⇨ (9.5,1.5) ⇨ (8.5,1.5) ⇨ (6.0,6.5) ⇨ (4.5,6.5) ⇨ (4.5,6.0) ⇨ (-1.0,6.0) ⇨ (-1.0,0.0). Be sure to enter all segments. It does not matter if you enter the segments in order. **Mesh** automatically sorts the vectors and arranges closed regions as they are entered.

The outline of Region 5 involves five lines and one arc. Change the current region to 5 and enter lines between the following coordinates: (2.5,10.0) ⇨ (10.0,10.0) ⇨ (10.0,2.5) ⇨ (9.0,2.5) ⇨ (7.0,9.5) ⇨ (3.0,9.5). Exit repeat mode and click on the *Arc* tool or the *Arc/Start-end-center* command in the *Insert* menu. Set the start point at (3.0,9.5), the end point at (2.5,10.0) and the center point at (3.0,10.0).

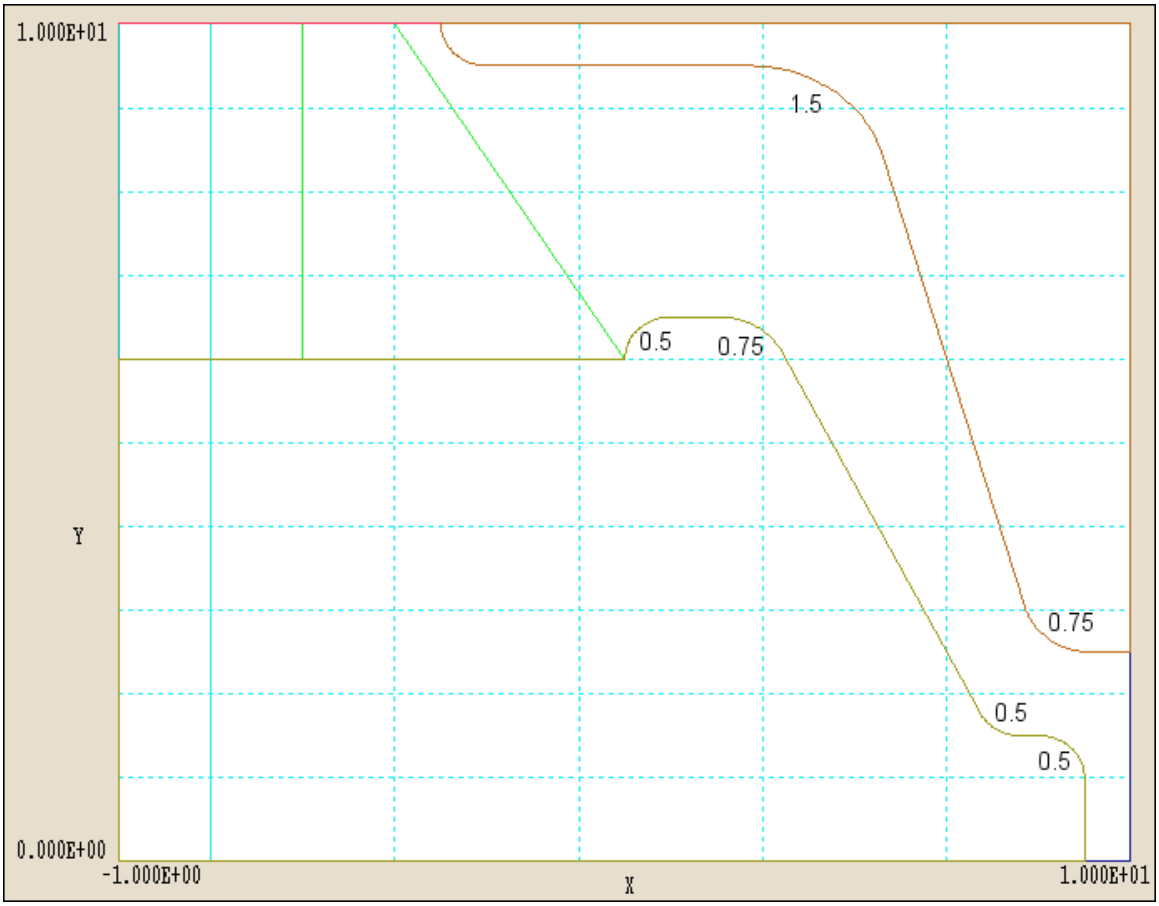

Figure 5.4. Finished drawing with fillet radii, *ElectronDiode* example

At this point the drawing should resemble Fig. 5.3. (Coordinates and labels have been added for clarity.) Finally, the outer wall of the vacuum chamber (Region 6) is a single line from (-1.0,10.0) to (2.5,10.0).

Save the rough drawing by choosing the *Export DXF file* tool or the *Export DXF file* command in the *File* menu. Type in the file prefix `ElectronDiodeRough` and click *OK*. The program makes the DXF file `ElectronDiodeRough.DXF` in the current directory. You can reload this file or open it in another CAD program.

Now we will use the *Fillet* command in the *Edit* menu to make the finished drawing. Click on the *Fillet/chamfer width* command in the *Edit* menu and change the fillet radius to 0.5. Choose the *Fillet* command and pick the two vectors on the right-hand side of the high-voltage electrode (*Region* 4). Figure 5.4 shows the locations and

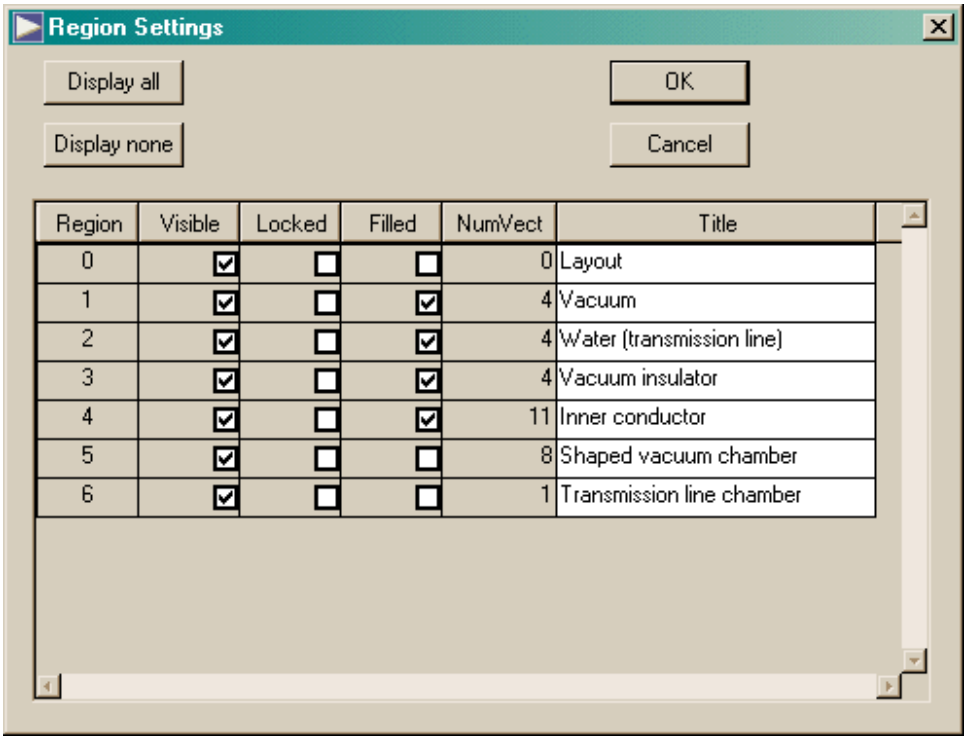

Figure 5.5. Region properties dialog

radii of fillets. If all is well, your drawing should look like the drawing of Fig. 5.4 after you complete the fillet operations. If not, use the *Undo* command or reload the file `ElectroDiodeRough.DXF`.

To complete the drawing we will set parameters for the regions. Choose the *Region properties* command in the *Settings* menu to call up the dialog shown in Fig. 5.5. Click check boxes to set *Regions* 1,2,3,4 and 5 to the *Filled* condition and type in descriptive names for the regions. To type in a name, click the mouse inside the desired title box. The *TAB* key cannot be used to move between title boxes. Click *OK* when you are finished. Note that the fill status of regions has been updated in the information area. To check if the filled regions have been defined correctly, choose the *Display region fill* command in the *Settings* menu. You should see the display of Fig. 5.1. The fill display give invalid results if a region with an unconnected boundary is set *Filled*. To turn off the fill display, click the *Display region fill* command again. Use the *Export DXF file* command to make a copy of the final drawing. The fill status of the regions and the titles are recorded in the DXF file in a form that is recognized by **Mesh** but not by other CAD programs. The settings are restored if you reload the

drawing. The final operation within the drawing editor is to make a **Mesh** script file. Choose the *Create MIN file* tool or the command in the *File* menu. Accept the default prefix and click *OK*. The program makes the file `ElectronDiode.MIN` and opens it in the text editor. Note that the region titles have been added as comment lines in the script. At this point you could make changes in the *XMesh* and *YMesh* statements to fine-tune the variable resolution. For this exercise, simply exit the text editor and click *Return* in the drawing editor menu. You can now load and process the file `ElectronDiode.MIN` with the commands discussed in the previous section.

5.3. Drawings menu

The commands of the *Drawings* menu control file input and output.

New drawing

Close the current drawing and start a new one. You will be prompted for a drawing title (1-20 characters) that is used as the prefix for DXF and MIN output files. You must also enter drawing limits. These quantities are used to set the size of the solution volume (x_{min}, y_{min}, x_{max}, y_{max}) in the `MIN` file. The program picks a default grid, displays the drawing space, and waits for input into Region 1.

Import DXF

Read an existing DXF file created by a CAD program. Changing directories in the choose-file dialog changes the working directory of the program. Valid entities in layers named *1, 2, 3, ...* are assigned to the corresponding regions. All other entities are assigned to the Layout region (0). If the file is a drawing previously created by **Mesh**, the region fill status and titles will be restored.

Export DXF

Create a bare-bones DXF file of the current drawing. This file can be read by other CAD programs or graphics programs that support the DXF format. You can also reload the file into **Mesh**. This command is useful to make backup drawings or to create a library of standard geometries.

Create MIN file

Translate the vectors in the current drawing and create a **Mesh** input script file (MIN). The fill status and region titles are transferred to the script. After translation, **Mesh** runs a text editor

and loads the output file. At this point you can make changes in the foundation mesh resolution (*XMesh* and *YMesh* commands) or accept the default values.

5.4. Insert menu

The commands of the *Insert* menu are used to place point, line and arc vectors in new or existing drawings. Some of the commands cause the program to enter the *repeat mode*. In this case, the command repeats automatically until you press the *ESC* key or the right mouse button. As an example, the repeat 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 on the appropriate tool or menu entry. Most commands in this section require mouse input of coordinates. A special status bar is displayed when the mouse is active. The bar shows instructions, local coordinates and the snap mode. You can change the methods for coordinate entry whenever the mouse is active. To enter coordinates from the keyboard, press the *F1* key. To change the snap mode, press the *F2* key. Section 5.6 discusses snap modes.

Set current region

All insert operations place vectors in the current region. Enter a number to start a new region or to add vectors to an existing region.

The next five commands are used to create vectors in the drawing.

Line

Enter two points to define a line. (Supports repeat mode.)

Rectangle

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

Arc

There are two ways to define an arc: start-end-center and start-end-radius. In the first option enter the start-point, end-point and center-point of the arc. 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°. Divide long arcs into two parts. (Supports repeat mode.)

Circle

There are two ways to define a circle: center-radius and center-point. In the first option, specify the center point with the mouse and then type the value of the radius in the dialog. In the second option, specify the center point and a point on the radius with the mouse. The program creates four 90° arc vectors.

Point

Enter a point. (Supports repeat mode.)

The following two commands are useful for tracing region boundaries using entities in the layout region for reference.

Copy to current region

Find the vector closest to the mouse cursor and copy it into the current region. Note that mouse snap modes are automatically turned off on during any search operation. (Supports repeat mode.)

Move to current region

Find the vector closest to the mouse cursor and move it into the current region. This operation is also useful if you want to move one or more vectors into a different region. (Supports repeat mode.)

5.5. Edit menu

The commands of the *Edit* menu are used to modify vectors and regions in the drawing.

Undo last operation

Mesh automatically makes a copy of the current drawing vectors in the file `TEMP.DXF` before executing most of the operations in the *Insert* and *Edit* menus. The *Undo* command reads the file and restores the drawing to its state before the operation. All changes made during repeat mode operations will be cancelled. The `TEMP.DXF` file also serves as an automatic backup. By reloading the file, you may be able to restore a drawing lost in a system crash.

Delete vector

Delete a single vector. To select the vector, move the mouse near it and click the left button. The selected vector is highlighted. When two regions share a boundary, mesh always selects the vector with the highest region number. If this is not the vector you want, use the *Region properties* command in the *Settings* menu to lock or to turn off the display of the overlapping region. In a selection process, **Mesh** will not include vectors in invisible or locked regions. (Supports repeat mode.)

Delete region

To identify the region move the mouse cursor close to one of the region vectors and click the left button. The region is immediately deleted and the other regions are reordered. To reverse the action, use the *Undo last operation* command. You can prevent the accidental deletion of a region by locking it or turning off its visibility.

Copy region

This command copies all vectors of a region to a new position. The number of regions is increased by one and the new vectors are assigned to the new region. The first step is to identify the region to be copied by moving the mouse cursor near one of the region vectors and clicking the left button. Next specify a reference point with the mouse using the current snap mode. Finally, specify a new point to define a displacement. You can use this command to create an array of objects. You can change the number of the new region with the *Increase region number* and *Decrease region number* commands.

Move region

Change the position of all vectors in a region. Identify the region with the mouse and then enter two points using the current snap mode: a reference point and a second point to define a displacement.

Rotate region

Rotate all vectors in a region Identify the region with the mouse and then enter the center point for the rotation using the current snap mode. Type the rotation angle (degrees) in the dialog. To make a circular array, use the *Copy region* command with zero displacement to make several copies of a region and then rotate the copies starting from the highest region number.

Split vector

Identify a line or arc vector with the mouse pointer and split it into two parts. Presently the only option is to split the vector at its midpoint.

Fillet

Smooth the edge between two connected line vectors by joining them with an arc of a given radius. Use the *Fillet/chamfer width* command to set the radius. Pick the two vectors with the mouse. The resulting arc has the specified radius and is tangent to the two lines. The program reports on error if it is impossible to satisfy this condition.

Chamfer

Smooth the edge between two connected lines by joining them with a bevel of specified length. Use the *Fillet/chamfer width* command to set the bevel length. Pick the two vectors with the mouse.

Filler/chamfer width

Set the fillet radius or chamfer width by typing values in the dialog. Be sure to include decimal points.

Increase region number, Decrease region number

You can change the region order by moving a region up or down in the list. Pick a region by moving the mouse cursor close to one of its vectors and clicking the left button.The status bar shows the region number. Press *RETURN* one or more times to change the number. **Mesh** shifts the region and automatically reorders the region numbers of all other vectors in the drawing. Press *ESC* or the right mouse button to exit.

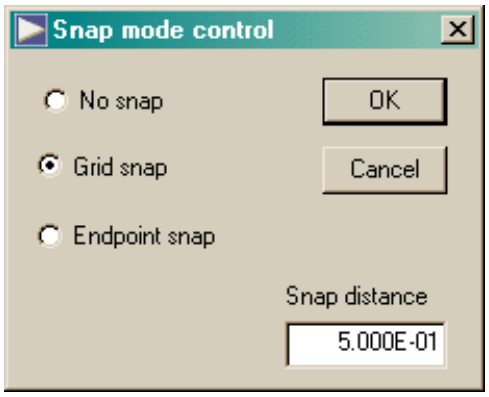

Figure 5.6. Snap control dialog

5.6. Settings and information menus

Region properties

Figure 5.5 illustrates the region properties dialog. The dialog shows status information and the number the vectors in the region. If you turn off the *Visible* attribute, the region will not be displayed and will not be chosen for any of the operations in the *Edit* menu. A *Locked* region is displayed but will not be chosen for *Edit* operations. When a region is *Filled,* **Mesh** writes the keyword *FILL* in the corresponding *REGION* line when it creates an MIN file. The program automatically sets the fill status when it reads vectors from a DXF file. During this process regions with closed boundaries are marked as *Filled*. Turn off the fill status if a region with a contiguous boundary should be *Open* (*i.e.*, a line region to set a Dirichlet condition around the solution volume). You can also type in descriptive titles for the regions. The titles are preserved in exported DXF files and are transferred as comment lines to `MIN` files. The region properties dialog is useful to check the order of regions after *Increase region number* and *Decrease region number* commands.

Display region fill

This command sets and resets the filled region display mode. The display is useful to check whether the vectors of a region set as *Filled* actually define a closed boundary. Valid filled regions give a plot like the one of Fig. 5.1. The fill logic is incorrect if the vectors are unconnected or do not surround the region.

Grid control

This command calls up the grid control dialog illustrated in Fig. 5.2. In the automatic grid mode, the program picks convenient grid intervals which may change under the *Zoom* and *Expand view* operations.

Snap control

This command calls up the snap control dialog shown in Fig. 5.6. Under the *No snap* option the program picks coordinates at the present mouse position (limited by the resolution of the screen). Under the *Grid snap* option the returned coordinate values are an integral multiple of the distance *DSnap*. Under the *Endpoint snap* option the program searches for the vector endpoint closest to the mouse position. You can call up the snap control dialog during any mouse operation by pressing the *F2* key. To snap to the displayed grid, set *DSnap = XGrid = YGrid*.

Vector information

In response to this command the program will give geometric and region information on a vector that you pick with the mouse.

Region information

List information on a region that you pick with the mouse.

The commands in the two remaining menus, *View* and *Hardcopy*, are identical to those in the *Plot-repair* menu (Sects. 4.5 and 4.9).

5.7. Using other CAD programs

Although the Drawing Editor has useful features, it lacks many capabilities of a complete CAD package. A dedicated drawing program may be useful for complex fitting and trimming operations. Ab excellent freeware package (QCAD) for two-dimensional drawings is supplied with **Mesh**.

To illustrate one use of **Mesh** with an external CAD program, suppose you want to prepare a **TriComp** simulation based on boundaries in an existing drawing. In your CAD program turn off any layers with information that is not relevant. You may also want to save a cropped version of the drawing that eliminates parts of the system that will not be included in the simulation. Save the modified drawing in DXF format and use the *Import DXF file* command to load it into **Mesh**.

The Drawing Editor recognizes entities of the types *Point*, *Line*, *Arc*, *Circle* and *Polyline*. It ignores all other entities such as text. The program splits circles into four arc vectors and splits polylines into individual line vectors. Note that **Mesh** only recognizes the line portions of polylines and ignores arcs, splines and other complex shapes. Entities are placed in the Layout layer (Region 0) with the following exception. Entities in layers with the special names *1*, *2*, *3*, ... are loaded into Region 1, Region 2, Region 3,.. (see below). Entities in the Layout layer are plotted in gray. You can use the *Copy to current region* command to reproduce drawing vectors in Regions 1, 2, 3, ... Alternatively, you can use the *Move to current region* command if you do not want to preserve the layout vectors. Vectors in the Layout region are ignored when creating a **Mesh** script file (`MIN`). Only vectors in Regions numbered 1 ... 127 are included.. Note that the layout vectors are saved when you export a DXF file.

As a second example, suppose you want to perform the full drawing process in your CAD program and then use the Drawing Editor only for translation to the **Mesh** script format. In this case, you must create layers named *1*, *2*, *3*, *4*, 5 ... in your drawing. The maximum numbered layer name is *127*. To help in this process, we have included a file `MESHTEMPLATE.DXF` with predefined layers and drawing colors. You can load this file and then create a template for your CAD program. When a DXF file from the drawing is loaded, **Mesh** assigns all valid entities in *Layer 1* of the drawing to *Region 1*, entities in *Layer 2* to *Region 2*, and so forth. Again, unrecognized entity types are ignored and all entities in other layers are assigned to the layout region (Region 0).

Chapter 6. Advanced foundation meshes

6.1. Motivations for variable resolution

Meshes with uniform element size (Sect. 3) are suitable for many applications. On the other hand, variations of element size can often improve solution accuracy and reduce computational time. Common applications of variable resolution include the following:

- Accurate determination of fields near small structures in a large solution space.

- Simulation of an infinite-space boundary condition by surrounding a volume with an extended region of coarse elements.

In contrast to many finite-element programs the **TriComp** programs use *structured* meshes. This term means that the elements have an ordered topology, making it possible to identify neighboring elements and nodes through index operations. Structure gives the advantage of high speed in programs like **Trak** that involve extensive search operations. On the negative side, the logical connections impose some limits on variable resolution.

6.2. Element variations along the axes: XMesh and YMesh

You can introduce variable element size along one or both axes by adding multiple data lines to the *XMesh* and *YMesh* statements that appear in the *Global* section. Consider the following example:

```
XMesh
  0.000  1.000 0.100
  1.000  2.550 0.200
  2.550  4.000 0.300
End
```

The above statements convey the information that x_{min} = 0.00 and x_{max} = 4.00. Furthermore, triangle size along *x* is approximately 0.10 between 0.00 and 1.00, increases to 0.20 over the interval 1.00 to 2.55, and equals 0.30 from 2.55 to 4.00. Note that the intervals along the *x*-axis in each data line must constitute a continuous range and be in order from x_{min} to x_{max}. The *YMesh* data lines have the same format.

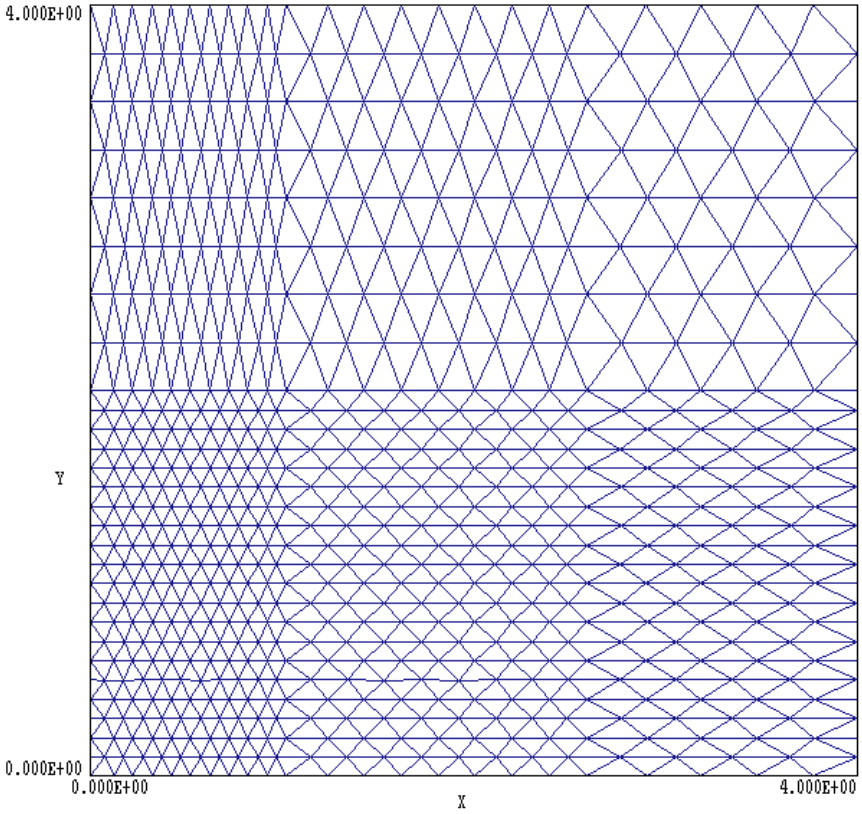

Figure 6.1. Variable resolution example

```
YMesh
  0.000  2.000 0.100
  2.000  4.000 0.250
End
```

Figure 6.1 shows the foundation mesh resulting from the two commands sets listing above.

6.3. Relaxing element sizes: PreSmooth

In the default setting, the *XMesh* and *YMesh* commands produce discontinuous changes in triangle size along each axis as in Fig. 6.1. Usually continuous variations in size lead to more reliable fitting and better element shapes. The *PreSmooth* command relaxes the variations in element size in the foundation mesh. The command appears in the *Global* section and has the form:

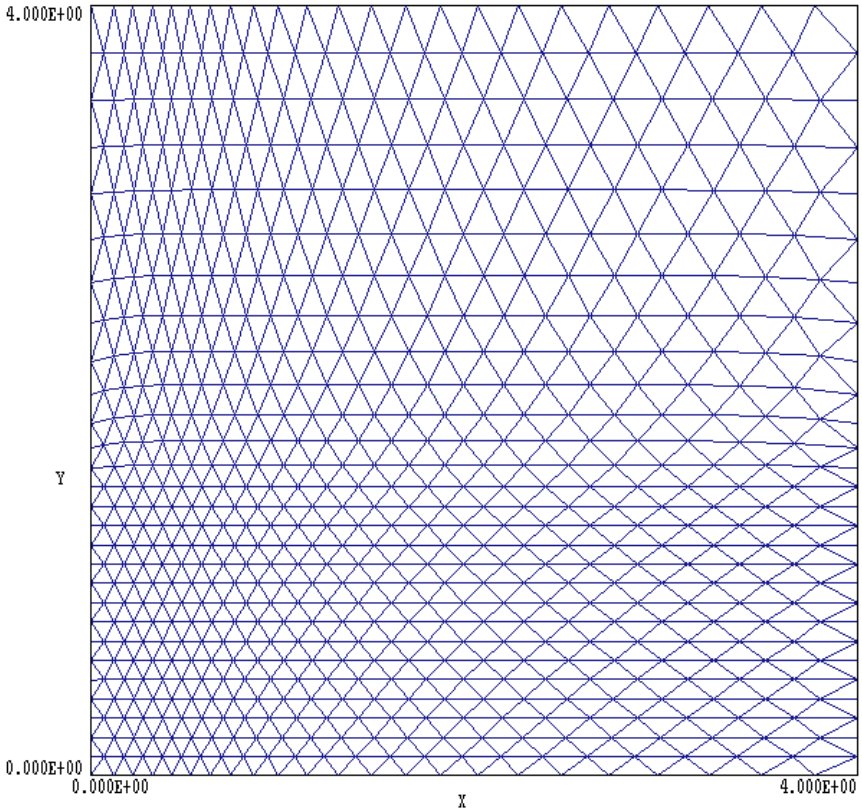

Figure 6.2. Foundation mesh with variable resolution and pre-smoothing

PreSmooth 6

The integer parameter is the number of smoothing cycles (default: 0). Higher values give more smoothing. Figure 6.2 shows the foundation mesh of Fig. 6.1 with 4 cycles of pre-smoothing.

6.4. Setting initial element shapes - TriType

By default, **Mesh** initially fills the foundation mesh with isosceles triangles. The triangles are almost equilateral when the element sizes are equal in the *x* and *y* directions (Fig. 6.3*a*). Equilateral triangles give the best performance in boundary fitting operations on slanted or curved surfaces. There are other element shapes that may be useful in special circumstances.

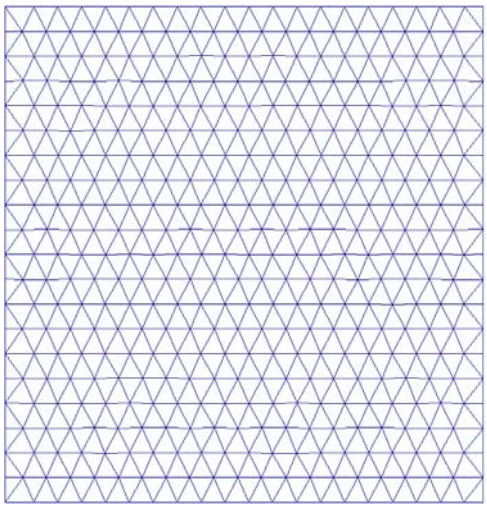

Figure 6.3*a*

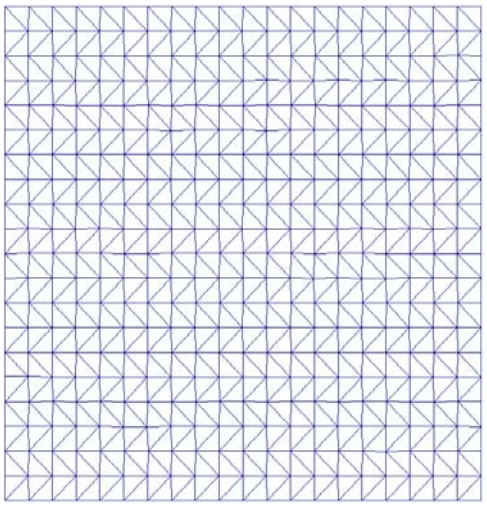

Figure 6.3*b*

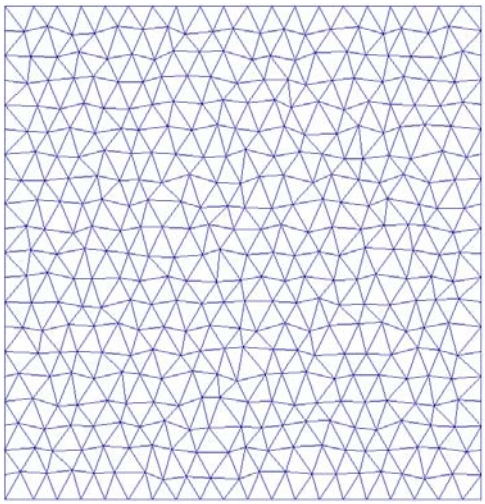

Figure 6.3*c*. *GlassAdjust* = 0.25

The *TriType* command appears in the *Global* section and controls the triangle shape.

TriType Iso

Fill the foundation mesh with isosceles triangles, as shown in Fig. 6.3*a* (default).

TriType Right

Fill the foundation mesh with right triangles (Fig. 6.3*b*). This option may give better element shapes in systems where all boundaries are parallel to the *x* and *y* axes. Here, the term *better* means that all elements have about the same shape and that there are few triangles with acute angles. Distorted triangles can reduce the accuracy of the subsequent field solutions. Do not use this option when there are slanted or curved boundaries.

TriType Glass [GlassAdjust]

Creates a foundation mesh of amorphous elements (Fig. 6.3*c*). In this option **Mesh** fills the foundation mesh with isosceles triangles and then adds random displacements to create a glass-like distribution. This option has been included to support the Field Precision **KB2** hydrodynamics code. In a shock problem ordered element boundaries can act as material dislocations, introducing spurious element displacements. The optional parameter, *GlassAdjust*, determines the degree of disorder. Generally the value should be between 0.0 and 0.5. The example of Fig. 6.3 uses *GlassAdjust* = 0.25. The default value is 0.2. Note that this mode gives no advantages for electromagnetic field simulations.

6.5. Mesh smoothing

The solution programs of the **TriComp** series achieve best accuracy when the elements all have about the same shape (equilateral triangles). Acute triangles with very small angles increase roundoff errors in the numerical calculations. After shifting vertices to boundaries, **Mesh** attempts to improve the shapes of elements by relaxing the positions of un-clamped vertices. The process of mesh smoothing is controlled by the *Global* command:

Smooth 10

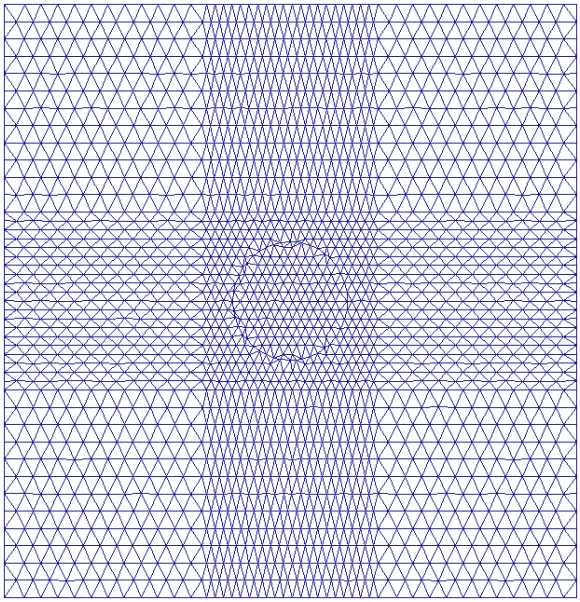

Figure 6.4*a*

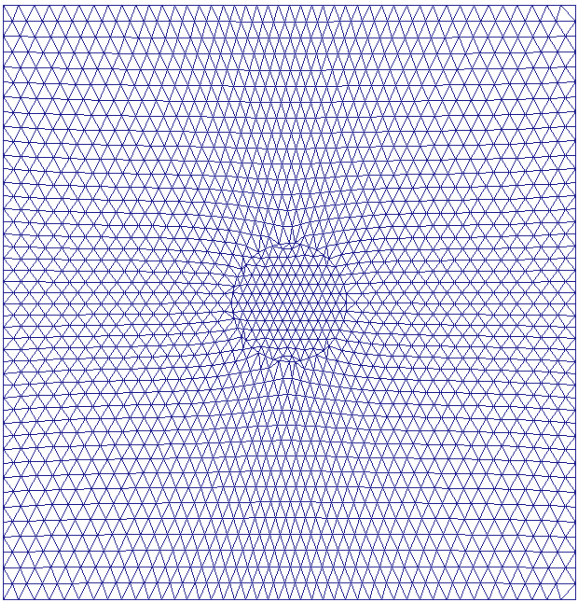

Figure 6.4b

The integer parameter is the number of relaxation cycles. Use *Smooth 0* if you want to preserve the discontinuities in triangle size defined by the *XMesh* and *YMesh* statements. The default value is *Smooth* = 15.

Figure 6.4 shows a mesh with and without smoothing. Note that clamped vertices along the sides of the foundation mesh volume (at x_{min}, x_{max}, y_{min} and y_{max}) are allowed to slide along the sides during the process. If you want to prevent this process, use the *Global* command

FixBound

6.6. Autocorrection and boundary fitting parameters

Autocorrection is a powerful new feature in Version 5.0 of Mesh. The program automatically shifts positions of nodes connected to any inverted elements detected after the fitting process. In most cases, the autocorrection process eliminates the problem of inverted elements. In the rare case where the procedure fails to fix all elements, you can change properties of the foundation mesh or use the repair tools described in Section 4.8. By default, autocorrection is active. Use the following command if you want to deactivate the process and make corrections manually:

AutoCorrect OFF

There are two specialized *Global* commands that may be helpful in the event of an error while fitting boundary vectors. **Mesh** moves vertices during the fitting process. The *Relax* parameter controls whether the six neighbors of the shifted vertex are also shifted in position. If a target vertex is displaced by a vector **d**, then the neighbors are displaced by *Relax* × **d**. This process is helpful so it is usually not necessary to change *Relax* from its default value of 0.2. On the other hand, displacing neighboring vertices may cause problems in tight corners or when tracing the same boundary several times.

You can adjust *Relax* with the command

Relax 0.25

The parameter should be a positive number smaller than 1.00. A value 0.00 turns off the adjustment process.

The *Tolerance* command sets a maximum distance criterion such that two coordinate locations are taken as the same point. In other words, two points are identical if their separation is less than *Tolerance*. An approximate criterion of equality is necessary when dealing with inexact floating point inputs. The command has the form

Tolerance 1.0E-5

where the parameter is the distance criterion.

Usually, you do not have to set the Tolerance. **Mesh** sets a default value equal to 10^{-2} times the minimum element dimension in the foundation mesh.

6.7. Mesh generation problems

We continually seek to improve the reliability of **Mesh**. Nonetheless, it is impossible to create a mesh generator that will accept any user input and always perform flawlessly. There is always the possibility that the triangles of the foundation mesh will be ill-suited to the geometry of one or more regions. With some care in specification of the foundation mesh, all reasonable physical geometries can be modeled. This section reviews ways to avoid problems and methods to diagnose them.

If the boundaries do not look at all like your conception of the system, the most likely cause is a typographical error in the command file. At the next level, **Mesh** may recognize the correct boundaries and process all the vectors but give a *Bad Triangle* message. This means that the program had to displace nodes so far to match a boundary that some elements have been turned inside-out. The autocorrection process may not be able to handle severe distortions. In this case, you should modify the foundation mesh to conform more closely to the system geometry. You can also repair a limited number of inverted elements using the interactive features of **Mesh** discussed in Chapter 4.

There are three main causes for bad elements.

- The local size of triangles is too large to fit the geometry of the specified objects.
- Objects have sharply pointed edges.
- The general shape of the triangles makes it impossible to fit one or more of the specified vectors

The first problem can usually be solved by decreasing the size of triangles, either globally or locally, using the *XMesh*,and *YMesh* commands to vary the resolution. The second problem is a special case

of the first. **Mesh** may get lost if it walks to a sharply pointed edge along one side and then cannot find available nodes to walk back along the other side. Again, the answer is to increase resolution near the trouble spot.

Regarding the third problem, a set of approximately equilateral triangles (base comparable to the height) gives the most reliable fit to curved and slanted surfaces. You must be cautious setting up foundation meshes where the base and height differ widely. For example, suppose the triangles are stretched in y so that they have a height about five times the base. With this geometry, it requires large triangle distortions to match a 45° line, increasing the chance of program errors.

6.8. Mesh errors in the solution program

The third problem level is where the mesh generation was apparently successful but the physical solution for the fields looks incorrect. The most common cause of trouble is incorrect ordering of regions in the **Mesh** command file. The common symptom is strange field line behavior at boundaries (Fig. 2.4*c*).

Another possible cause of a non-physical solution is the incorrect identification of the elements and vertices inside a closed region. **Mesh** counts the number of intersections of a projected line with the vectors that define the boundary of a closed region in order to determine if a point is inside. This procedure may fail if there are a large number of vectors in the problem with endpoints at exactly the same *y* coordinate. The way to diagnose the problem is to use the *Region* and *Vertex* plot features of **Mesh** to spot points or elements that are out of place. The problem usually does not occur for command files prepared from DXF files. Most CAD programs add small floating-point errors to the coordinates that remove the degeneracies. This problem has been largely eliminated in Version 5.0 of **Mesh**.

On rare occasions you may observe local distortions of fields in contour plots and scans with an apparently good mesh. The problem arises when elements in an area have bad shapes (i.e., very small angles). Check the mesh in the problem area by zooming in on the problem region with a mesh plot. Often, you can fix the triangles by making small changes in the local resolution using the *XMesh* and *YMesh* commands. You can also move and relax selected vertices using the interactive features of **Mesh**.

6.9. Trouble-shooting

Here are some tips to diagnose mesh generation problems.

■ Check the file `FPrefix.MLS`. All **TriComp** programs make listing files that document the course of the run and contain valuable debugging information.

■ Temporarily change the input script to narrow down problems in boundary fitting. Move the *EndFile* command, adding regions one-at-a-time to pinpoint problems. You can also check individual vectors in a region by deactivating the others with asterisks at the beginning of lines. (If you deactivate some vectors of a filled region you should temporarily remove the *Fill* option in the *Region* command.)

■ Use *Region* and *Vertex* plot commands of **Mesh** to confirm how the program has assigned region numbers to the components of the mesh.

■ There are several possible causes of a logical path error. The most likely is a syntax error in the `MIN` file leading to an inconsistent vector definition. To fix the problem, use the *Edit current script* command to check the file. The editor automatically places the cursor near the line where the error occurred. A logical path error may occur when a line or arc vector is too small to resolve with the foundation elements. In this case, decide if the small detail is physically significant. If not, remove it. If the object is critical, reduce the element size or employ variable resolution in the *XMesh* and *YMesh* statements. A third possibility is that element shapes in the foundation mesh are inappropriate for the vector. An error may occur if you try to fit an arc in an area with very tall or short elements. Again, the solution is to change the element sizes in the *XMesh* and *YMesh* statements. Another possible cause of a logical path error is that the mesh has more than 2000 elements on a side and the vector extends the full length. The solution is to split the vector into smaller parts. Finally, a logical path error will occur if two vectors of an open region cross each other or intersect at positions other than the endpoints.

■ As a last resort, you can move nodes or change region numbers using **Mesh** in the interactive mode. The drawback is that the mesh cannot be reconstructed completely from the succinct input file script (`MIN`). For future runs, you must save the complete mesh output file (`MOU`).

Chapter 7. Creating meshes from images

7.1. Introduction

The methods described in the previous vhapters were designed to represent a moderate number of objects with relatively simple shapes and precise dimensions. The approach is well suited to simulations of mechanical systems but may be inefficient for systems with indefinite boundaries. As an example, suppose we wanted to study the effect of small electric fields on regeneration of bones in the human leg. To construct a mesh, we might start from an MRI image of a leg cross-section to identify regions of different electrical conductivities (such as bone and muscle tissue). To employ the standard method, we would outline region boundaries, measure dimensions, and convert the results to a series of line and arc vectors. The process involves a great deal of work because biological systems usually include irregular and complex boundaries. Furthermore the effort of defining exact dimensions would be largely wasted because there is considerable variability between legs.

Mesh 5.0 has new features for automatic generation of conformal meshes from image data. These capabilities may save hours of labor. The program can convert several types of two-dimensional data directly to triangular meshes:

- Medical images (MRI, X-ray, ...).
- Digital photographs of equipment.
- Scans of anatomical charts, blueprints and maps.
- Data files describing continuous variations of quantities like temperature and density in space.

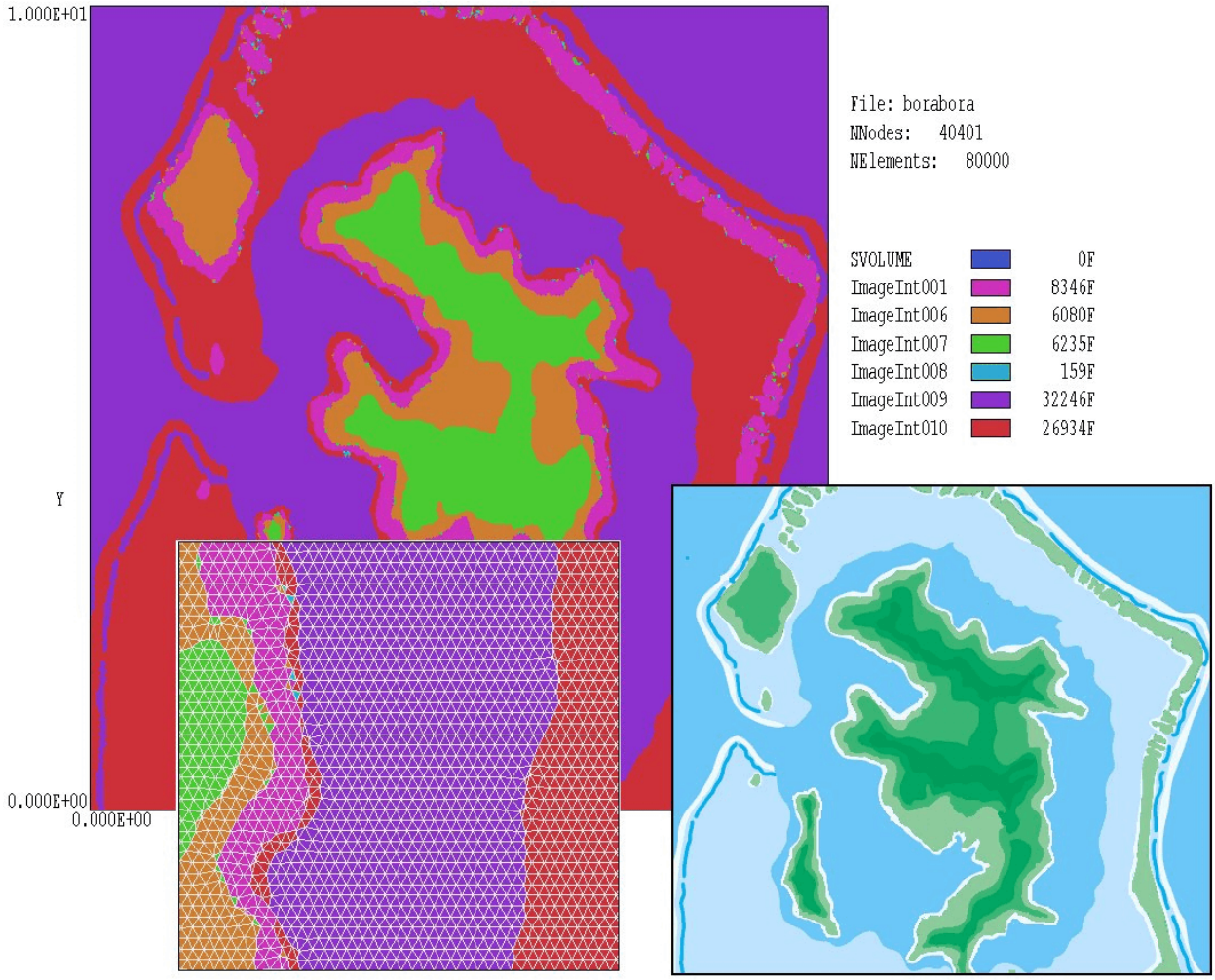

Figure 7.1. Generation of a conformal triangular mesh from a bitmap image. Lower right: input image. Lower left: detail of the mesh showing element boundaries.

Figure 7.1 illustrates an example, a mesh of Bora Bora resolved into regions by elevation. The original bitmap image is shown at bottom right. The inset at bottom left shows a details of the 40,401 node mesh with element boundaries displayed. **Mesh 5.0** required less than 2 seconds to perform the conversion.

The remainder of this section discusses characteristics of the two types of images recognized by **Mesh 5.0**: visual images and data images. Section 7.2 covers modifications of the Mesh 5.0 script while Section 7.3 describes a walkthrough example to illustrate mesh generation from a visual image. Section 7.4 illustrates the use of data images and Section 7.5 summarizes image processing commands and rules.

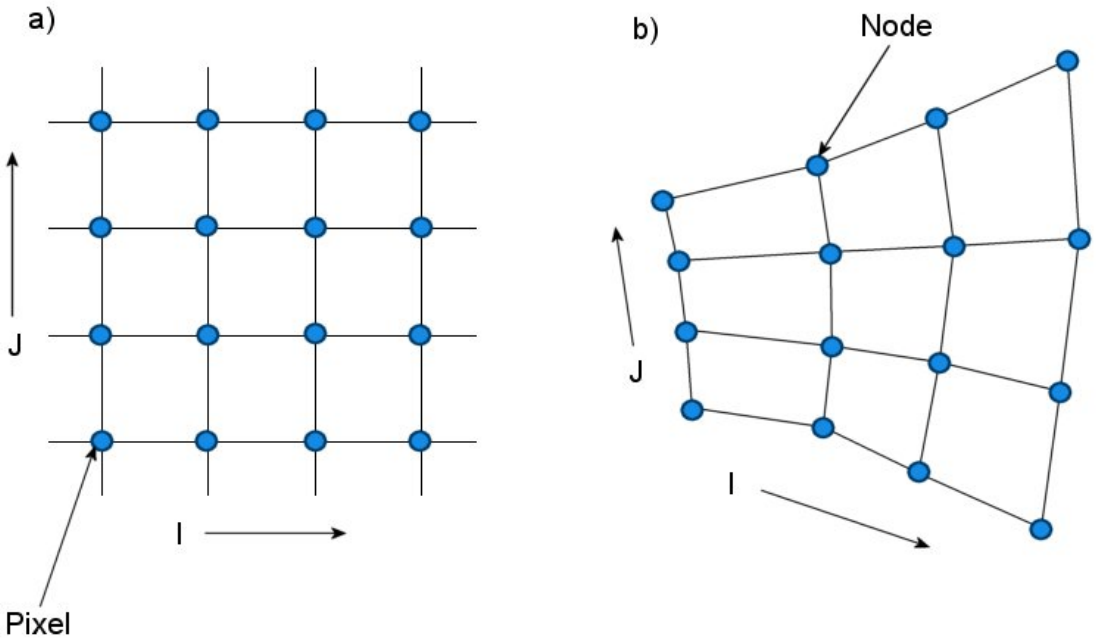

Figure 7.2. Types of image files. *a*) Visual image, pixel values on a square mesh. *b*) Data image, values of a physical quantity defined at the nodes of a generalized quadrilateral mesh.

In **Mesh 5.0** an *image* is a record of a quantity at the nodes of a two-dimensional mesh with quadrilateral logic. The term *quadrilateral logic* means that we can designate node positions with indices (*I*, *J*), where *I* gives the relative position of the node in the horizontal direction and *J* corresponds to the vertical direction. **Mesh 5.0** deals with two types of images.

■ *Visual images* (such as those generated by scanners and digital cameras) are in bitmap format (BMP, PCX and PNG). The mesh for this image type has the regular geometric arrangement shown in Fig. 2*a* where mesh cells are squares. The recorded quantity is the RGB (red-green-blue) value of the pixel at the node.

■ Data images are defined by text files with data lines that contain the values:

X(I,J) Y(I,J) F(I,J)

where *[X(I,J), Y(I,J)]* is the location of the node and *F(I,J)* is the real-number value of any physical quantity at the node. For data

images the mesh cells need not be square or rectangular (Fig. 2*b*). **Mesh 5.0** uses sophisticated interpolation techniques to handle generalized quadrilateral meshes

7.2. Script file organization

A standard **Mesh 5.0** script consists of a *GLOBAL* section followed by one or more *REGION* sections. The *GLOBAL* section defines the properties of the *foundation mesh* (the shape of the solution volume and the approximate sizes of triangular elements that fill it). The *REGION* sections contain vectors that outline objects in the solution space. To process a region, the program shifts nodes so that they lie on boundaries and marks nodes and elements contained within the boundary with the current region number.

Mesh 5.0 scripts may now include one or more *IMAGE* sections with the form:

```
IMAGE
 (Image commands)
END
```

The function of the section is to load a file that contains two-dimensional information and to create several regions based on values in the file. The *IMAGE* sections must follow the *GLOBAL* section and may be included between *REGION* sections. In most cases, a script has the following structure:

```
GLOBAL
 (Global commands)
END

REGION FILL SolutionArea
 (Region commands)
END

IMAGE
 (Image commands)
END

REGION [FILL] Object01
  (Region commands)
END
```

```
REGION [FILL] Object02
  (Region commands)
END

...

ENDFILE
```

At least one *REGION* section must precede the first *IMAGE* section. This special section defines the boundaries of the *solution area*. The solution area may or may not fill the rectangle defined in the *GLOBAL* section. Regions created from the image are clipped to fit inside the solution area. *REGION* sections that follow the *IMAGE* section may overwrite portions of the image. For example, an *IMAGE* section could be used to define regions of different electrical conductivity in a cross-section of the liver. Subsequent *REGION* sections could insert electrodes and insulators with precise dimensions.

The order in which *REGION* and *IMAGE* sections appear **Mesh 5.0** scripts is important. The following rules apply:

- The currently processed *REGION* or *IMAGE* over-writes the region identities of nodes and elements in the shared space.

- In most cases, the function of first *REGION* is to define the solution area. **Mesh 5.0** shifts and clamps nodes on the specified boundary and sets *RegNo* = 1 for all included nodes and elements in response to the *FILL* keyword.

- An *IMAGE* section reassigns elements (and associated nodes) to one or more new regions. An element is reassigned only if it already has a valid region number (*RegNo* > 0). In this way, the solution area region provides clipping information for the image.

- An *IMAGE* section may reassign element identities, but it will not shift clamped nodes on the boundaries of previously-processed regions. In this way, the precise boundaries of the solution area (and any other regions that proceed the image) are preserved.

- *REGION* sections that follow an *IMAGE* may change the region numbers in the shared space and may also shift nodes to conform to the specified region boundary vectors.

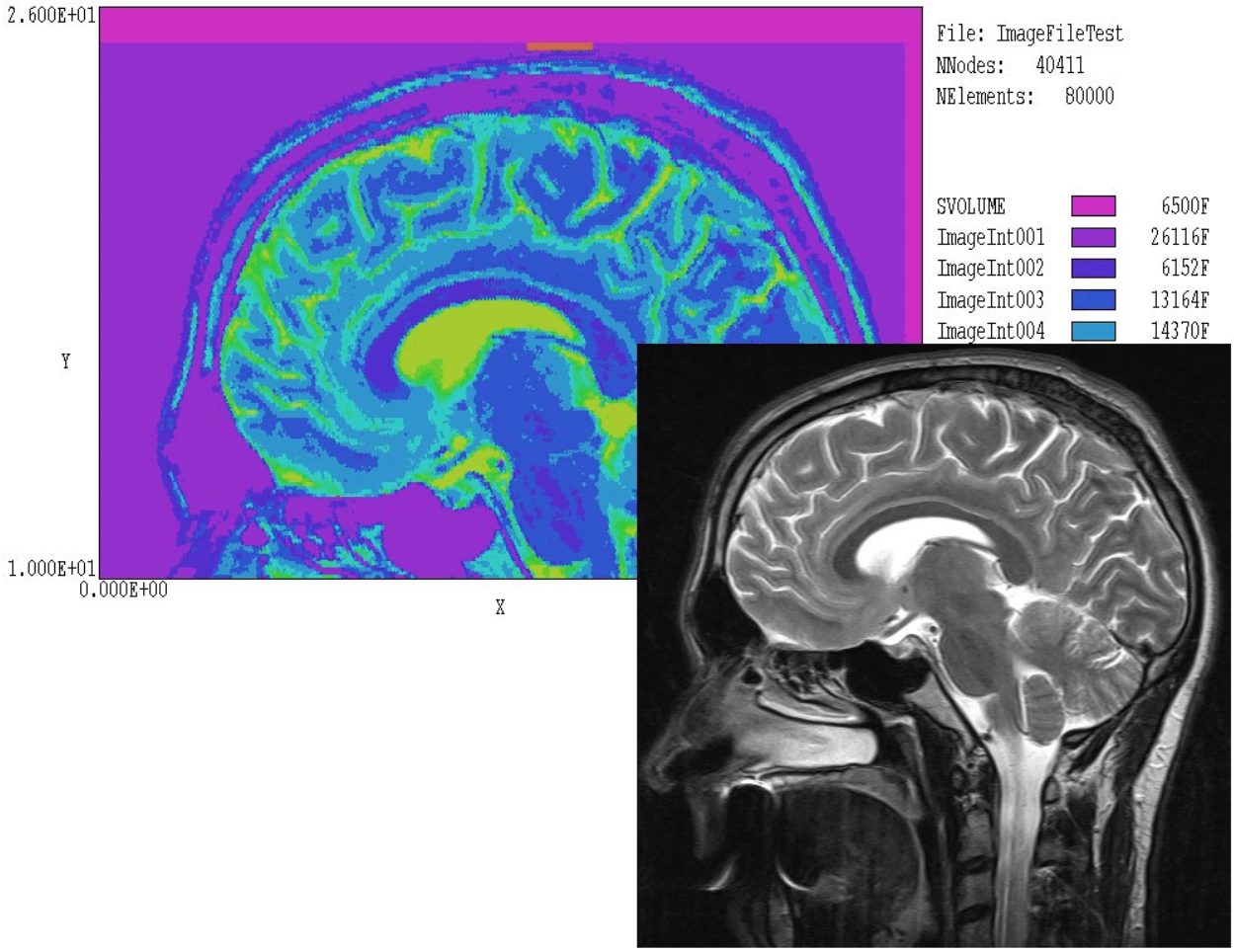

Figure 7.3. Example `BRAIN_MRI`, conversion of a bitmap image to a conformal triangular mesh.

At this point, the rules may seem somewhat obscure. The process is actually simple and easy to implement. The examples in the following sections will clarify image processing operations.

7.3. Visual image example

This section gives a detailed description of the example illustrated in Figure 3, conversion of a gray-scale MRI bitmap image to a conformal mesh. The example illustrates many of the advanced features in **Mesh 5.0**. It also raises issues about what we can hope to accomplish with image conversion. Before beginning, it is important to understand some limitations and constraints.

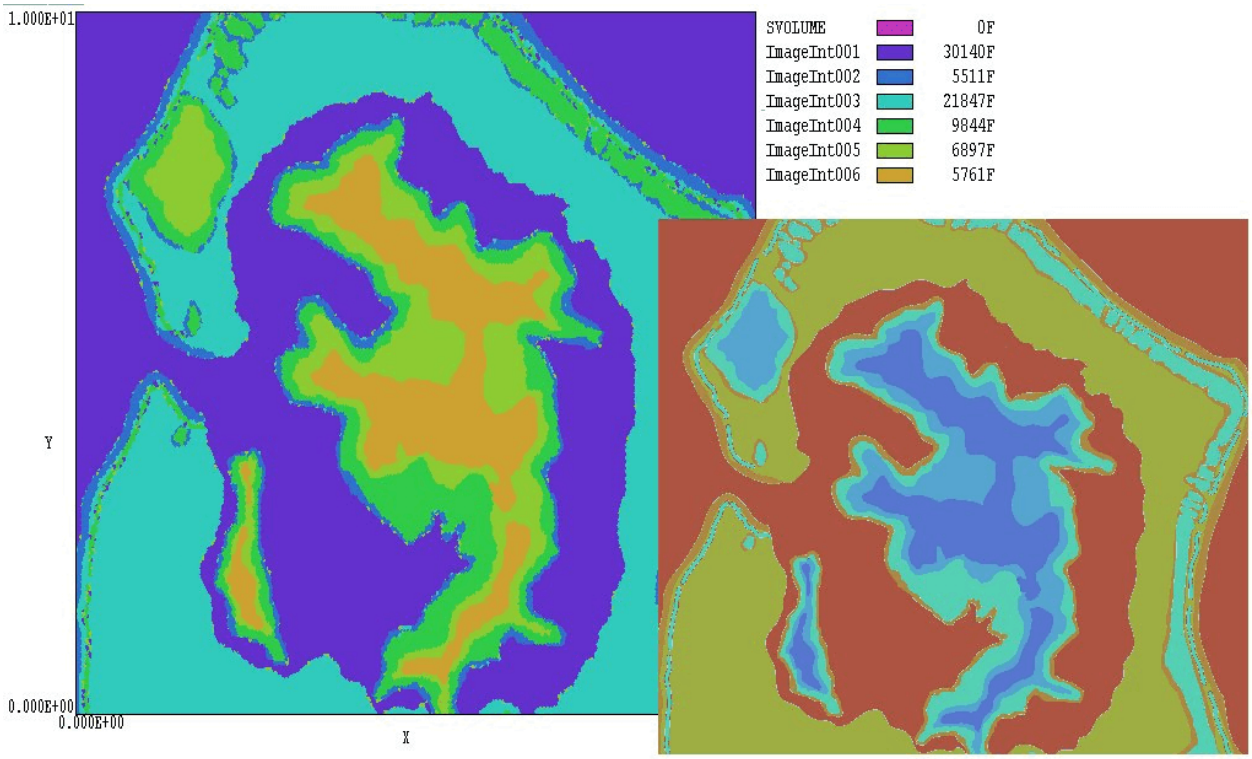

Figure 7.4. Modification of the hues in Fig.7.1 using the color-replacement tool in PaintShop Pro (bottom right) and the resulting mesh with regions ordered by elevation.

■ To perform electric field or thermal solutions, the image must be consistent with the two-dimensional symmetries handled by the codes of the **TriComp** series. The leg cross-section mentioned in Sect. 7.1 is well-suited to the planar symmetry option. On the other hand, the image of Fig. 7.3 is clearly not consistent with planar or cylindrical symmetry. We should consider this example as a demonstration of mesh techniques.

■ In processing visual images, **Mesh 5.0** can divide the solution area into regions based on *LIGHTNESS* or *HUE* values. These values must have some correlation with the physical properties assigned to the regions in subsequent solution programs. Sometimes images may be directly useful. For example, we could photograph the cross-section of a complex iron pole piece with a distinctive color and use the picture to generate the corresponding region in a magnetic field solution. The image of Fig. 7.3 is probably not directly useful. The *LIGHTNESS* values depend on the concentration of nuclear species and there is probably little direct correlation with quantities like thermal or electrical conductivity.

■ The utility of images can be greatly enhanced by pre-processing with programs like PaintShop Pro, PhotoShop or GIMP. For example, we could use the lasso method to select critical brain structures in Fig. 3 and then to change them to a distinctive color that could be easily identified by Mesh 5.0. Figure 7.4 illustrates another useful technique. The hues of the colors in the bitmap image of Fig. 7.1 have degeneracies and are not arranged by order of elevation. We could use a color replacement tool to create a bitmap image with hues arranged in the correct order (bottom right-hand side of Fig. 7.4). The resulting mesh has regions ordered by increasing elevation.

With these precautions in mind, we can proceed to a discussion of the example. The following input files have been included in the **Mesh 5.0** example library:

`BRAIN_MRI.BMP`
: The bitmap image (shown on the lower right-hand side of Fig. 7.3) in Windows/OS2 bitmap format.

`BRAIN_MRI.MIN`
: The **Mesh 5.0** input script listed in Table 7.1.

`BRAIN_MRI_INTERVAL.DAT`
: Gray-scale values for division of the image into regions.

The *GLOBAL* section defines a foundation mesh with dimensions 25.0 cm in *x* and 26.0 cm in *y*. The first *REGION* section outlines the complete area, aligns element nodes along the straight-line boundaries and sets all included elements and nodes to *RegNo* = 1. The *IMAGE* section that follows includes four commands:

`IMAGEFILE BRAIN_MRI.BMP  0.0 0.0 24.5 25.0`
: This command loads the image file. The optional real-number parameters specify that the image should be mapped into a rectangular portion of the solution volume with corners at [0.0, 0.0] and [24.5, 25.0]. The parameters can be used to specify physical dimensions and also to shift or to scale image. If they are omitted, the program adjusts the image to fit the foundation mesh rectangle.

Table 7.1. Contents of the file BRAIN_MRI.MIN

```
GLOBAL
  XMesh
    0.00 25.0  0.10
  End
  YMesh
   10.00 26.0  0.10
  End
END

REGION FILL SVolume
  L  0.0  10.0  25.0  10.0
  L 25.0  10.0  25.0  26.0
  L 25.0  26.0  00.0  26.0
  L  0.0  26.0  00.0  10.0
END

IMAGE
  ImageFile Brain_MRI.bmp 0.00 0.00 24.5 25.0
  IntervalType Lightness
  IntervalFile Brain_MRI_Interval.dat
  ImageSmooth 3
END

REGION FILL Electrode01
  L  24.0  13.0  24.2  13.0
  L  24.2  13.0  24.2  15.0
  L  24.2  15.0  24.0  15.0
  L  24.0  15.0  24.0  13.0
END

REGION FILL Electrode02
  L  13.0  24.8  15.0  24.8
  L  15.0  24.8  15.0  25.0
  L  15.0  25.0  13.0  25.0
  L  13.0  25.0  13.0  24.8
END

ENDFILE
```

`INTERVALTYPE LIGHTNESS`

This command specifies that mesh elements will be assigned to regions according to the *LIGHTNESS* value of the image at the mapped position of the element center-of-mass. **Mesh 5.0** converts RGB pixel values to HLS (hue-lightness-saturation). Hue values range from 0.0° to 360.0° and lightness values from 0.0 to 100.0.

`INTERVALFILE BRAIN_MRI_INTERVAL.DAT`

In response to this command, **Mesh 5.0** reads a file that contains specified intervals in *LIGHTNESS* for the region division. The file defines nine intervals. The first interval has width 20.0 (to consolidate dark areas) and subsequent intervals have width 10.0.

`IMAGESMOOTH 3`

This command specifies that the program should perform three cycles of smoothing after element assignment. The process reduces jagged edges on boundaries between image regions.

The remaining two *REGION* sections place electrodes on the skull. The region designated *ELECTRODE02* is visible as the orange rectangle near the top of the mesh in Fig. 3.

The listing file `BRAIN_MRI.MLS` contains useful information on the image conversion process. For example, **Mesh 5.0** creates the following table if an INTERVALFILE command appears in the script:

```
Intervals from file BRAIN_MRI_INTERVAL.DAT
 Interval   FMin        FMax
============================
     1   0.000E+00   2.000E+01
     2   2.000E+01   3.000E+01
     3   3.000E+01   4.000E+01
     4   4.000E+01   5.000E+01
     5   5.000E+01   6.000E+01
     6   6.000E+01   7.000E+01
     7   7.000E+01   8.000E+01
     8   8.000E+01   9.000E+01
     9   9.000E+01   1.000E+02
```

The program also creates a table of image information if an *IMAGEFILE* command appears in the script:

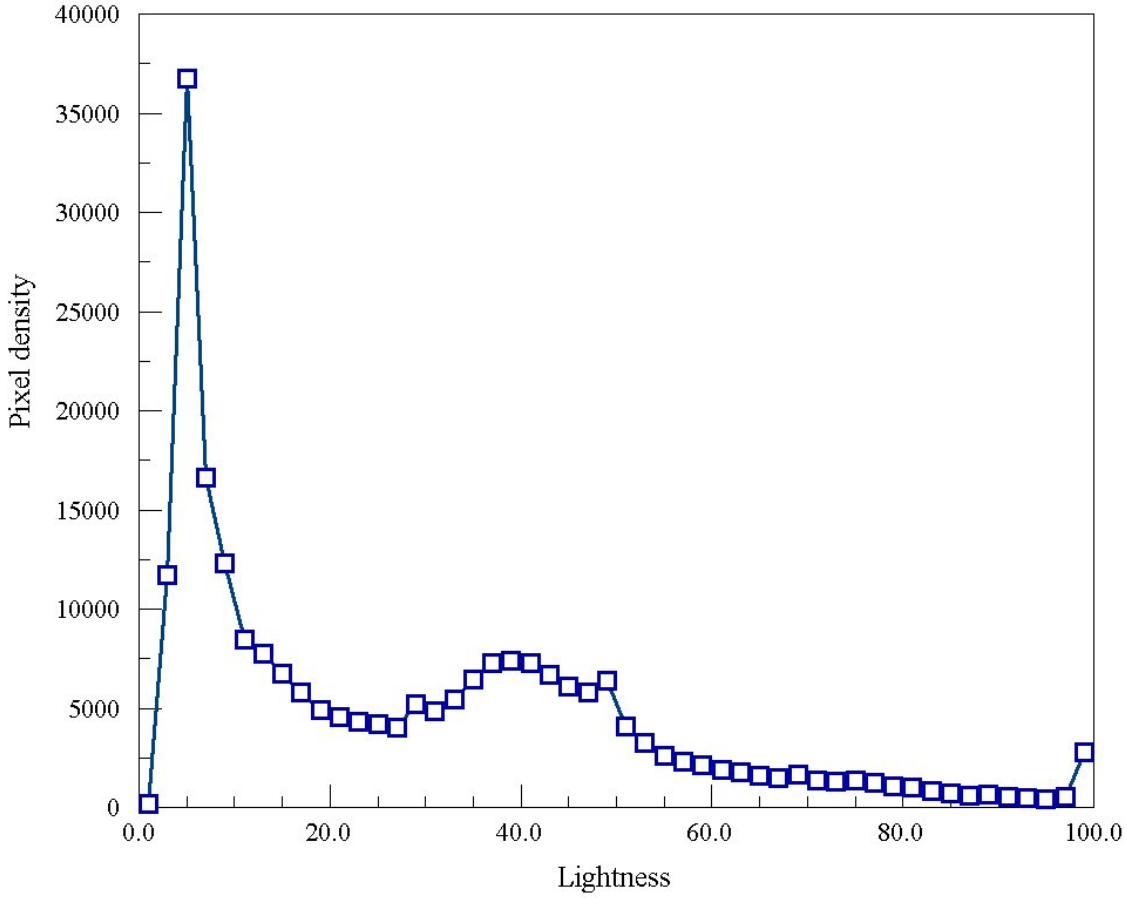

Figure 7.5. Pixel density versus *LIGHTNESS* in the file `BRAIN_MRI.BMP`.

```
   Image file size    NX:  480 NY:  489
   Image number of colors:      256
   Image function limits   Min:    0.000E+00
Max:    1.000E+02
   Image analyzed by LIGHTNESS
      LightMin    LightMax      Pixels
     ================================
        0.0000      2.0000         193
        2.0000      4.0000       11698
        4.0000      6.0000       36758
        6.0000      8.0000       16646
       ....
       94.0000     96.0000         431
       96.0000     98.0000         525
       98.0000    100.0000        2764
```

The data may be helpful in determining good intervals for the division into regions. Figure 5 shows the *LIGHTNESS* distribution of the input image of Fig. 3.

After assigning elements to intervals based on the values of *HUE* or *LIGHTNESS*, **Mesh 5.0** creates the following table:

```
Distribution of elements in intervals
 Interval  FMin        FMax       FAverage     NElem
=================================================
     1   0.000E+00   2.000E+01   7.404E+00     26275
     2   2.000E+01   3.000E+01   2.600E+01      6153
     3   3.000E+01   4.000E+01   3.622E+01     13164
     4   4.000E+01   5.000E+01   4.524E+01     14370
     5   5.000E+01   6.000E+01   5.473E+01      5835
     6   6.000E+01   7.000E+01   6.525E+01      2734
     7   7.000E+01   8.000E+01   7.538E+01      1742
     8   8.000E+01   9.000E+01   8.507E+01      1113
     9   9.000E+01   1.000E+02   9.797E+01      2114
```

The table shows the bounding values of the interval and associated number of mesh elements. The fourth column (marked *FAverage*) is the area-weighted average of *HUE* or *LIGHTNESS* over the assigned elements. This quantity may be helpful dealing with the data images discussed in the next section. The final table shows the association of the intervals with new regions in the mesh. **Mesh 5.0** creates new regions only for intervals that have a non-zero number of elements (*NElem* in the preceding table). For the example, elements have been assigned to all intervals so the program creates nine new regions numbered 2 through 10.

```
Association of regions with element intervals
  RegNo    FMin        FMax        RegName
=========================================
     2   0.000E+00   2.000E+01   ImageInt001
     3   2.000E+01   3.000E+01   ImageInt002
     4   3.000E+01   4.000E+01   ImageInt003
     5   4.000E+01   5.000E+01   ImageInt004
     6   5.000E+01   6.000E+01   ImageInt005
     7   6.000E+01   7.000E+01   ImageInt006
     8   7.000E+01   8.000E+01   ImageInt007
     9   8.000E+01   9.000E+01   ImageInt008
    10   9.000E+01   1.000E+02   ImageInt009
```

7.4. Data image example

As an example of a data image, we shall set up regions to represent the density profile of supersonic gas expanding from a nozzle for a **FullMonte** simulation. The example uses the following input files: `GAS_DENSITY.MIN, GAS_DENSITY.DAT` and `GAS_DENSITY_INT.DAT`. The text file `GAS_DENSITY.DAT` contains information on density profile. The first few lines of the file are listed below:

```
* X(m) R(m) Rho(kg/m3)
145  65
 0.00000 0.0000000000000 102.65000
 0.00000 0.0000961718680 102.65000
 0.00000 0.0001923437440 102.65000
 0.00000 0.0002885156190 102.65000
...
```

The first line is a comment. The second line contains two integer numbers that give the number of columns (*NxImage*) and rows (*NyImage*) in the data array. The remaining lines (one for each mesh vertex) contain three real numbers:

```
X(I,J)   Y(I,J)   F(I,J)
```

where *X* is the horizontal position of the vertex, *Y* is the vertical position and *F* is the value of a quantity. Note that the numbers may be in any valid real-number format separated by the following delimiters: space, tab, comma, colon, equal sign, left paren or right paren. Values are stored in the order

```
DO I=1,NxImage
  DO J=1,NyImage
    (read)
  END DO
END DO
```

In the example the data mesh is not rectangular – it expands radially as *x* increases. Data are recorded in the range x = 0.0 m to x = 0.5 m. We shall use only the first 0.1 m to create a triangular mesh.

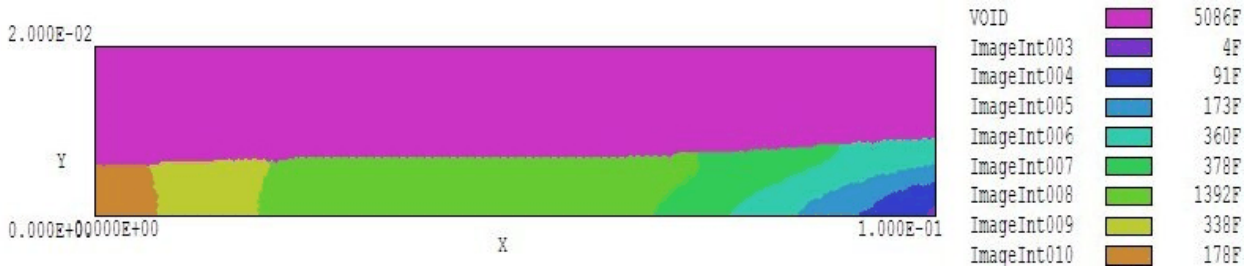

Figure 7.6. Region divisions in the GAS_DENSITY example. Gas expands from a nozzle throat on the left hand size. Region 9 (left-hand size) has density $\rho = 110.0$ kg/m^3 and Region 2 (right-hand-side) has density $\rho = 17.36$ kg/m^3.

Table 7.2. Contents of the file GAS_DENSITY.MIN

```
* Conversion of data on a quadrilateral mesh
* Gas density in kg/m3
* Units in meters

GLOBAL
  XMesh
    0.00 0.100  0.0010
  End
  YMesh
    0.00 0.020  0.0005
  End
END

REGION FILL Void
  L  0.00  0.00  0.10  0.00
  L  0.10  0.00  0.10  0.02
  L  0.10  0.02  0.00  0.02
  L  0.00  0.02  0.00  0.00
END

IMAGE
  DataFile Gas_Density.DAT
  IntervalFile Gas_Density_Int.DAT
  ImageSmooth 8
END

ENDFILE
```

The file GAS_DENSITY_INT.DAT has the entries:

```
8.0
10.40
13.52
17.57
22.83
29.67
38.56
50.12
65.14
84.66
110.00
```

that define intervals for region divisions. Table 7.2 shows the **Mesh 5.0** input script. The *GLOBAL* section defines a rectangle that covers the range 0.0 m ≤ x ≤ 0.1 m and 0.0 ≤ y ≤ 0.02 m. The first *REGION* activates the entire rectangle for data mapping. The image section determines values from the file GAS_DENSITY.DAT and creates new regions according to the intervals in the file GAS_DENSITY_INT.DAT. Figure 7.6 shows the resulting mesh.

Mesh 5.0 makes the following records of the results of the analysis in the file GAS_DENSITY.MLS:

```
Distribution of elements in intervals
 Interval  FMin         FMax        FAverage      NElem
==================================================
     1   8.000E+00   1.040E+01   0.000E+00          0
     2   1.040E+01   1.352E+01   0.000E+00          0
     3   1.352E+01   1.757E+01   1.736E+01          4
     4   1.757E+01   2.283E+01   2.059E+01         91
     5   2.283E+01   2.967E+01   2.641E+01        173
     6   2.967E+01   3.856E+01   3.436E+01        360
     7   3.856E+01   5.012E+01   4.411E+01        378
     8   5.012E+01   6.514E+01   6.003E+01       1392
     9   6.514E+01   8.466E+01   7.285E+01        338
    10   8.466E+01   1.100E+02   9.335E+01        178
```

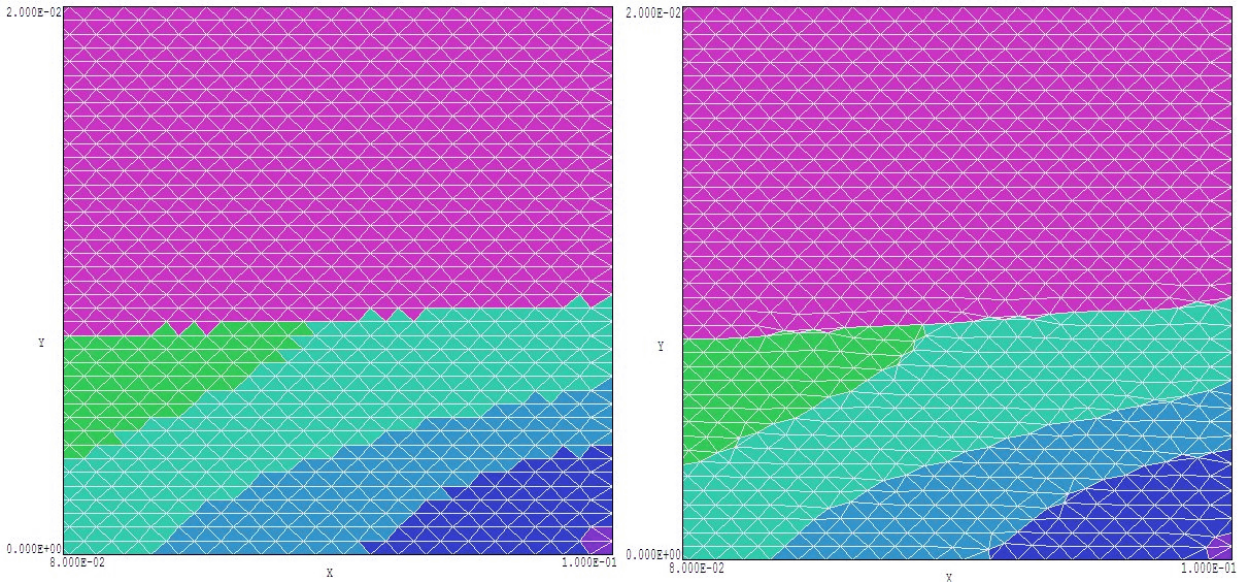

Figure 7.7. Detail of the mesh for example `GAS_DENSITY` with no smoothing (left) and with eight cycles of smoothing (right).

```
Association of regions with element intervals
  RegNo    FMin        FMax        RegName
==========================================
     2   1.352E+01   1.757E+01   ImageInt003
     3   1.757E+01   2.283E+01   ImageInt004
     4   2.283E+01   2.967E+01   ImageInt005
     5   2.967E+01   3.856E+01   ImageInt006
     6   3.856E+01   5.012E+01   ImageInt007
     7   5.012E+01   6.514E+01   ImageInt008
     8   6.514E+01   8.466E+01   ImageInt009
     9   8.466E+01   1.100E+02   ImageInt010
```

The first table states that elements were assigned to only eight of the intervals. Therefore, the program added only eight new regions as shown in the second table. The area-averaged values listed in the first table are used to set material densities for the regions in the **FullMonte** simulation.

To conclude, we shall discuss operation of the *IMAGESMOOTH* function. Figure7.7 shows region boundaries with and without smoothing. Besides a better appearance, boundary smoothing is important if the post-processor for the physical application calculates surface integrals. (For example, surface integrals are used to determine induced charge on a region in an electrostatic simulation.) Jagged surfaces give errors in the calculation as high as 50%. A smooth surface gives a much better approximation, even if the surface does not correspond exactly to the physical surface.

It is important to emphasize that image conversion followed by surface smoothing is an approximating process. For example, if you use a photograph of a cross-section of a mechanical piece, the image function of Mesh 5.0 gives an acceptable mesh representation if the elements are small compared to details in the piece. Nonetheless, the dimensions will not be numerically exact as they would be using line and arc vectors in a standard region section. In summary, image methods are suitable for initial estimates in mechanical systems but the vector method should be used for high-accuracy calculations.

7.5. Image command reference

One or more *IMAGE* sections may be mixed with *REGION* sections. At least one *REGION* section must appear before the first *IMAGE* section. The first region defines the boundary of the solution volume for image clipping. The over-writing rules described in Sect. 7.2 apply to the regions defined by *REGION* and *IMAGE* sections. The following commands may appear in any order within an *IMAGE* section:

IMAGEFILE FileName [XIMin YIMin XIMax YIMax]
IMAGEFILE = tulsa.bmp (0.0, 0.0, 5.0, 5.0)

Load a visual image file in bitmap format (BMP, PCX or PNG). The file must be available in the working directory. If the optional real number parameters *XIMin, YIMin, XIMax* and *YIMax* appear, map the image to a rectangle with corners at (*XIMin,YMin*) and (*XIMax,YIMax*). Otherwise, map the image to the boundaries of the solution rectangle (*XMin,YMin,XMax,YMax*).

DATAFILE FileName
DATAFILE = VarDielectric.DAT

Load a data image file in the free-form text format described in Sect. 4. The file must be available in the working directory.

INTERVALS NInterval [LIN, LOG]
INTERVALS = 15 (LOG)

Set intervals automatically, based on the minimum and maximum values detected in the currently-loaded visual or data image file. The integer parameter *NInterval* is the number of intervals. In

response to the keyword *LIN* the division is linear. The keyword *LOG* gives logarithmic intervals. **Mesh 5.0** reports an error if negative values of the image quantity are detected under the *LOG* option. (Default: *LIN* with *NInterval* = 10).

INTERVALFILE FileName
INTERVALFILE = VarMu_Int.DAT

Read intervals from a file available in the working directory. The file contains a list of interval boundaries, one per line. The file must be available in the working directory. If the file contains the values V_1, V_2, V_3, ... , V_N, the program creates N-1 intervals with boundaries $V_1 \Rightarrow V_2$, $V_2 \Rightarrow V_3$, ..., $V_{N-1} \Rightarrow V_N$.

INTERVALTYPE [HUE, LIGHTNESS]
INTERVALTYPE = Hue

This command is valid only if a visual image has been loaded. It determines whether intervals are created on the basis of pixel *HUE* or *LIGHTNESS*. **Mesh 5.0** converts RGB values to *LIGHTNESS* values in the range 0.0 to 100.0. The *HUE* varies from 0.0° to 360.0°, where 0.0° corresponds to red, 120.0° to green, 180° to cyan and 240° to blue.

IMAGESMOOTH NISmooth
IMAGESMOOTH = 8

The assignment of triangular elements of the foundation mesh to regions leaves jagged boundaries. The smoothing process averages positions along region boundaries (Fig. 7). A zero value of *NISmooth* gives no smoothing and a high value gives strong smoothing. Smoothing is effective only when mesh regions contain extended blocks of contiguous elements. Smoothing should not be used on speckled images where a region may consist of discontinuous single elements. Excessive smoothing may lead to mesh distortions. (Default: *NISmooth* = 0)

Appendix. Format of the Mesh output file

All output files from **TriComp** programs are in ASCII format. This feature makes it easy for you to access information with editors, spreadsheet programs and user analysis programs. You can also use **Mesh** as standalone mesh generator tool for your own applications. The files occupy more space than binary files and take somewhat longer to write. These drawbacks are minor for the modest file sizes associated with two-dimensional solutions.

The format for **Mesh** output files is straightforward. The start of a typical file is illustrated below. Line numbers have been added on the right-hand side to facilitate the discussion. Lines 1 and 8 through 11 are text lines added for readability. They can be ignored by analysis programs.

Lines 2 through 7 gives information about the spatial limits and indices of the foundation mesh. The formats used are E13.6 (columns 7-19) and I4 (columns 7-11). The main mesh data is contained in Lines 12-15 and following lines. There is one line for each node The first two quantities in a line are the (*k*,*l*) indices (horizontal and vertical) of the node. The third quantity, *RegNo*, is the region number associated with the node. Nodes outside the solution volume have *RegNo* = 0. The structured meshes used in **TriComp** have six elements surrounding each vertex. On the average there two elements per vertex. Therefore, each node line contains the region numbers of two elements. By convention, the elements associated with a node are 1) upward and to the right (*RegUp*) and 2) downward and to the right (*RegDn*). Elements outside the solution volume have *RegUp* = 0 or *RegDn* = 0. The final two real number parameters are the (*x*,*y*) or (z,r) coordinates of the node. A data line has the following FORTRAN format: (5I4, 2E14.6).

```
--- Run parameters ---                                              (1)
XMin:  0.000000E+00                                                 (2)
XMax:  5.000000E+00                                                 (3)
KMax:   51                                                          (4)
YMin:  0.000000E+00                                                 (5)
YMax:  5.000000E+00                                                 (6)
LMax:   51                                                          (7)
                                                                    (8)
--- Vertices ---                                                    (9)
  k    l  RgNo RgUp RgDn       x             y                    (10)
 ================================================================   (11)
   1    1    4    1    0  0.000000E+00  0.000000E+00               (12)
   2    1    4    1    0  1.030236E-01  0.000000E+00               (13)
   3    1    4    1    0  2.058450E-01  0.000000E+00               (14)
   4    1    4    1    0  3.083962E-01  0.000000E+00               (15)
   5    1    4    1    0  4.106382E-01  0.000000E+00               (16)
```