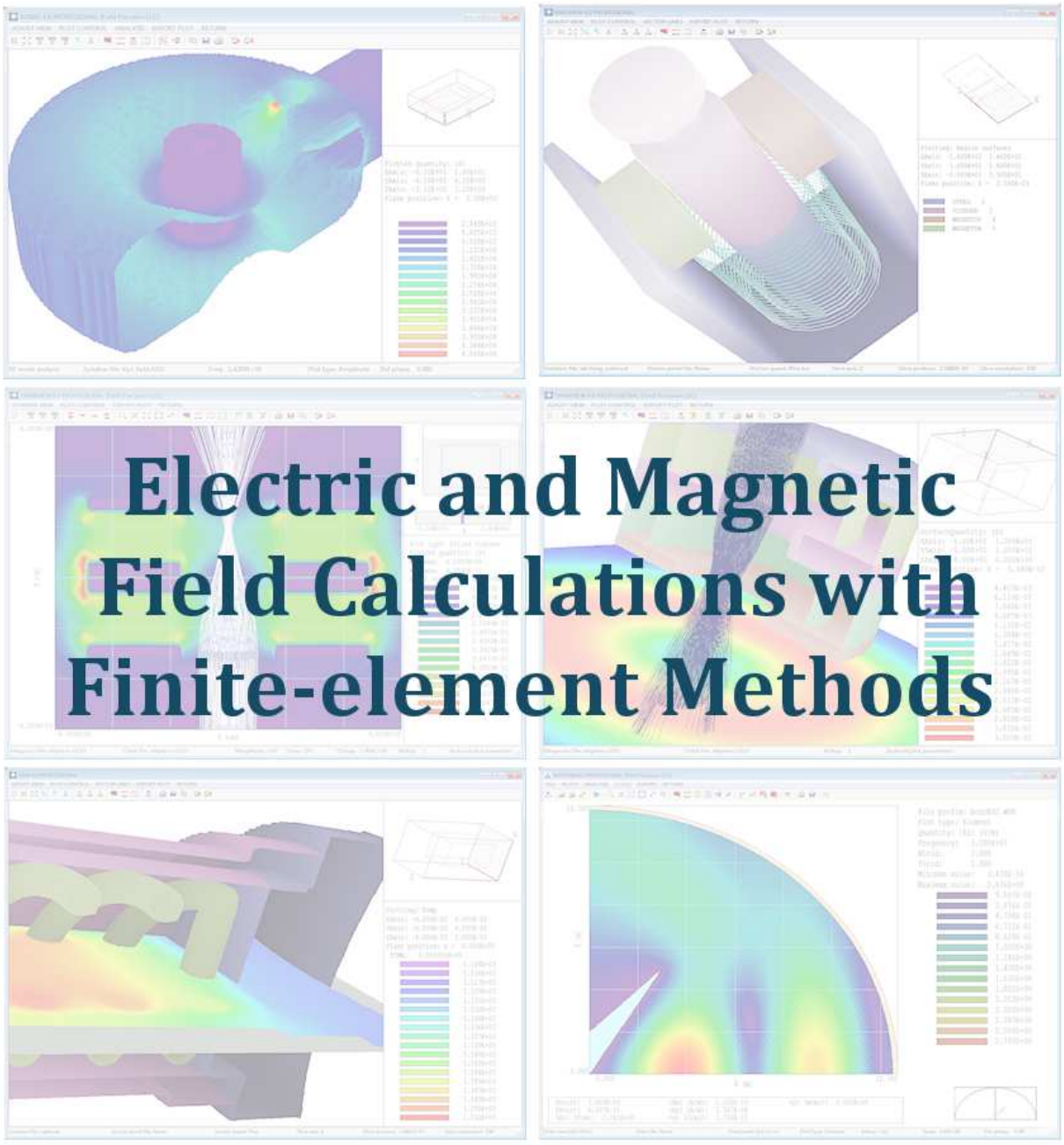


Electric and Magnetic Field Calculations with Finite-element Methods

Electric and magnetic field calculations with finite-element methods

Stanley Humphries, Ph.D.
President, Field Precision LLC
Professor Emeritus, University of New Mexico

Published by
Field Precision LLC
E mail: techinfo@fieldp.com Internet: https://www.fieldp.com

Contents

1 Introduction

Field Precision finite-element programs cover a broad spectrum of physics and engineering applications, including charged particle accelerators and X-ray imaging. The core underlying most of our software packages is the calculation of electric and magnetic fields over three-dimensional volumes. To use our electric and magnetic fields software effectively, researchers should have a background in electromagnetism and should be able to make informed decisions about solution strategies. First-time users of finite-element software may feel intimidated by these requirements. My motivation in writing this book is to share my experience in field calculations. I hope to build users' knowledge and experience in steps so they can apply finite-element programs confidently. In the end, readers will be able to solve real-world problems with the following programs:

EStat (2D electrostatics)

HiPhi (3D electrostatics)

PerMag (2D magnetostatics)

Magnum (3D magnetostatics)

To begin, it's important to recognize the difference between 2D and 3D programs. All finite-element programs solve fields in three-dimensions, but often systems have geometric symmetries that can be utilized to reduce the amount of work. The term 2D applies to the following cases:

Cylindrical systems with variations in r and z but no variation in θ (azimuth).

Planar systems with variations in x and y and a long length in z.

Which brings us to the first directive of finite-element calculations: never use a 3D code for a calculation that could be handled by a 2D code. The 3D calculation would increase the complexity and run time with no payback in accuracy.

We need to clarify the meaning of *static* in electrostatics and magnetostatics. The implication is that the fields are constant or vary slowly in time. The criterion of a *slow* variation is that the systems do not emit electromagnetic radiation. Examples of electrostatic applications are power lines, insulator design, paint coating, ink-jet printing and biological sorting. Magnetostatic applications include MRI magnets, particle separation and permanent magnet devices. A following coarse will cover simulations of electromagnetic radiation (*e.g.*, microwave devices).

Secondly, it's important to have a clear understanding of the purpose of computer calculations of electric and magnetic fields. Numerical methods should be used when it is not possible to generate accurate results with analytic methods. Numerical solutions are necessary in the following circumstances:

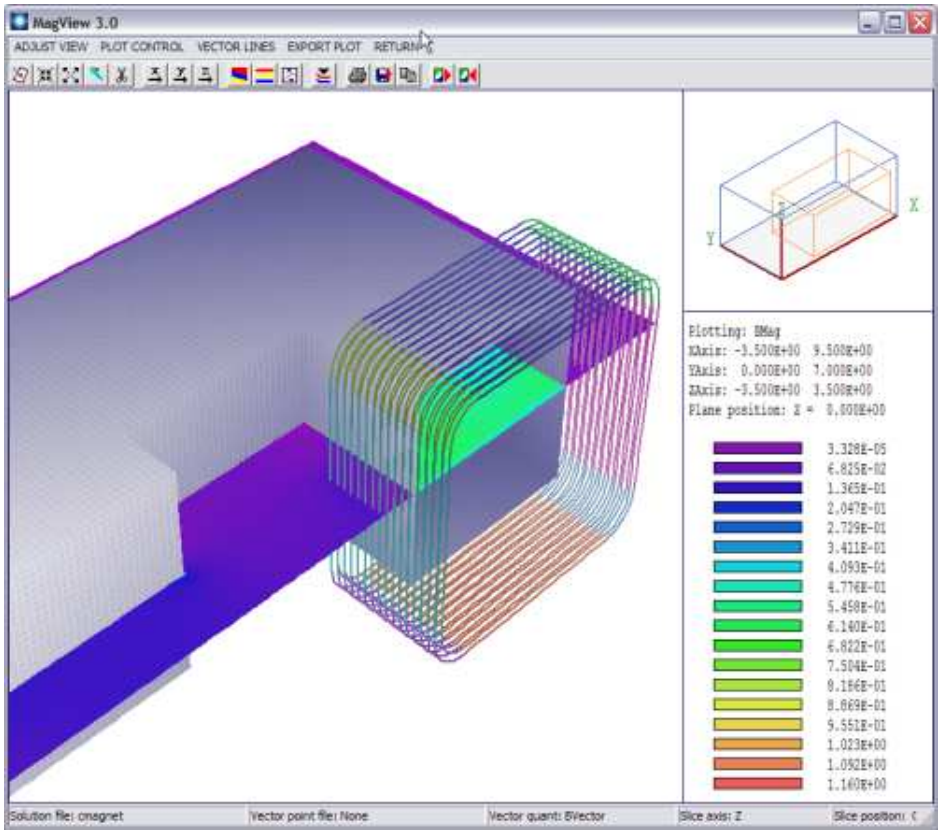


Figure 1: Screenshot of the **MagView** postprocessor for 3D magnetostatics.

The system has a complex asymmetric geometry.

The solution volume contains many objects with different material properties.

Materials have complex properties (*e.g.*, saturation of iron in magnetic circuits)

In an ideal case, a user makes analytic estimates of field values and then applies numerical methods to improve the accuracy. The initial analysis gives an understanding of the physics involved and the anticipated scales of quantities – essential information for effective solution setups. The worst case is when a user treats a program as an omniscient black box. No matter what software manufacturers may claim, using a field program without understanding fields is at best a gamble. Sometimes you may get lucky, but most of the time considerable effort is wasted generating meaningless results.

In summary, I would like to help you become an informed software user. I suggest you start by downloading a free textbook that will help you brush up on electric and magnetic field theory. The book also gives a detailed description of the FEM techniques I will discuss:

S. Humphries, **Finite-element Methods for Electromagnetics** (CRC Press, Boca Raton, 1997) (available for free download at `http://www.fieldp.com/femethods.html`).

2 First 2D electrostatic solution

In preparation, install either the professional or basic version of the **Electrostatics Toolkit**. In this chapter, we'll run through the steps of the solution and analysis of a 2D electrostatic problem without going into detail. The goal is a quick demo of the capabilities of **EStat**. Subsequent articles will cover details of program techniques.

A notable feature of our programs is dual input – there are two options for supplying geometric and material data for solutions:

> **Interactive**: the standard method for modern finite-element programs where you fill in items in dialogs. This option is useful for new users and for a quick setup of a new system.
>
> **Text**: the classic method using input scripts. This option allows experienced users to make changes to setups easily and allows automatic program operation under the control of external programs or batch files. Scripts also provide a permanent archive of setups.

In this demo calculation, we will check out prepared input scripts before running them using the built-in text editors of **Mesh** and **EStat**. For more detailed work, it's useful to have a good text editor like ConText. Before starting, we'll need to make some provisions for data organization. A little effort in the beginning circumvents headaches later. Using a file manager (like FreeCommander), navigate to a location where you would like to create a general directory for your finite-element calculations and create the directory `Simulations` Create the sub-directories `Electrostatics` and `Electrostatics\Practice`. In the right-hand window, navigate to `C:\fieldp_basic\tricomp\Examples\EStatExamples`. We will copy a prepared example for our work. Highlight the files with prefix `ElectronDiode` and copy them to the `Practice` directory. Figure 2 shows the resulting setup.

Click the desktop shortcut to run **FPController** (`fpcontroller.exe`, Fig. 3). The **Mesh** and **EStat** buttons should be active as shown. If not, click the *Program folder* button and navigate to `C:\fieldp_basic\tricomp`. Click the *Data folder* button and navigate to `c:\Simulations\Electros` Subsequently, all input/output operations will target this folder.

There are three steps in a finite-element solution:

> Define the geometry of the solution space and divide objects into small volumes (*elements*). The process is called *mesh generation*.
>
> Create and solve a large set of linear equations to approximate the governing partial differential equation (such as the Poisson equation for electrostatics). The goal is to determine the electrostatic potential on points of the mesh (*nodes*).
>
> Analyze the results – use the potential values to find physical quantities of interest (*e.g.*, electric field, field energy, capacitance,....).

The first function is performed by the **Mesh** program (`mesh.exe`) and the second and third functions by the **EStat** program (`estat.exe`). Meshing is performed by a separate program because the same mesh may be used for different types of solutions. Output from **Mesh** is compatible with **PerMag** (magnetic fields), **TDiff** (thermal transport) and other solution

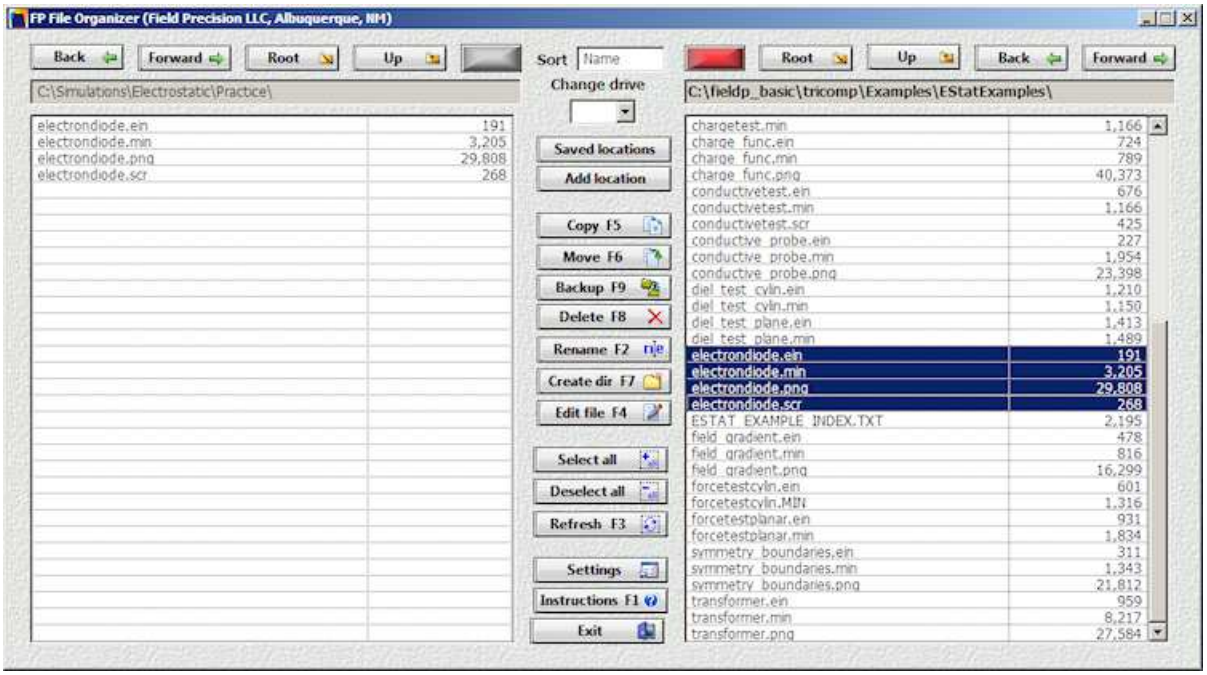


Figure 2: Set up a data directory and copy the examples.

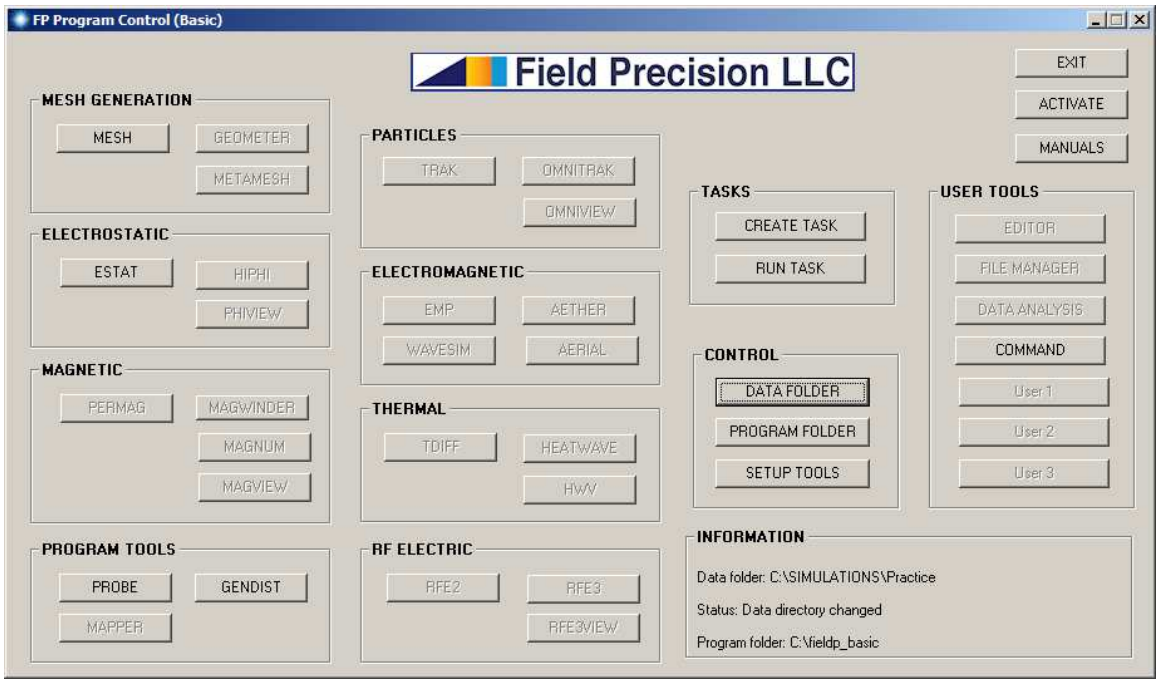


Figure 3: **FPController** program launcher.

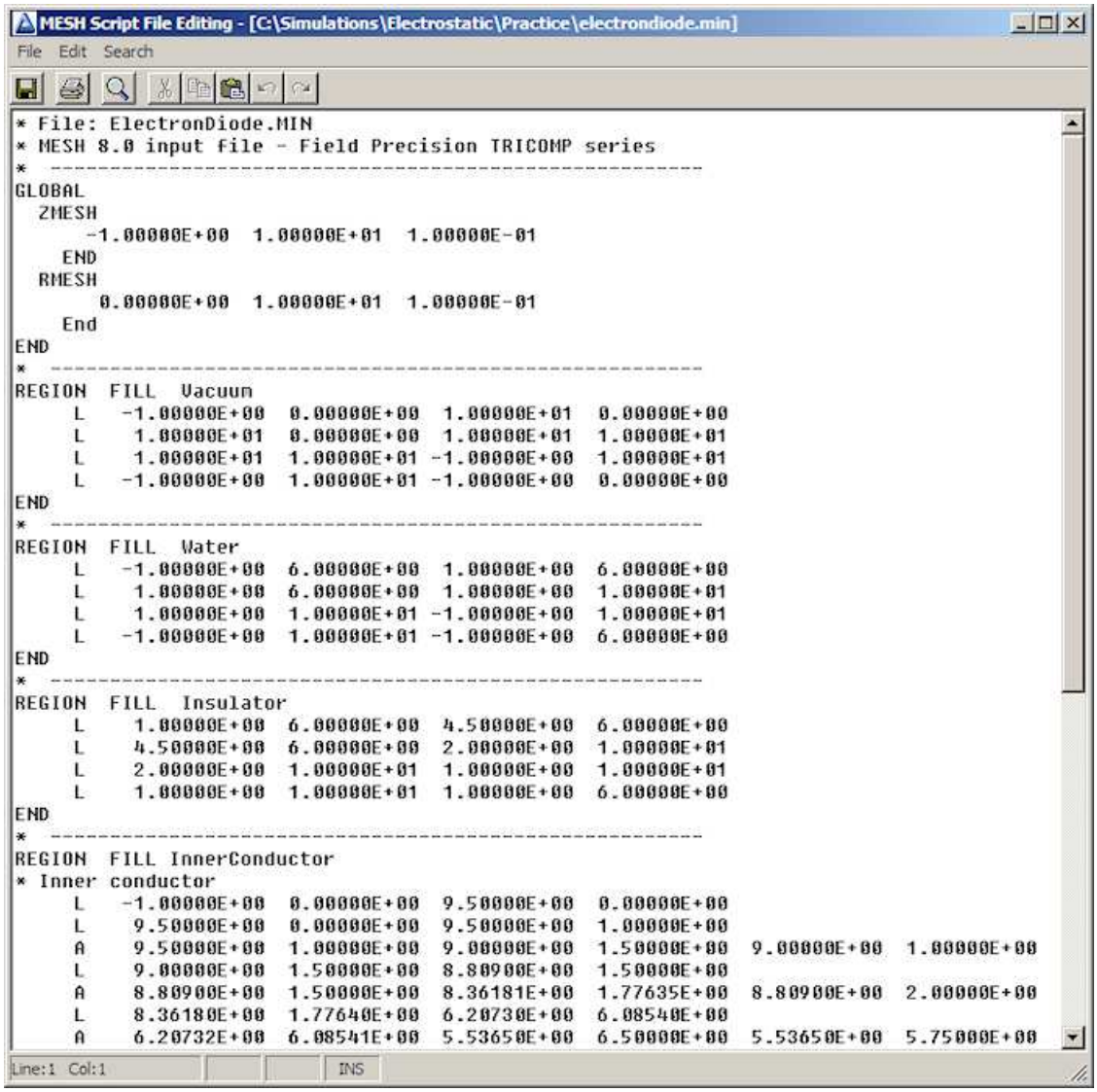


Figure 4: Internal **Mesh** editor display.

programs of the **TriComp** series. In other words, what we learn about **Mesh** for electrostatics is also useful for the magnetic solutions we will discuss later.

Click the **Mesh** button to open the program. Initially, the screen is blank. Choose *File/Edit file* from the menu at the top. The selection dialog shows the four files in the data folder. Pick `ElectronDiode.MIN`, where `MIN` designates **M**esh **IN**put. The internal editor shows the file contents (Fig. 4), a set of organized numbers. For now, note that there are *Region* sections that represent the different physical objects in the solution space. Each region section contains a set of line or arc vectors that gives the region shape. The numbers are the coordinates of the vectors. There are two ways to create or to modify the content of `MIN` files:

Use the **Mesh** *Drawing Editor*, an interactive 2D CAD program.

Use a text editor to change numbers directly.

We will discuss both options in following chapters. For now, exit the editor with no changes.

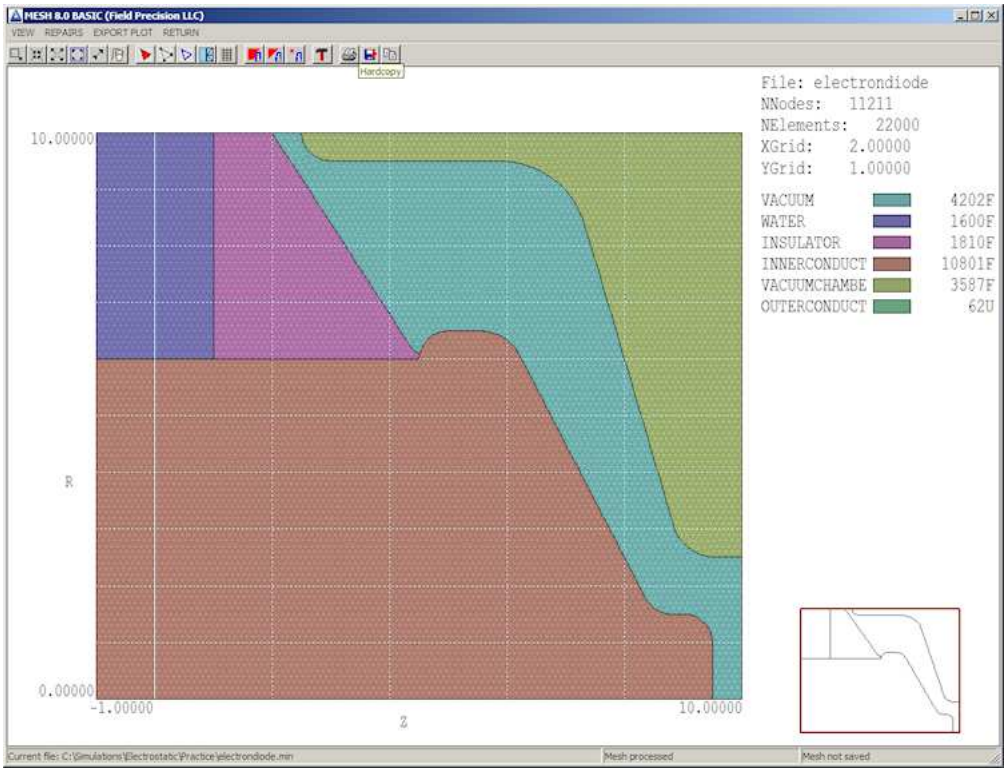


Figure 5: View of the full mesh.

Choose *File/Load script(MIN)* in the menu or use the *Open-file* tool on the left hand side of the toolbar to load the contents of the file `ElectronDiode.MIN`. Then pick *Process* or click the tool with the green square. In response, **Mesh** analyzes the desired element sizes and the region vectors to create a set of small elements. To view the result, choose *Plot/Repair* from the menu or click the *Plot/repair* tool to show the display of Fig. 5. The solution volume has been divided into the regions listed in the script to represent physical objects (electrodes and dielectrics). The cross section has been divided into triangular areas. Note that the calculation represents a cylindrical system, a figure of revolution about the bottom boundary ($r = 0.0$). When first viewing a z-r plot, many users ask where's the bottom? The answer is that there is no bottom. Negative values of the radius r are undefined. On the other hand, a plot in a Cartesian slice plane like $y = 0.0$ would have both upper and lower components. In cylindrical solutions, elements are tori with triangular cross-sections that extend over the full range of azimuth ($\theta = 0^{\circ}$ to 360°).

Let's take a closer look at the mesh. Choose *View/Zoom window* and specify the corners of a box by moving and left-clicking the mouse. The magnified view of Figure 6 gives a better look at the element cross-sections. The inset at lower-right shows the view limits. Note that the triangles were flexed for a good match to region boundaries. The fitting allows high-accuracy calculations of surface fields. A mesh with element shapes customized to the geometry is called a *conformal* mesh. To conclude work with **Mesh**, return to the main menu and choose *File/Save mesh (MOU)*. Refresh the display in **FP File Organizer** (press F3). Two new files have been added to the folder:

`ElectronDiode.MLS`: a text listing file of diagnostic information that may be useful if there is a problem.

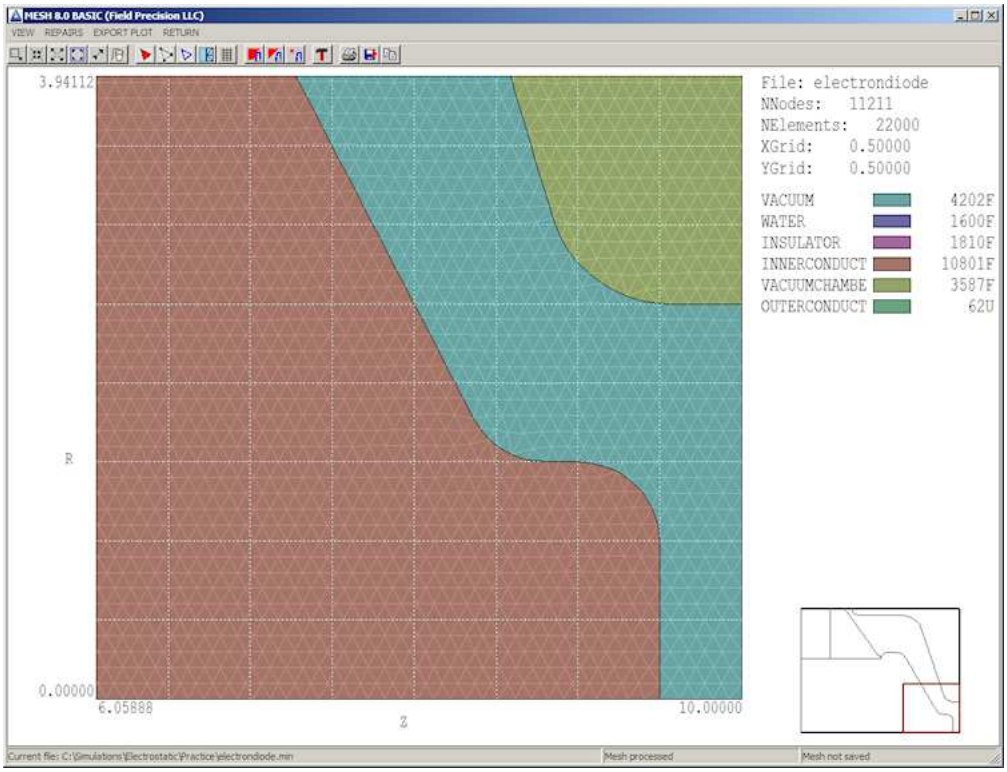


Figure 6: Detailed view of a mesh section.

`ElectronDiode.MOU`: the main output file specifying element identity (region association) and coordinates of nodes (boundaries between elements). This file may be ported to any of the **TriComp** solution programs. The file is in text format, so you can inspect it with an editor.

Next, run **EStat** from **FPController**. Choose *File/Edit script (EIN)* and pick the file `ElectronDiode.EIN`. The editor shows the content

```
* File ElectronDiode.EIN
DUnit = 39.37
Geometry = Cylin
Epsi(1) = 1.0
Epsi(2) = 81.0
Epsi(3) = 2.7
Potential(4) = -500.0E3
Potential(5) = 0.0E3
Potential(6) = 0.0E3
ENDFILE
```

The two general commands at the top specify that the dimensions from the **Mesh** file should be interpreted in inches and that the system has cylindrical symmetry. The other commands with indices set the physical properties of regions of the solution volume. The first three regions are dielectrics (vacuum, purified water and cast epoxy) and the other regions are fixed-potential electrodes. Again, there are two options for creating **EStat** input scripts – supplying values in an interactive dialog or direct input with a text editor.

The tools labeled *1, 2* and *3* invoke the three main functions of **EStat**:

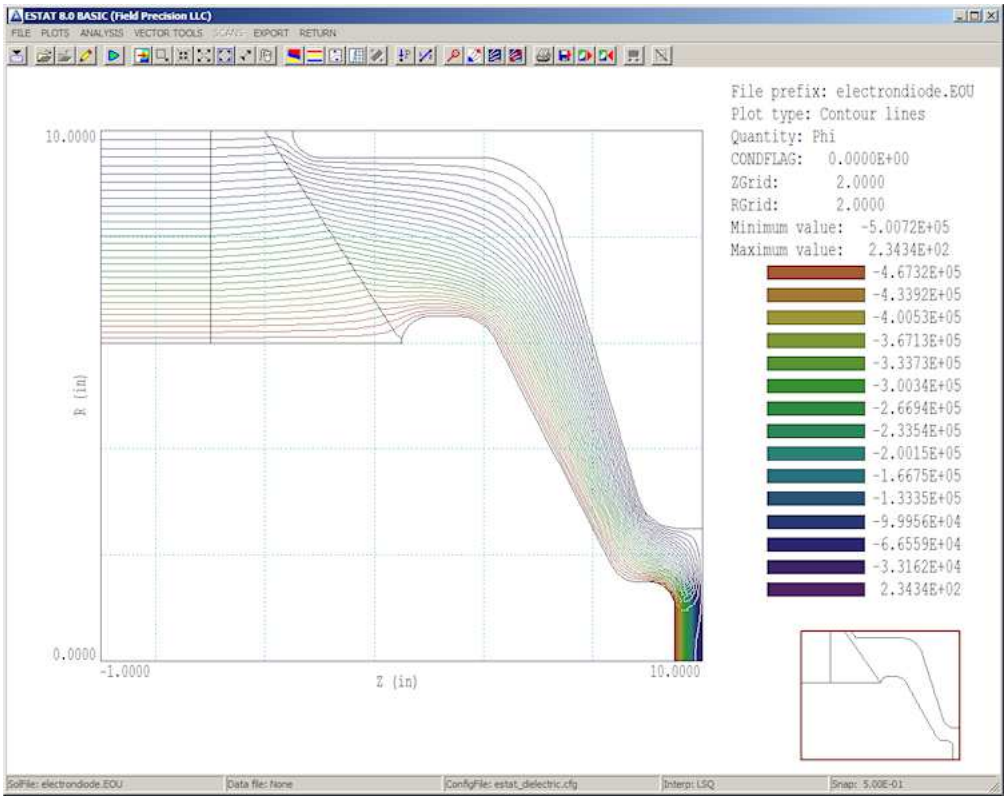


Figure 7: Equipotential line plot of the electrostatic solution.

Create the input script.

Perform the finite-element solution.

Analyze the results.

Step *1* has already been performed, so click the tool marked *2* and pick `ElectronDiode.EIN`. **EStat** reads the geometry data in `ElectronDiode.MOU`, generates the set of finite-element coupled linear equations and solves them, all within a second. **FP File Organizer** shows that two new files have been created: `ElectronDiode.ELS` (a diagnostic text listing file) and `ElectronDiode.EOU` (the main solution file containing potential values at nodes). Click the *3* tool in **EStat** and pick `ElectronDiode.EOU`. The program shifts to the *Analysis* menu and displays the default equipotential-line plot of Fig. 7. This type of plot is useful for experienced users because it shows complete information. Section 7.5 of the companion text **Finite-element Methods for Electromagnetics** describes how to interpret equipotential plots.

There are many capabilities of the *Analysis* menu that you can explore. Let's check out two of them. A plot of the magnitude of the electric field $|\mathbf{E}|$ is useful to pinpoint areas where breakdowns may occur in high-voltage systems. To create the plot, press *Plots/Plot settings/Plot type* in the menu and choose *Element*. Then press *Plots/Plot settings/Plot quantity* and pick $|\mathbf{E}|$. The resulting plot (Fig. 8) shows that the peak field in the electron emission region is about 442 kV/cm and that the maximum field on the insulator vacuum surface is about 40 kV/cm. A second activity is to find the capacitance of the system downstream from the insulator. We can determine this quantity from the volume integral of electrostatic field

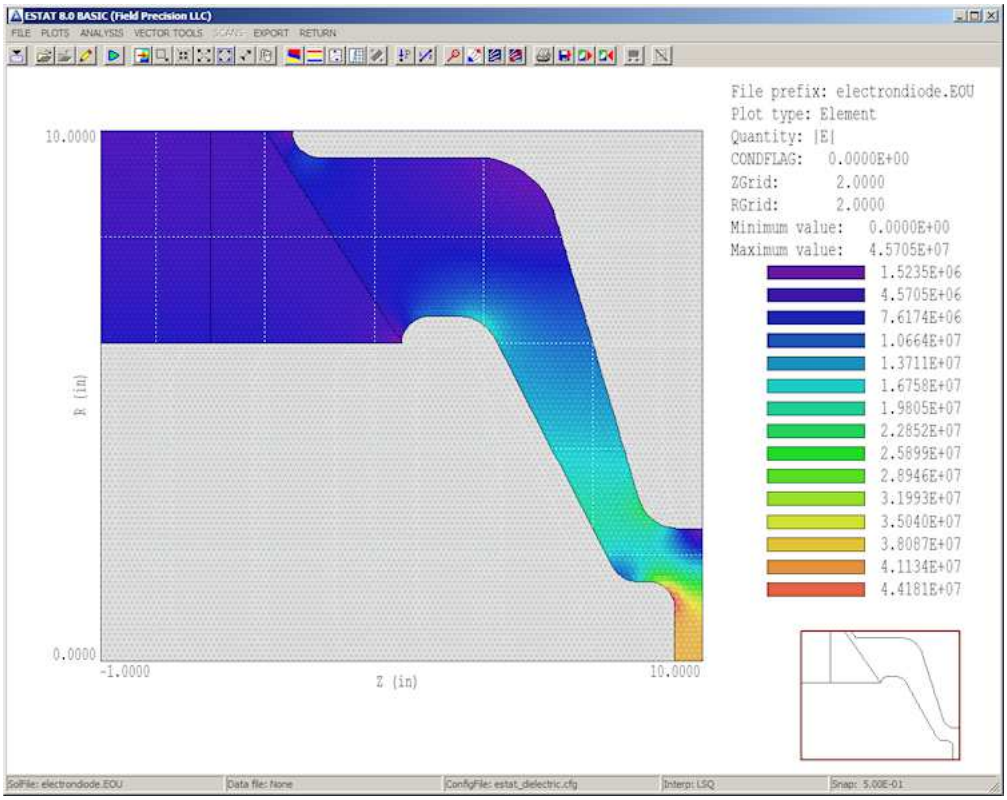


Figure 8: Variation of electric field magnitude in the solution volume.

energy density in the vacuum region. Press *Analysis/Volume integral*. **EStat** offers to open a data record file with the default name `ElectronDiode.DAT`. Global information will appear on the screen, but we need to check the file for a detailed breakdown by regions.

Press *File/Close data record* and then *File/Edit files*. Open `ElectronDiode.DAT`. Here is the information of interest:

```
Quantity: Energy
Global integral:   6.358995E+01
RegNo   Integral
=========================
 1    5.534785E+00
 2    5.602793E+01
 3    2.027232E+00
 4    4.472539E-29
```

The capacitance of the vacuum volume (Region 1) may be determined from the equation $U_e = CV^2/2$ as $(2)(5.5347/(5.0 \times 10^5)^2 = 44.3$ pF.

We have completed the first electrostatic solution. An inspection of the final state of the data folder shows two features of Field Precision programs:

> The compact input files `ElectronDiode.MIN` and `ElectronDiode.EIN` contain complete information to reconstruct the solution.
>
> All the data that we generated has been recorded in accessible files. For example, if you archive the file `ElectronDiode.EOU`, you can always return for additional analyses.

The purpose of this example was to give an overview of the program features. In following chapters, we shall create solutions from scratch, learning about meshing and analysis techniques.

3 Electrostatic application: building the mesh

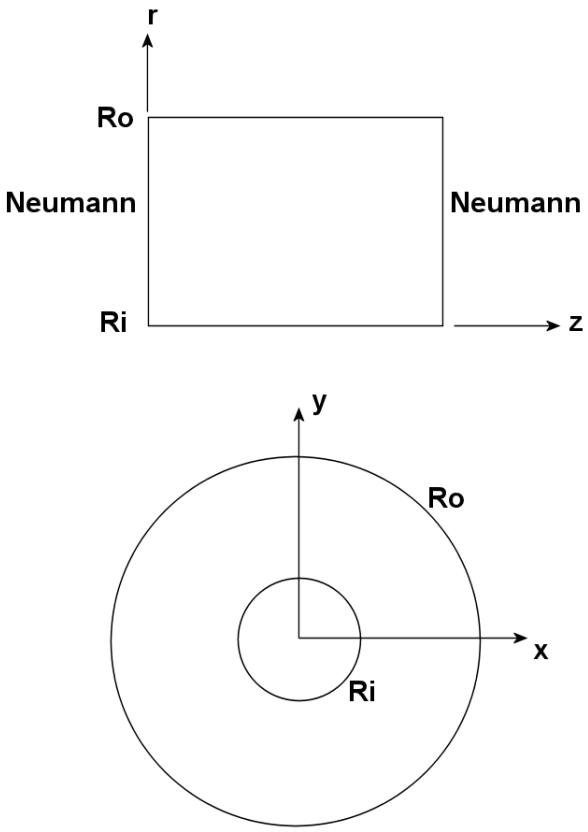


Figure 9: Alternative ways to model coaxial cylinders with a 2D code.

In the previous chapter, we followed a prepared example without going into the details of how the input files were generated and how the mesh parameters were chosen. Now, we're ready to build a complete solution and to learn about how the geometry of the mesh affects the accuracy of the solution. The best approach to make comparisons is to model a system that has an analytic solution. A good choice is a set of coaxial cylindrers with an applied voltage where there are spatial variations of potential and electric field.

There are two ways to model coaxial cylinders with a 2D code (Fig. 9). The first is to use cylindrical coordinates z-r and to model the infinite length in z with Neumann conditions[1] at the upper and lower z boundaries. This approach, where the electrode boundaries are simply straight lines, would not exercise the conformal mesh capability. Instead, we will use planar coordinates (Fig. 9, lower) where the cross section is in the x-y plane and the system extends infinitely in z (out of the page). For specific parameters, take $r_i = 5.0$ cm, $r_o = 15.0$ cm and $V_i = 100.0$ V. The space between the cylinders is filled with polyethylene ($\epsilon_r = 2.7$) and the outer electrode is grounded. Using formulas available in introductory electromagnetism texts,

[1]The *Neumann condition* specifies that the normal component of electric field, E_z, is zero at the boundaries.

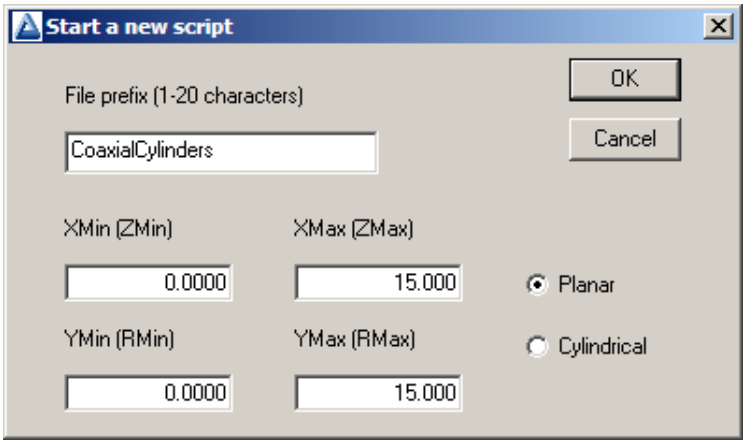


Figure 10: Dialog to start an interactive drawing session.

the radial variation of potential and electric field and the capacitance per unit length is given by:

$$\begin{aligned} E_r(r) &= \frac{91.024}{r}, \\ \phi(r) &= 100.0 \left[1 - \frac{\ln(r/5)}{1.099}\right], \\ c &= 1.3567 \times 10^{-10}. \quad \text{(F/m)} \end{aligned} \tag{1}$$

To start, we will define the geometry for the numerical calculation. Run the **Mesh** program and choose *File/Create script/Create script graphics*. Fill out the dialog entries as shown in Fig. 10. For the dimensions given, note that the solution volume covers only the first quadrant of the space. We shall apply symmetry to reduce the amount of work – why calculate all four quadrants when the solution is the same in each one?

When you click *OK*, the program enters the drawing editor of Fig. 11. The display includes a menu at the top, a set of useful drawing tools beneath it, the main drawing area and a status bar at the bottom. We will define the following physical regions:

Region 1: the polyethylene dielectric between the cylinders.

Region 2: the inner boundary at 100.0 V.

Region 3: the outer boundary at 0.0 V.

It is important to note that as we enter regions in sequence, the present region over-writes any shared elements or nodes of previous ones.

By default, the drawing editor is ready to add outline vectors for *Region 1*. Click the *Line* tool and move the mouse cursor into the drawing area. Note that the cursor changes to a cross and that there is an orange box showing the current coordinates. Snap mode is in effect by default, and the box moves in discrete steps to exact coordinate locations. Move the mouse cursor to the origin ($x = 0.0$, $y = 0.0$) and then click the left mouse button to set to start point of a line vector. Then move to the end of the x axis ($x = 15.0$, $y = 0.0$) and click the button again to set the end point. A line in the color of *Region 1* appears along the bottom.

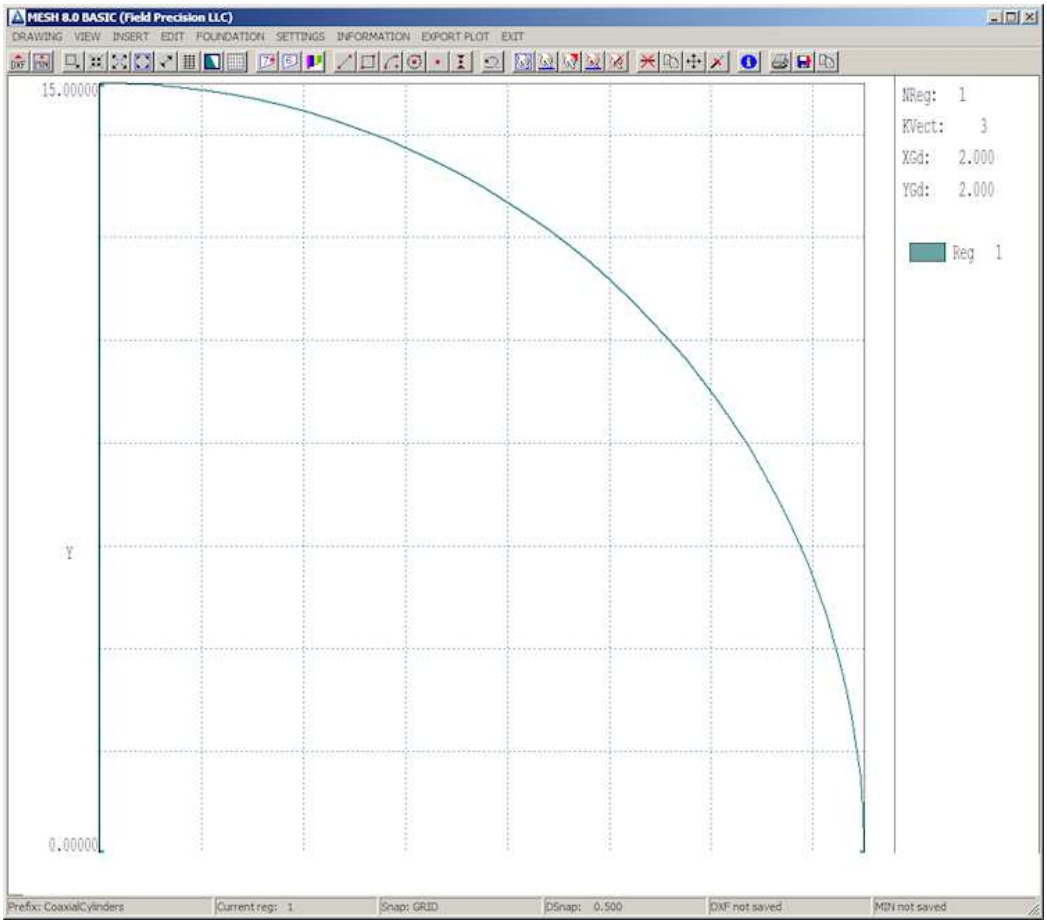


Figure 11: The **Mesh** drawing editor.

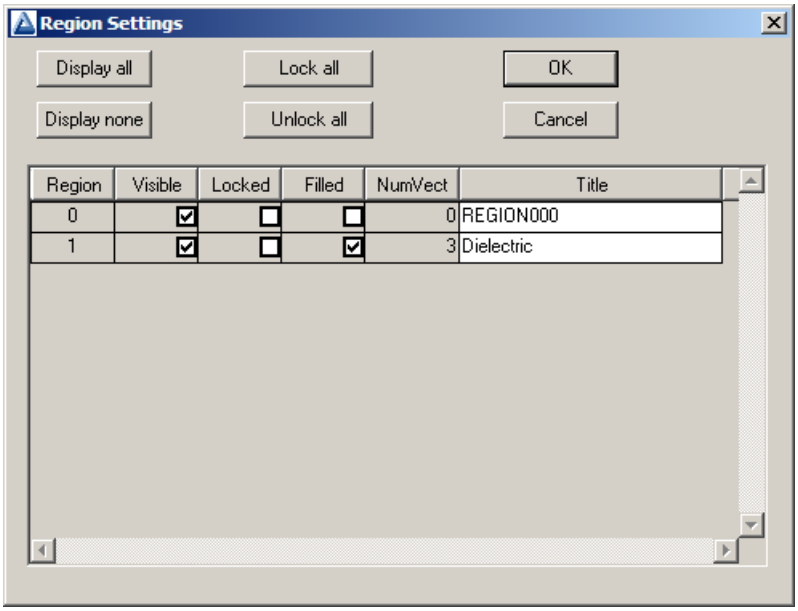


Figure 12: Region properties dialog.

The program remains in line entry mode. Draw another vector along the y axis from [0.0, 0.0] to [0.0, 15.0]. Right click the mouse to exit line entry mode. To add the circular edge of the region, choose the *Arc/Start-end-center* tool. Move the mouse to the following locations in sequence and click the left button at each one: [15.0, 0.0], [0.0, 15.0], [0.0, 0.0]. You should see the arc vector of Fig. 11.

With the outline of *Region 1* complete, we shall set the region properties. Choose the command *Settings/Region properties* to bring up the dialog of Fig. 12. *Region 0* is a special place to store reference vectors that will not appear in the output script. Supply the name *Dielectric* for *Region 1* and check the *Filled* box. For a filled region, **Mesh** assigns not only the nodes of the boundary to *Region 1*, but also the elements enclosed. The *Filled* property applies to regions with non-zero volume. Exit the dialog.

Next, we are going to create the inner electrode by over-writing a portion of the dielectric volume. Click the *Start next region* tool, Now, all vectors you enter will be associated with *Region 2*. This region has the same shape as *Region 1* except that the radius is 5.0 rather than 15.0. Use the same procedure except the first line should extend from [0.0, 0.0] to [5.0, 0.0] and so forth. When complete, go to *Settings/Region properties*. Assign the name *InnerElectrode* to *Region 2* and set the *Filled* property. Exit the dialog.

To check the work so far, click the *Toggle fill display* tool to bring up the plot of Figure 13. The fill display mode is both a diagnostic and a visual aid. It checks that the vectors of filled regions define a closed surface and shows how the enclosed elements have been assigned. Click the tool again to return to the normal vector display.

We'll conclude by defining *Region 3*. This region is not a filled volume but rather a set of nodes set to the fixed-potential condition on the outer boundary. We call such a region an un-filled or *line* region. Click the *Start next region* tool and then the *Arc/Start-End-Center* tool. Move to the following coordinates in sequence and click the left mouse button: [15.0, 0.0], [0.0, 15.0], [0.0, 0.0]. The nodes on the outer edge of *Region 1* are re-assigned to *Region 3* (color-coded violet). Set the region name to *OuterBoundary* and be sure that the *Filled* box in unchecked. You may ask whether it's necessary to set special conditions on the dielectric

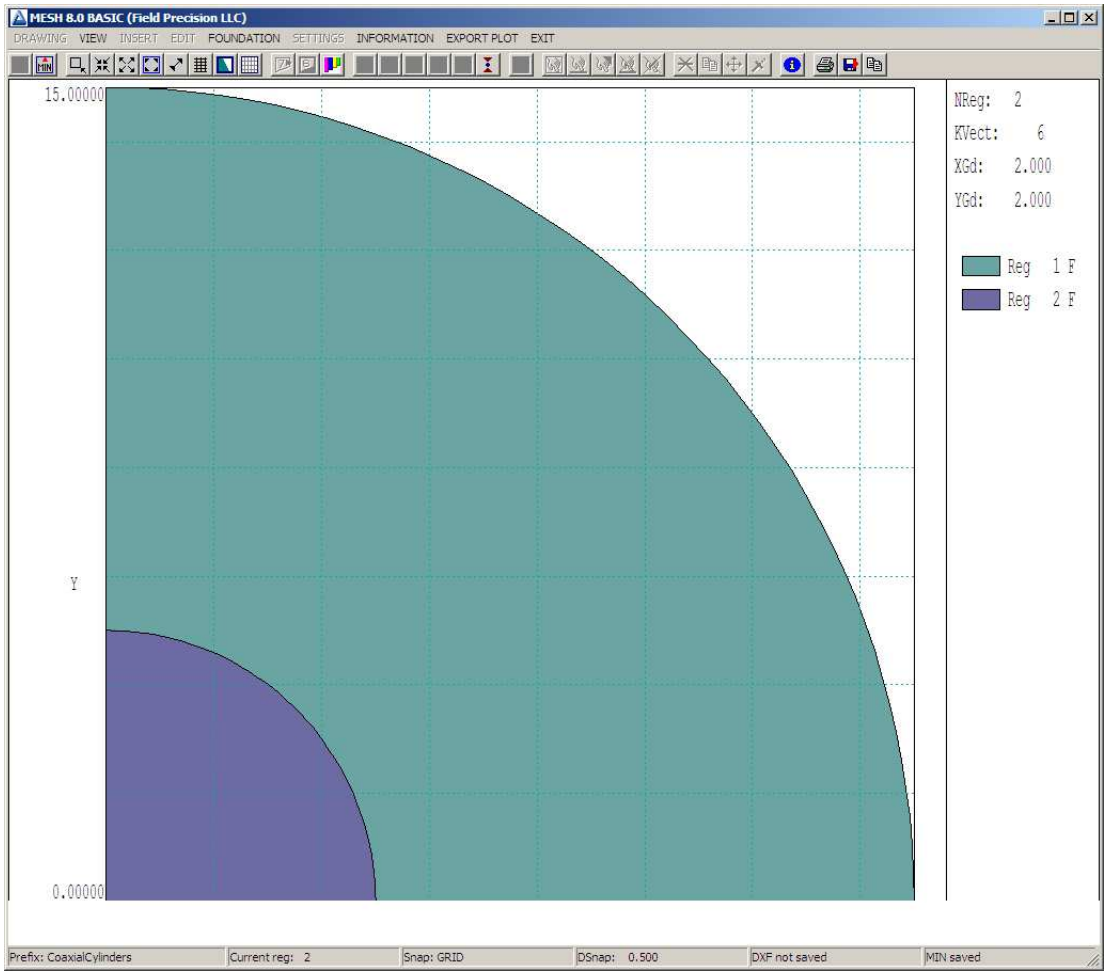


Figure 13: Checking the integrity of filled *Regions 1* and *2*.

boundaries on the bottom and left sides. A useful feature of finite-element solutions is that unspecified boundaries automatically assume the Neumann condition, exactly what we what. We'll simply leave the boundaries alone.

Click the *Export MIN* tool to save a copy of the work and then exit the drawing editor. To see the result of the work, choose *File/Edit file* and pick `CoaxialCylinders.MIN`. Here is the content:

```
* File: CoaxialCylinders.MIN
*  ------------------------------------------------------
GLOBAL
  XMESH
        0.00000   15.00000    0.20000
  END
  YMESH
        0.00000   15.00000    0.20000
  END
END
*  ------------------------------------------------------
REGION  FILL DIELECTRIC
  L     15.00000    0.00000    0.00000    0.00000
  L      0.00000    0.00000    0.00000   15.00000
  A      0.00000   15.00000   15.00000    0.00000    0.00000    0.00000
END
*  ------------------------------------------------------
REGION  FILL INNERELECTRODE
  L      5.00000    0.00000    0.00000    0.00000
  L      0.00000    0.00000    0.00000    5.00000
  A      0.00000    5.00000    5.00000    0.00000    0.00000    0.00000
END
*  ------------------------------------------------------
REGION OUTERBOUNDARY
  A     15.00000    0.00000    0.00000   15.00000    0.00000    0.00000
END
*  ------------------------------------------------------
ENDFILE
```

The region vectors and coordinates should look familiar. The commands at the top set the target element sizes. For now, we'll accept the defaults.

The final step is generate a mesh file (`CoaxialCylinders.MOU`) from the region boundary specifications (`CoaxialCylinders.MIN`). In the main **Mesh** menu, choose *File/Load/Load script (MIN)* and pick the file you just created. Choose *Process* and then **Plot-repair** to view the result (Fig. 14). Is it a good mesh? Qualitatively, the answer is yes because the boundaries are relatively smooth and all features are well resolved (*i.e.*, spatial variations of potential will be spread over many elements). We shall make this conclusion more quantitative in future chapters.

In the next chapter, we will use the mesh for field calculations and comparisons.

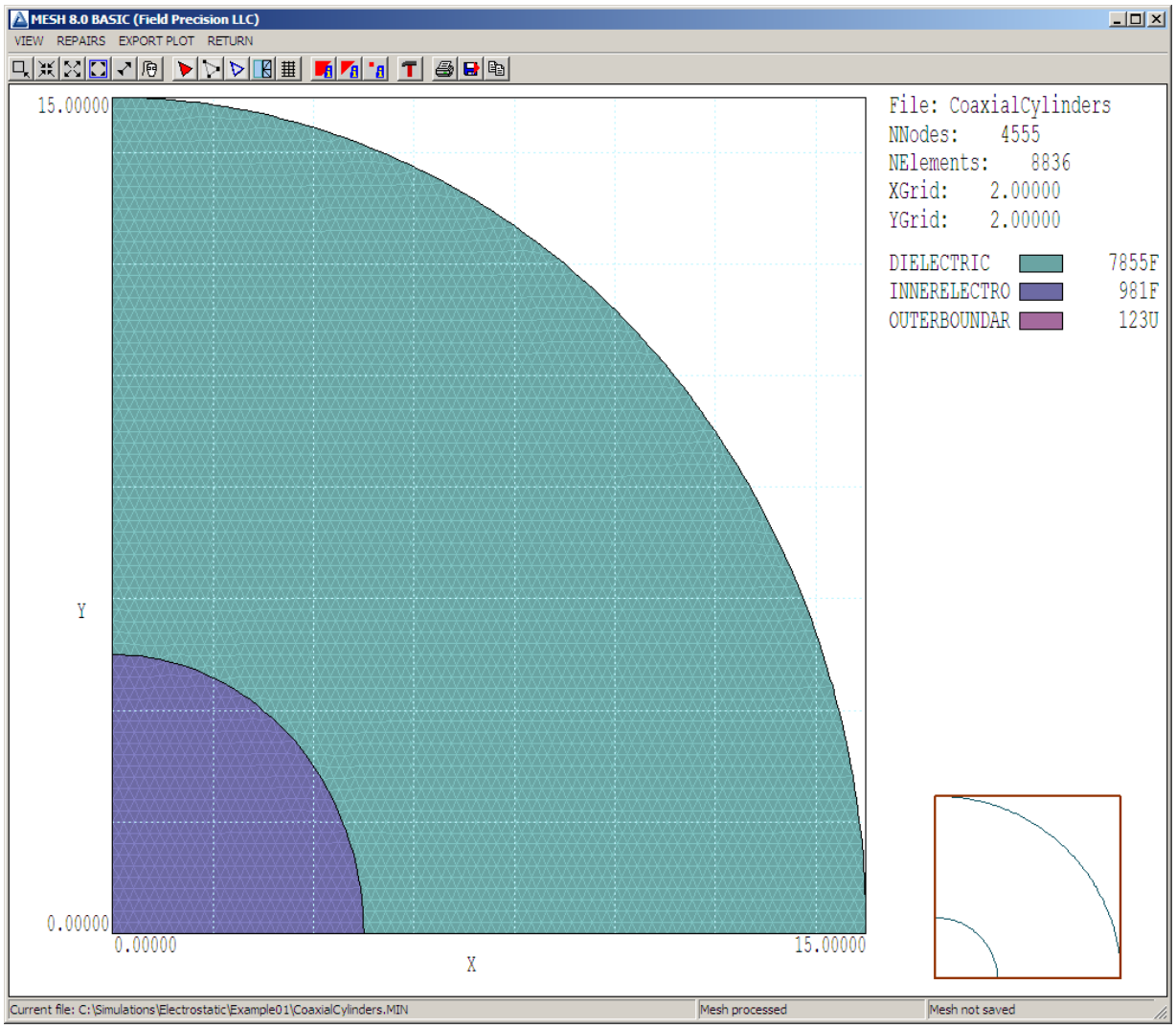


Figure 14: Completed mesh.

4 Electrostatic application: calculating and analyzing fields

In this chapter, we will use the mesh file we created to find a field solution. To get started, run **EStat** from the **FPController**. Click the *1* tool and choose the **Mesh** file `CoaxialCylinders.MOU`. The program determines the defined regions in the mesh and displays the dialog of Fig. 15. The values shown are appropriate to the solution parameters discussed in the previous chapter. Fill in the fields and click *OK*, accepting the output file name `CoaxialCylinders.EIN` (**E**Stat **IN**put).

To understand the action of the dialog, choose *File/Edit script (EIN)* and load the file in the editor. Here is the content:

```
* File: CoaxialCylinders.EIN
Mesh = CoaxialCylinders
Geometry = Rect
DUnit =   1.0000E+02
ResTarget =   5.0000E-08
MaxCycle =    5000
* Region 1: DIELECTRIC
Epsi(1) =   1.0000E+00
* Region 2: INNERELECTRODE
Potential(2) =   1.0000E+02
* Region 3: OUTERBOUNDARY
Potential(3) =   0.0000E+00
EndFile
```

You'll recognize many of the entries. The *ResTarget* and *MaxCycle* parameters that control the solution accuracy are described in the **EStat** manual. The default values are fine for this calculation.

Exit the editor and click the *2* tool. After you choose `CoaxialCylinders.EIN`, **EStat** generates and solves the finite-element equations in less than a second. Larger meshes will take longer, but generally the run times of practical solutions are less than a minute. The program creates the files `CoaxialCylinders.ELS` (a diagnostic listing file that you can inspect with an editor) and `CoaxialCylinders.EOU` (a record of the mesh coordinates and potential values at the nodes).

To see the results, press the *3* tool and choose `CoaxialCylinders.EOU`. You can generate interesting plots in the *Analysis* menu (Fig. 16). For this discussion, let's concentrate on hard numbers. First, let's check the absolute accuracy of the solution with a single point calculation. At radius $r = 10.0$ cm, the formulas listed in the previous article give the values:

$$\begin{aligned} E_r(r) &= 910.23923 \quad (\mathrm{V/m}), \\ \phi(r) &= 36.90703 \quad (\mathrm{V}). \end{aligned} \tag{2}$$

Numerical interpolations in **EStat** involve the collection of potential values from surrounding nodes. On symmetry boundaries, there are only half the available nodes, so the accuracy is not optimal. We should use an internal point for a good comparison. On a 45° line, the

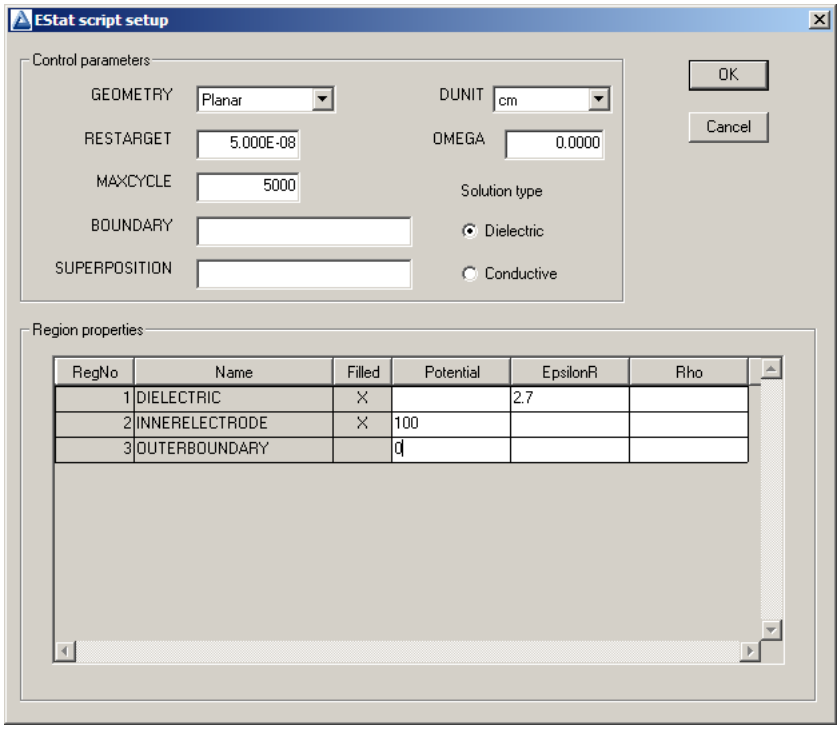


Figure 15: Dialog to create the **EStat** input script.

position $r = 10.0$ cm corresponds to $x = y = 7.071068$ cm. Click the *Point calculation* tool. We can specify the location by moving the mouse inside the solution volume and clicking the left button. This selection method is not accurate enough for the comparison. Instead, press the *F1* key after starting the *Point calculation* tool to enter coordinates manually. Fill in the x-y values and click *OK*. Values of several calculated quantities are displayed at the bottom. The calculated values of potential (36.90767 V) and electric field (910.23111 V/m) agree with the theoretical value to within thousands of a percent.

Rather than copy numbers from the screen, why not let **EStat** write them for us? Click the *Open data record* command and accept the default name `CoaxialCylinders.DAT`. Now, every time we do an interactive calculation, the results are recorded in the text file. For example, choose the menu command *Analysis/Volume integrals*. The results are displayed on the screen and written in the file `CoaxialCylinders.DAT`. You can inspect the file with an external editor. To use the internal **EStat** editor, click the *Close data record* command first because two instances of a same file cannot be opened simultanously in a program. Open the data file to see the result of the calculation, the volume integrals of the field-energy-density:

```
Quantity: Energy
  Global integral:   1.708936E-07
   RegNo   Integral
   ==========================
       1    1.708936E-07
       2    9.881184E-37
```

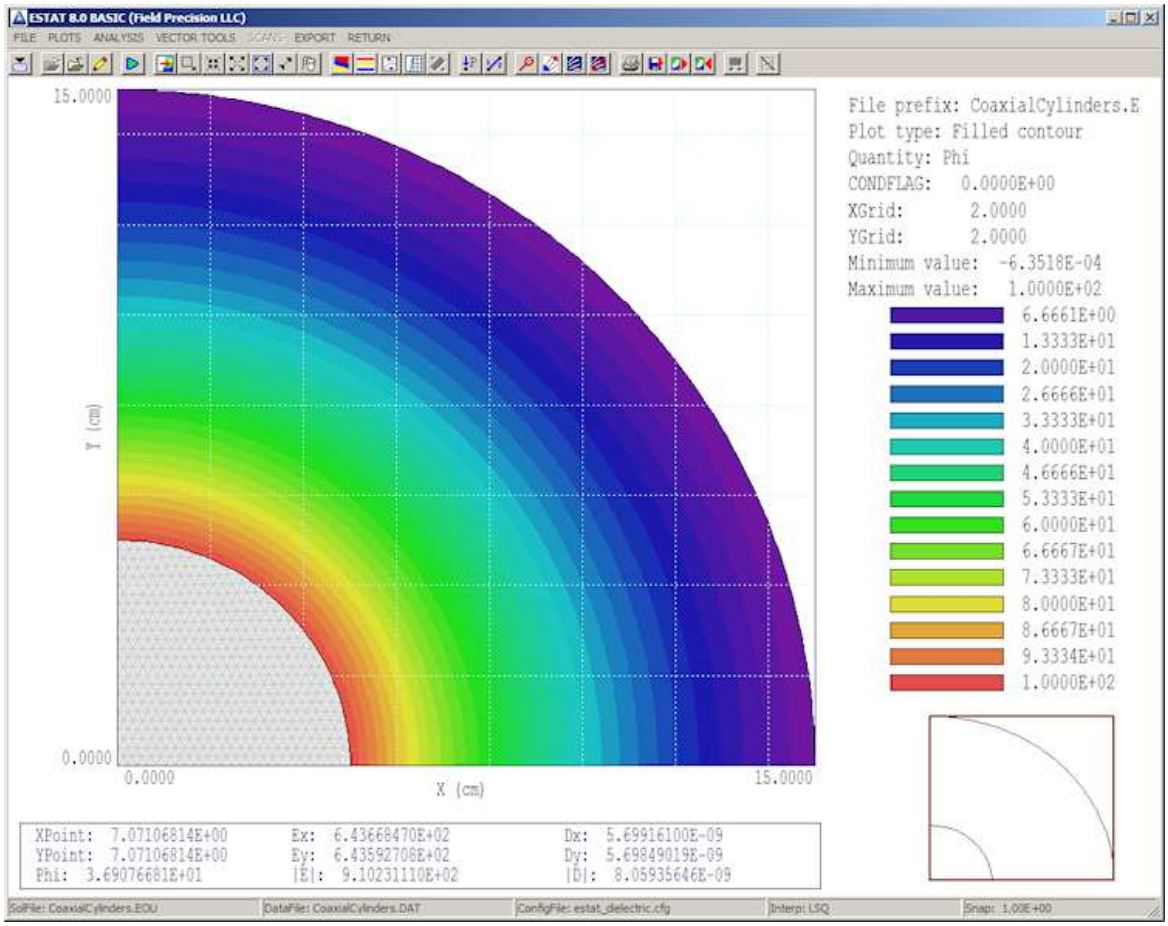


Figure 16: Filled-contour plot of potential with results of a point calculation displayed.

The field inside the inner electrode (*Region 2*) is numerically zero. We can use the field energy in the dielectric (*Region 1*) to find the capacitance per length. Because the calculation covers only the first quadrant, we need to multiply by a factor of four to find the field energy per length of coaxial cylinders at 100 V, $U_e = 6.835744 \times 10^{-7}$ J/m. The equation $c = (2U_e)/100^2$ gives $c = 1.367149 \times 10^{-10}$ F/m, within 0.8% of the theoretical value listed in the previous chapter.

To understand the effects of element size, we want to compare accuracy at several radii for different choices of element dimensions. The procedure would be tedious if we had to run the *Point calculation* tool and type coordinates for each datum. In the next chapter, we'll automate the analysis procedure.

5 Electrostatic application: meshing and accuracy

In this chapter we'll continue with the solution discussed in the previous chapters, making several calculations to understand the effect of element size on solution accuracy. To expand our toolbox of techniques, we'll automate steps in the analysis procedure. The best way to start a project is with a clear statement of the goal. In this case, we want to check the accuracy of numerical estimates of the electric field $E_r(r)$ at locations in the dielectric as a function of element size. Specifically, we'll check field interpolations near the inner electrode ($r = 5.0$ cm), near the outer electrode ($r = 15.0$ cm) and at the center of the gap ($r = 10.0$ cm) for the following choices of approximate element size: 0.1 cm, 0.2 cm, 0.3 cm, 0.4 cm and 0.5 cm.

As mentioned in previous chapters, good supporting software (text editors, file managers, image editors,...) is essential for effective technical work. In this case, we'll use a spreadsheet created with **OpenOffice Calc**. Figure 17 shows the upper section of the sheet. Parameters at the top include the inner and outer radii, the applied voltage and the theoretical value for the capacitance per length in z. The section beneath lists the radial positions for the calculations. Note that the inner radius is slightly larger than r_i and the outer radius smaller than r_o. Remember that the electrode boundaries are a set of straight lines that approximate sections of circles. We have to include a little slack to make sure that the test points lie within the dielectric region for all choices of element size. We may as well let the spreadsheet do the mathematical work. The cells for the x-y coordinates of the measurement points on a 45° line contain the formula $r/\sqrt{2}$. Similarly, the cells for the theoretical values contain the formulas for $E_r(r)$ and $\phi(r)$ (Eq. 1).

The section below the theoretical values illustrates another use of spreadsheets. Why not let the program prepare the input scripts? In this case, we'll generate an analysis script – a set of instructions to **EStat** for standard calculations to carry out for each of the solutions:

```
* Coaxial cylinders analysis script
INPUT COAXIALCYLINDERS.EOU
OUTPUT COAXIALCYLINDERS.DAT
POINT 3.53624 3.53624
POINT 7.07107 7.07107
POINT 10.60589 10.60589
VOLUMEINT 1
ENDFILE
```

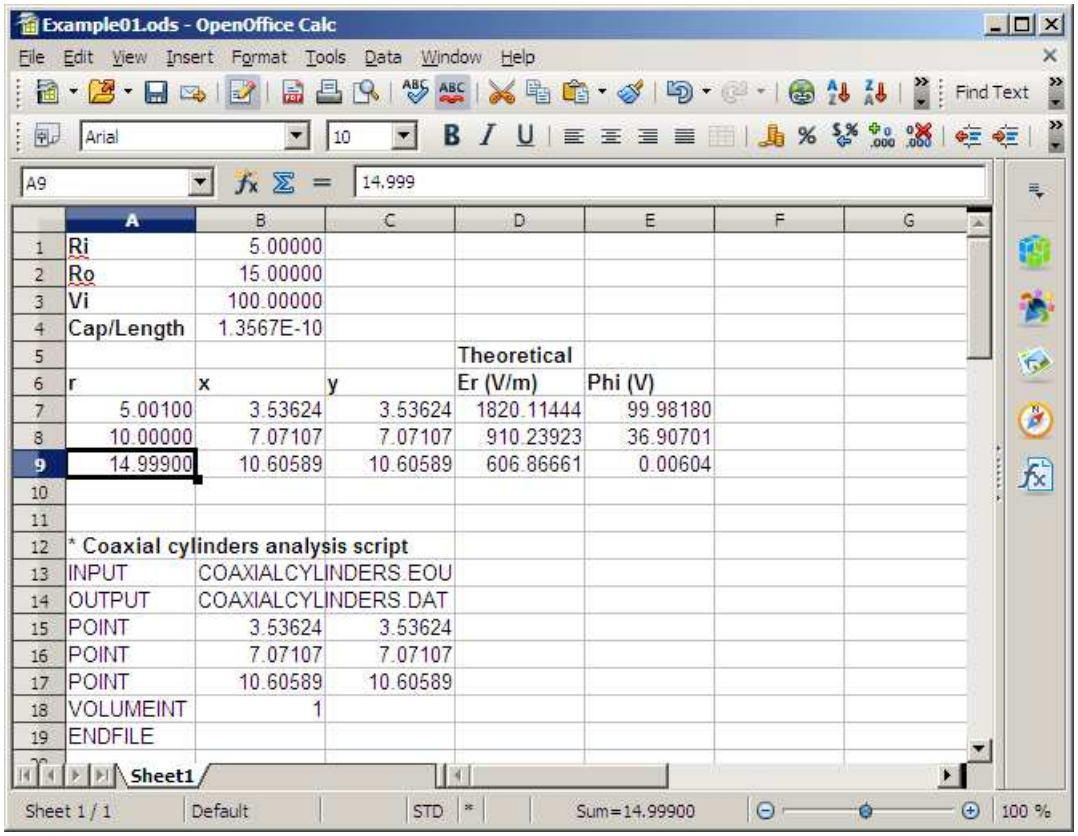
The spreadsheet has supplied the coordinates. To use the information, simply copy it and paste it into a text file `CoaxialCylinders.SCR` (**SCR**ipt). The commands have the following actions:

Open the **EStat** output file `CoaxialCylinders.EOU`.

Write the results of calculations to `CoaxialCylinders.DAT`.

Perform three point interpolations at the desired locations.

Calculate volume-integral quantities over *Region 1* to find the capacitance per length.



	A	B	C	D	E	F	G
1	Ri	5.00000					
2	Ro	15.00000					
3	Vi	100.00000					
4	Cap/Length	1.3567E-10					
5				Theoretical			
6	r	x	y	Er (V/m)	Phi (V)		
7	5.00100	3.53624	3.53624	1820.11444	99.98180		
8	10.00000	7.07107	7.07107	910.23923	36.90701		
9	14.99900	10.60589	10.60589	606.86661	0.00604		
10							
11							
12	* Coaxial cylinders analysis script						
13	INPUT	COAXIALCYLINDERS.EOU					
14	OUTPUT	COAXIALCYLINDERS.DAT					
15	POINT	3.53624	3.53624				
16	POINT	7.07107	7.07107				
17	POINT	10.60589	10.60589				
18	VOLUMEINT	1					
19	ENDFILE						

Figure 17: Top section of a **Calc** spreadsheet: solution setup

We can quickly change the element size by editing the **Mesh** input script. Figure 18 shows the file displayed in the **Context** editor. The red arrows show the quantities to be modified. Given a modified **Mesh** input script, there are three steps to regenerate the solution and set new values in the data file `CoaxialCylinders.DAT`:

Run **Mesh**, load `CoaxialCylinders.MIN`, process the mesh and save the result.

Run **EStat**, load `CoaxialCylinders.EIN` and run the solution.

Run the analysis script `CoaxialCylinders.SCR`.

With only five solutions, the procedure wouldn't take long. On the other hand, it takes only a few seconds to set up a Windows batch file that handles everything automatically.

In **FPController** (`fpcontroller.exe`), press the *Create task* button to bring up the dialog of Fig. 19. Fill in the file prefix *CoaxialCylinders* and activate the *Beep at completion* box. We need to define the three tasks of a solution. Clicking a cell in the first column raises a popup menu (inset) that shows your installed Field Precision programs as well as useful batch commands. For the first task, choose **Mesh**. Then, click in the second column and choose the *Select file* button. The program displays a standard dialog to select appropriate input files with suffix `MIN`. Pick `CoaxialCylinders.MIN` to fill in the first row. Similarly, define two more tasks for an *EStat* solution and an analysis. When the display looks like Fig. 19, press *OK*. The program creates the file `CoaxialCylinders.BAT` in the working directory. Here is the content:

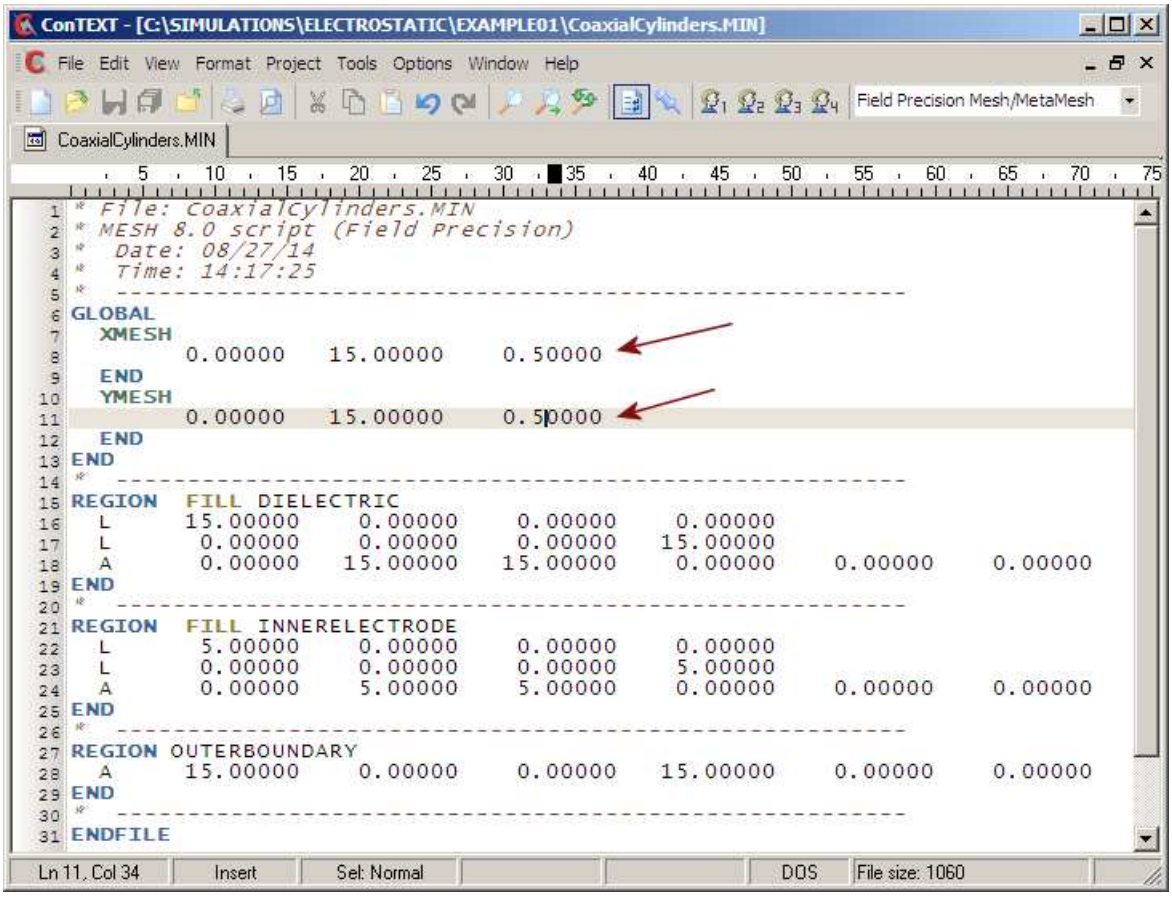


Figure 18: File `CoaxialCylinders.MIN` displayed in **Context**.

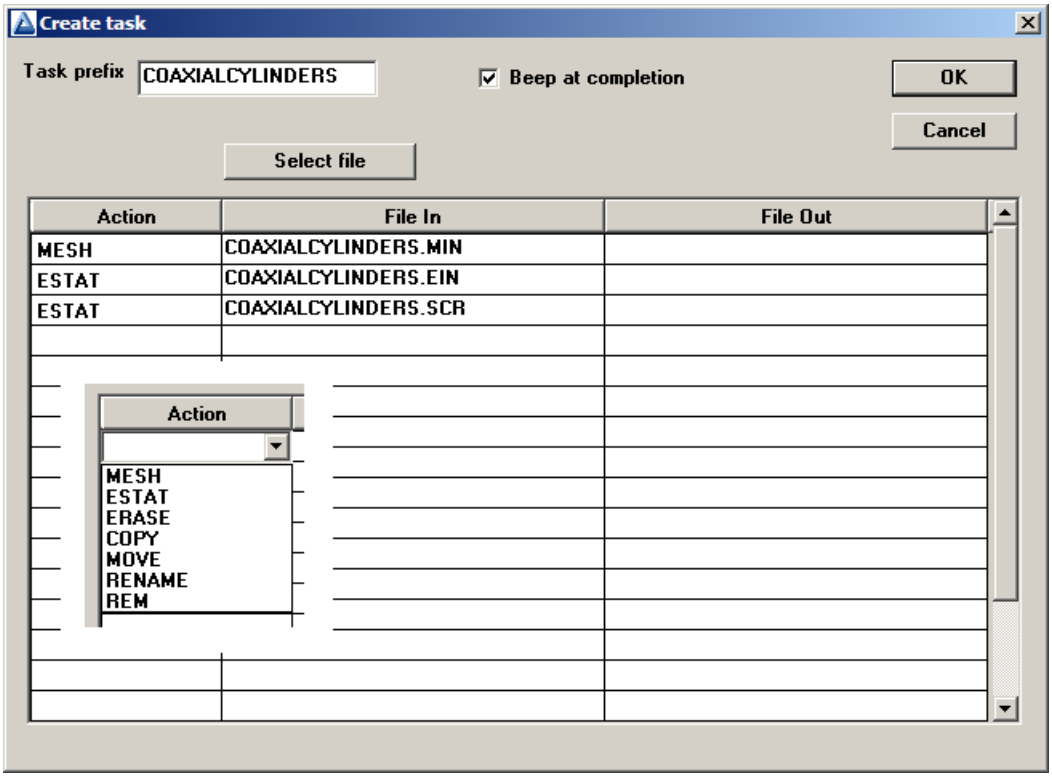


Figure 19: Create task dialog in `tc.exe`.

```
REM TriComp batch file, Field Precision
START /B /WAIT C:\fieldp_basic\tricomp\mesh.exe COAXIALCYLINDERS
START /B /WAIT C:\fieldp_basic\tricomp\estat.exe COAXIALCYLINDERS.EIN
START /B /WAIT C:\fieldp_basic\tricomp\estat.exe COAXIALCYLINDERS.SCR
START /B /WAIT C:\fieldp_basic\tricomp\NOTIFY.EXE
IF EXIST COAXIALCYLINDERS.ACTIVE ERASE COAXIALCYLINDERS.ACTIVE
```

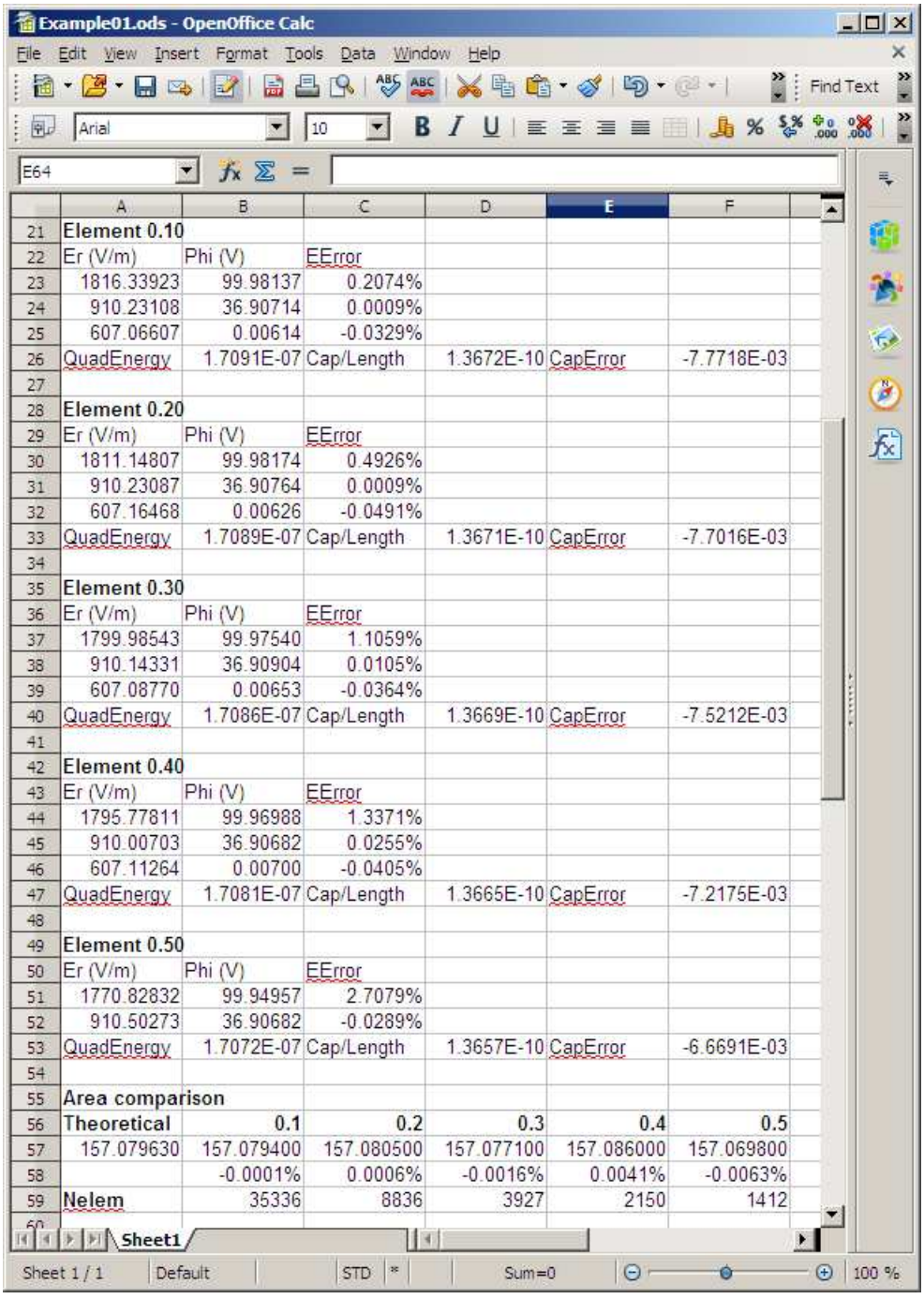
Note that most Field Precision programs can run from the command line with the input file as a heading parameter. The `/WAIT` option ensures that a program does not start before the previous program completes it action. Under batch file control, a full solution consists of the following user operations:

Edit and save `CoaxialCylinders.MIN` with the desired element size.

Press `Run task` in **FPController** and choose `CoaxialCylinders.BAT`.

Edit the resulting file `CoaxialCylinders.DAT` and extract values of interest.

I performed the solutions and copied and pasted values to create the sections of the spreadsheet shown in Fig. 20. I also set up formulas in the spreadsheet to determine the relative errors in electric-field values. For a visual reference, Fig. 21 shows the mesh for element sizes of 0.2 and 0.5 cm. We can now consider some implications of the results:



	A	B	C	D	E	F
21	**Element 0.10**					
22	Er (V/m)	Phi (V)	EError			
23	1816.33923	99.98137	0.2074%			
24	910.23108	36.90714	0.0009%			
25	607.06607	0.00614	-0.0329%			
26	QuadEnergy	1.7091E-07	Cap/Length	1.3672E-10	CapError	-7.7718E-03
27						
28	**Element 0.20**					
29	Er (V/m)	Phi (V)	EError			
30	1811.14807	99.98174	0.4926%			
31	910.23087	36.90764	0.0009%			
32	607.16468	0.00626	-0.0491%			
33	QuadEnergy	1.7089E-07	Cap/Length	1.3671E-10	CapError	-7.7016E-03
34						
35	**Element 0.30**					
36	Er (V/m)	Phi (V)	EError			
37	1799.98543	99.97540	1.1059%			
38	910.14331	36.90904	0.0105%			
39	607.08770	0.00653	-0.0364%			
40	QuadEnergy	1.7086E-07	Cap/Length	1.3669E-10	CapError	-7.5212E-03
41						
42	**Element 0.40**					
43	Er (V/m)	Phi (V)	EError			
44	1795.77811	99.96988	1.3371%			
45	910.00703	36.90682	0.0255%			
46	607.11264	0.00700	-0.0405%			
47	QuadEnergy	1.7081E-07	Cap/Length	1.3665E-10	CapError	-7.2175E-03
48						
49	**Element 0.50**					
50	Er (V/m)	Phi (V)	EError			
51	1770.82832	99.94957	2.7079%			
52	910.50273	36.90682	-0.0289%			
53	QuadEnergy	1.7072E-07	Cap/Length	1.3657E-10	CapError	-6.6691E-03
54						
55	**Area comparison**					
56	**Theoretical**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
57	157.079630	157.079400	157.080500	157.077100	157.086000	157.069800
58		-0.0001%	0.0006%	-0.0016%	0.0041%	-0.0063%
59	**Nelem**	35336	8836	3927	2150	1412

Figure 20: Bottom section of a **Calc** spreadsheet: solution analysis.

For an element size of 0.1 cm, the mesh had 35336 elements. The mesh with size 0.5 cm had only 1412 elements. Therefore, the solution with the fine mesh required about 25 times as much computational work.

The error in the electric field calculation at the center of the gap ($r = 10.0$ cm) was infinitesimal for element size 0.1 cm (0.0009%) and acceptable for many applications at 0.5 cm (0.0289%).

The error in electrical field near the electrodes was higher, as we would expect when representing a curved surface with a set of line segments. The relative error was higher on the inner electrode because there were fewer segments. For the 0.5 cm elements, the error of 2.7% on the inner electrode could be of concern for some applications.

Beginning users often employ far more elements than is necessary. For the determination of electric field values in the dielectric volume (removed from electrodes), the accuracy with the fine mesh of Fig. 21 may not justify the extra computational work. If you have concerns about mesh resolution, the standard procedure is to do two calculations with different mesh sizes and to check whether values in critical regions differ by more than the accuracy requirement. Note that there are two techniques to increase accuracy while minimizing the number of element:

Variable mesh resolution.

The *Boundary* method to create a microscopic solution within a macroscopic solution.

Variable resolution is discussed in Chap. 7. The boundary method, an advanced technique to resolve small features in a large solution space, is covered in the instruction manuals for **EStat**, **PerMag**, **HiPhi** and **Magnum**.

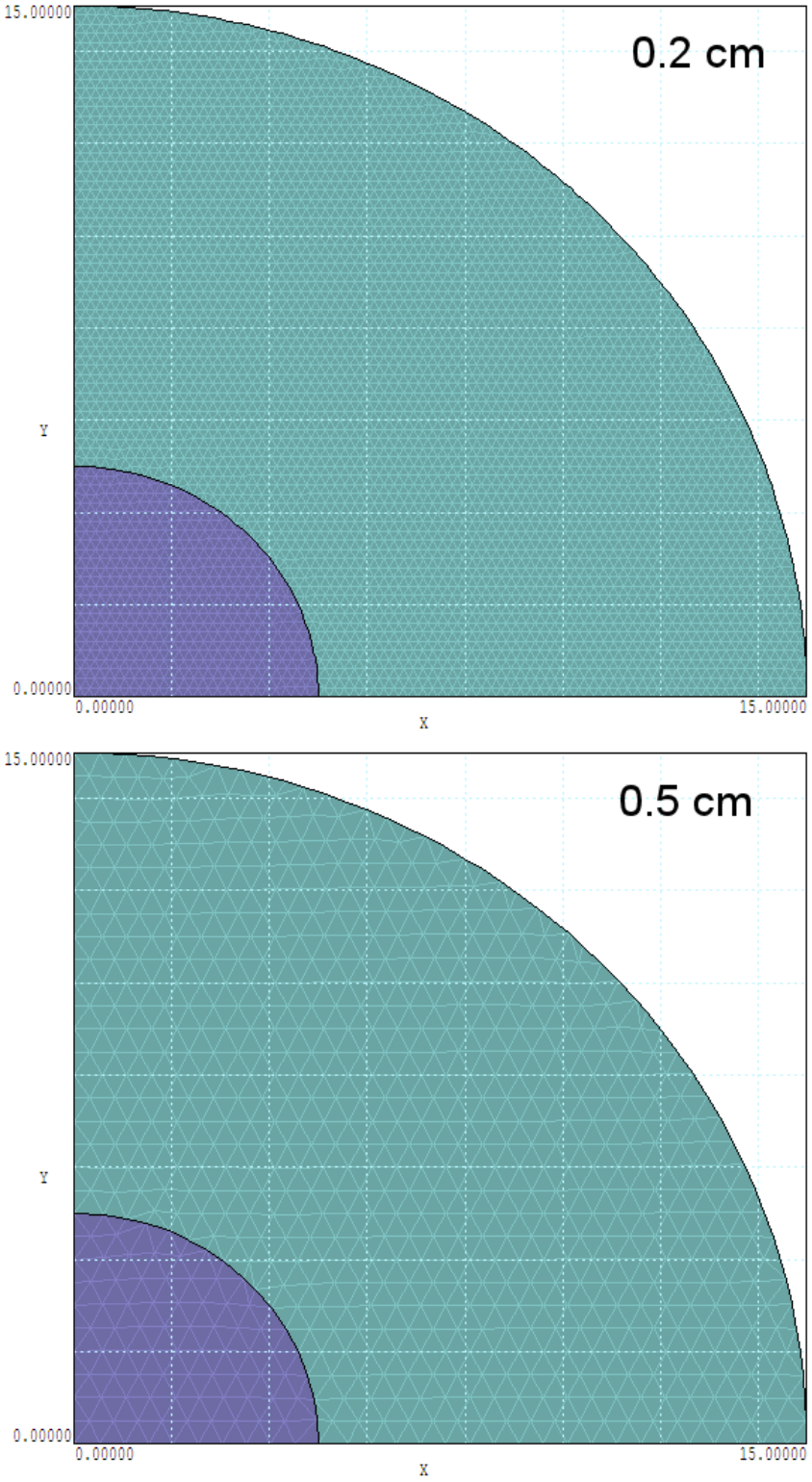


Figure 21: Mesh geometries for small and large elements.

We've covered a lot of ground in the last three chapters following a simple electrostatic solution:

Organizing simulation data and using a file manager.

Defining region boundaries with the **Mesh** *Drawing Editor*.

Employing symmetry boundaries to reduce the size of a calculation.

Using a spreadsheet to plan and to analyze calculations.

Understanding the set of input and output files for an electrostatic solution.

Creating program input scripts in interactive dialogs and modifying them with a text editor.

Generating and inspecting standard mesh definition files with **Mesh**.

Defining run-control and material parameters for **EStat** solutions.

Running **Mesh** and **EStat** interactively or from the Windows command prompt.

Automatically controlling **EStat** with analysis scripts.

Creating batch files for automatic run control with the task generator of **FPController**.

Gaining insights into the relationship between element size and interpolation accuracy.

In the next chapter, we'll turn our attention to 2D magnet-field calculations.

6 Magnetostatic solution: simple coil with boundaries

In this chapter, we'll advance to 2D magnetostatic solutions using the programs **Mesh** and **PerMag**. The previous chapters on electrostatics covered the basic concepts of finite-element calculations and the operation sequences of the programs. Therefore, in this chapter and following ones, we'll concentrate on the special features of magnetic field calculations.

To review, finite-element solutions of the the previous chapters determined the electrostatic potential ϕ (a scalar quantity) at the node points of the mesh that we created. Components of the electric field vector at a location could then be determined by collecting local values of ϕ and taking numerical derivatives:

$$\mathbf{E} = -\nabla\phi. \tag{3}$$

The components are E_x and E_y for planar solutions and E_z and E_r for cylindrical solutions.

Things get more involved for magnetic field calculations. The calculated node quantity is the vector potential $\mathbf{A}$. The magnetic flux density $\mathbf{B}$ is given by

$$\mathbf{B} = \nabla \times \mathbf{A}. \tag{4}$$

Fortunately, there is only a single component of $\mathbf{A}$ in 2D calculations. The node quantity is A_z for planar solutions (with flux density components B_x and B_y) and rA_θ for cylindrical solutions (with components B_z and B_r). The vector potential has useful properties for making plots:

> In planar solutions, contours of A_z separated by a uniform interval ΔA_z lie along lines of $\mathbf{B}$ separated by equal intervals of magnetic flux per length.
>
> In cylindrical solutions, contours of rA_θ separated by a uniform intervals $\Delta(rA_\theta)$ lie along lines of $\mathbf{B}$ separated by equal intervals of magnetic flux.

Let's get to work and generate some magnetic fields. We'll start with a cylindrical coil in free space. Because the geometry is simple, we'll write the boundary specifications directly. Run **Mesh** and click the *New mesh (text)* tool to bring up the dialog of Fig. 22. The values define a solution volume that covers the range[2] -10.0 cm $\le z \le$ 10.0 cm, 0.0 cm $\le r \le$ 15.0 cm. When you click `OK`, the program opens the internal text editor with the default content shown in Fig. 23. The *Global* section at the top sets the *foundation mesh* (the set of elements before conformal shifting of boundary nodes). It covers the range we specified with a default element size of 0.1 cm. Advanced commands (like *TriType*) are listed as comment lines. We won't need them for this calculation, so delete the comments.

A default region named *SolVolume* (*Region 1*) covers the solution volume, appropriate for this calculation[3]. **Mesh** has also started a default second region. We'll use it to define the rectangular cross section of a cylindrical coil. The comment lines show the entries that could appear within a region section. Erase the comments and rename the region *Coil*. The coil has dimensions -3.0 cm $\le z \le$ 3.0 cm and 4.0 cm $\le r \le$ 6.0 cm. Copy and paste the line vectors of

[2] **Mesh** displays an error message if you try to open a cylindrical solution with $r < 0.0$.

[3] Although *Region 1* often covers the full solution, it is not a necessary condition as we saw in the previous electrostatic solution.

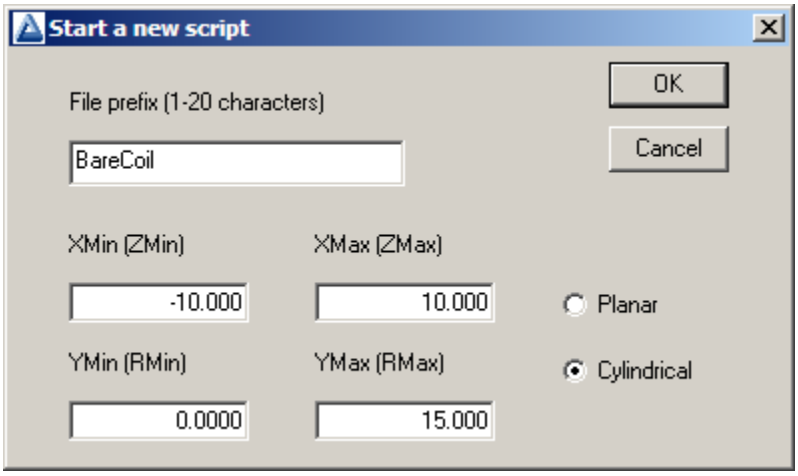


Figure 22: Dialog to start a **Mesh** script in the text mode.

Region 1 and use *Search/Replace* to modify the dimensions. We'll also add a third region by copying and pasting *Region 1* verbatim. Rename it *Boundary* and remove the word *Fill* in the *Region* command. We'll discuss the purpose of this extra region later. For now, the modified script should look like Table 1. Save your work and exit the editor.

Click the *Load script (MIN)* tool to read the boundary specifications and then click *Process*. When complete, click *Save mesh (MOU)* to create the file `BareCoil.MOU` for use by **PerMag**. You can go to the *Plot/Repair menu* tool to view a cross-section of the resulting mesh, consisting of 60,000 elements.

Run **PerMag** and click the *1* tool to bring up the dialog of Fig. 24. As in the previous calculation, this dialog creates a script of run and material parameters to control the solution program. We can ignore several of the fields because the values control advanced runs with non-linear materials (*e.g.*, saturable iron). For this run, we need only set the *Geometry* to *Cylindrical* and fill in the magnetic properties of the regions. Air has relative permeability $\mu_r = 1.0$. The coil also has $\mu_r = 1.0$. In addition, we need to set a total current that flows in the θ direction for a cylindrical solution. If the coil has $N = 1000$ turns and the drive current is $I = 2.5$ A, then the total A-turns over the cross section is 2500.0. Equivalently, the region carries a uniform azimuthal current density[4]. Finally, we want to see what happens if we make no special provisions for the *Boundary* region, so don't set any property. Click *OK* and save the data in the file `BareCoil.PIN`. You can check the file content with the internal **PerMag** editor.

Click the *2* button and choose `BareCoil.PIN` to create a finite-element solution. Then click *3* and load `BareCoil.POU` to analyze the solution. To begin, let's check the lines of **B**. Click the *Plot type* tool and set the option *Contour lines*. Then click the *Plot quantity* tool and choose `rATheta` to produce the plot of Fig. 25*a*. There are two interesting features:

[4]Regions in **PerMag** always have uniform average current density. To represent a coil with variations of winding density, divide it into multiple regions.

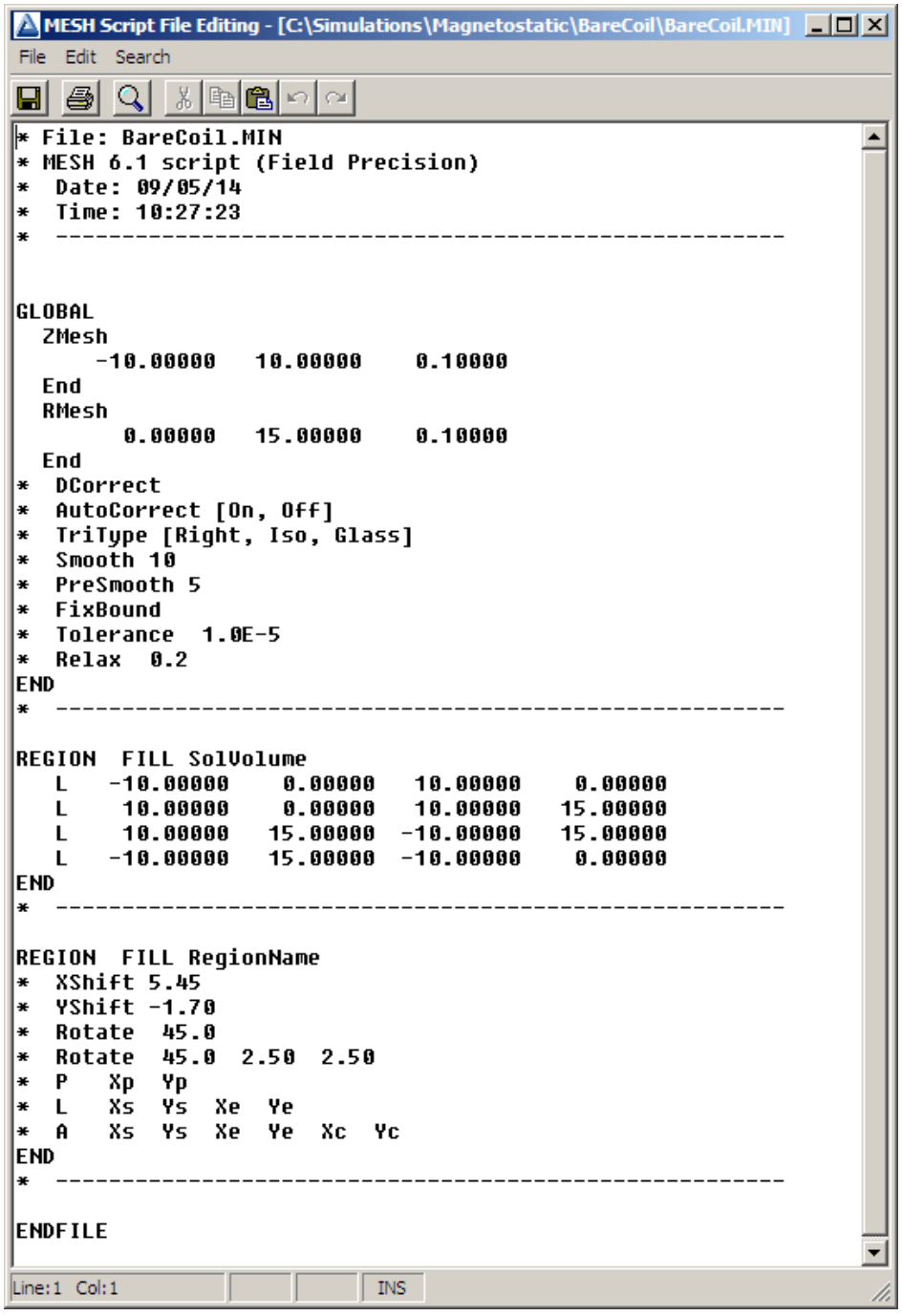


Figure 23: Starting a **Mesh** script in text mode, display of the internal editor.

Table 1: **Mesh** script `BareCoil.MIN`.

```
GLOBAL
  XMesh
      -10.00000   10.00000    0.10000
  End
  YMesh
        0.00000   15.00000    0.10000
  End
END
*  ------------------------------------------------------
REGION  FILL Air
   L   -10.00000    0.00000   10.00000    0.00000
   L    10.00000    0.00000   10.00000   15.00000
   L    10.00000   15.00000  -10.00000   15.00000
   L   -10.00000   15.00000  -10.00000    0.00000
END
*  ------------------------------------------------------
REGION  FILL Coil
   L    -3.00000    4.00000    3.00000    4.00000
   L     3.00000    4.00000    3.00000    6.00000
   L     3.00000    6.00000   -3.00000    6.00000
   L    -3.00000    6.00000   -3.00000    4.00000
END
*  ------------------------------------------------------
REGION  Boundary
   L   -10.00000    0.00000   10.00000    0.00000
   L    10.00000    0.00000   10.00000   15.00000
   L    10.00000   15.00000  -10.00000   15.00000
   L   -10.00000   15.00000  -10.00000    0.00000
END
*  ------------------------------------------------------
ENDFILE
```

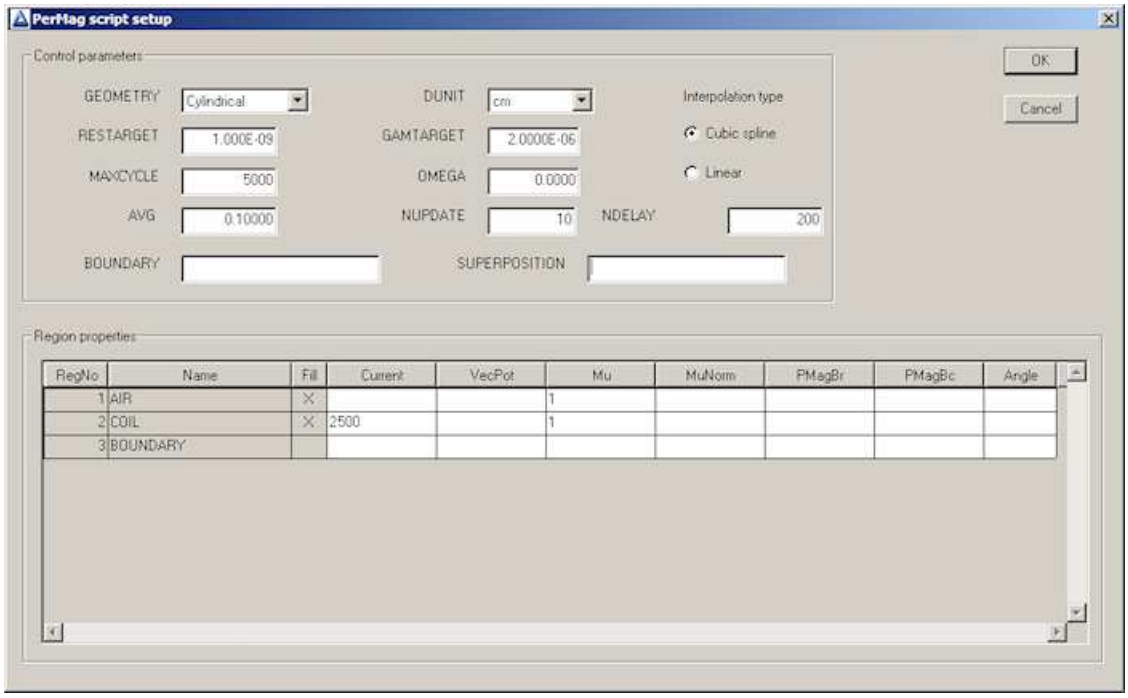


Figure 24: **PerMag** dialog to define run parameters and material properties.

Even though the field inside the coil is fairly uniform, the space between lines is larger near the axis. This is because the lines are separated by intervals of equal magnetic flux and the area normal to z gets smaller as r approaches zero. In other words, the density of lines is not proportional to $|\mathbf{B}|$ in cylindrical coordinates.

The lines of $\mathbf{B}$ are normal to the boundaries, the standard default Neumann condition for a finite-element magnetic solution (*i.e.*, the derivative of vector potential normal to the boundary is zero). With regard to the axial boundaries, the condition would occur if there were an infinite array of coils with alternating positive and negative currently. Certainly, this is not what we intended. Furthermore, the boundary on the outer radius is non-physical in the sense that we could find no set of real coils that would generate the field pattern.

With regard to the second point, we must clearly do some thinking about the boundary condition. The alternative is to set the vector potential equal to a constant on the boundary. The Dirichlet condition is $rA_\theta = 0.0$. In this case, the component of $\mathbf{B}$ normal to boundary is zero, so that lines of $\mathbf{B}$ are parallel. To do this, we change the specification of Region 3 to:

```
* Region 3: BOUNDARY
VecPot(3) = 0.0000E+00
```

The command states that the node quantity assumes a fixed value.

Figure 25*b* shows the modified solution. It is better in one respect. The field pattern could be achieved in an actual physical system: a coil inside a superconducting can. At this point, you may object that you wanted a coil in infinite space, not in an enclosure. This illustrates a limitation of the finite-element method. Calculations must be performed in a finite volume.

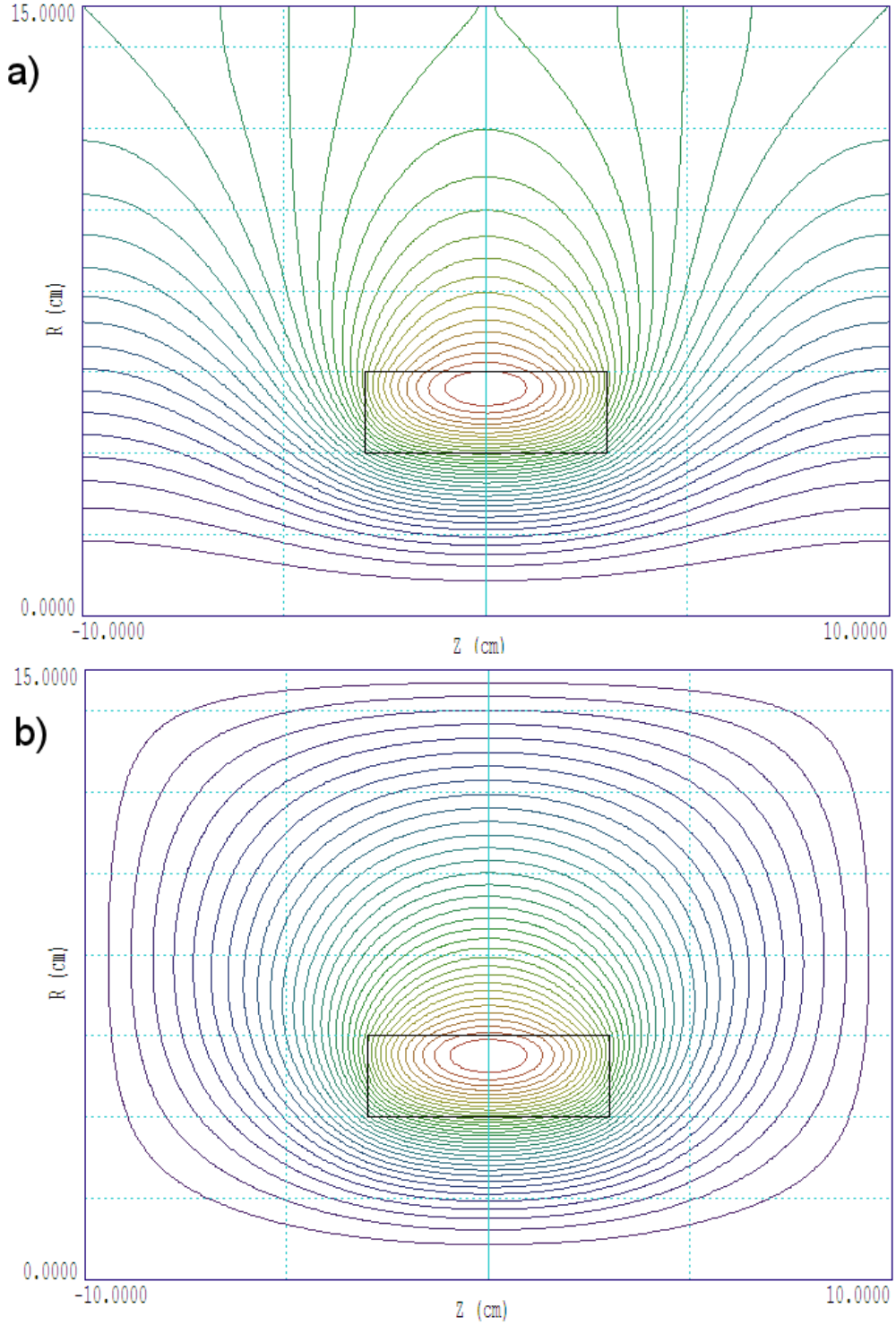


Figure 25: Lines of **B** for the solution. *a*) Default Neumann boundary condition. *b*) Dirichlet condition, fixed vector potential.

There will be boundaries, and conditions on the boundaries must be specified. The common conditions are Neumann or Dirichlet.

But, is it a limitation? Sometimes, issues in a numerical solution mirror issues in the physical world. When is the last time you operated a magnet in infinite space? In a real system, there is usually an assortment of nearby objects in unpredictable locations. These objects would strongly alter the far fields and may affect field components inside the working volume of the magnet. In this sense, the magnet of Fig. 25, with its strong fringing fields, is a poor design. Another drawback is that significant fraction of magnetic field energy is outside the working volume (the interior of the coil). The wasted energy is reflected in higher power to drive the coil. In Chap. 8, we will consider the use of ferromagnetic methods to improve the design. The next chapter describes a numerical study to quantify how much the boundaries in a finite-element magnetic-field solution affect the results.

7 Magnetostatic solution: boundary effects and automatic operation

The previous chapter emphasized that finite-element calculations are performed in a finite volume and that conditions on the the boundaries must be specified. The Neumann condition (field lines normal to the boundary) is usually used along a symmetry plane. An example is one half of a magnetic mirror split at the midplane. Otherwise, the most common boundary is a Dirichlet condition (*i.e.*, fixed vector potential), equilvalent to a perfectly-conducting wall. Although the boundary affects the fields of coils in infinite space, this does not represent a practical limitation on the finite-element method because you would normally seek to design a magnet to limit the extent of fringing fields.

This chapter has has three learning goals:

Quantify the effect of the Dirichlet boundaries in magnetostatics.

Introduce the use of a variable-resolution meshes.

Set up an automatic calculation to do a parameter search.

We'll continue with the cylindrical coil from the previous chapter. It carries a total current of 2500 A-turn uniformly distributed over the cross section, -3.0 cm $\leq z \leq$ 3.0 cm, 2.0 cm $\leq r \leq$ 4.0 cm. We'll start with close boundaries (-5.0 cm $\leq z \leq$ 5.0 cm, 0.0 cm $\leq r \leq$ 7.5 cm) and then expand them to see how the fields approach the infinite-space result. The element size should be relatively small near the coil, but we can use larger elements in the expanding surrounding volume to minimize computational work.

For the smallest solution, the foundation mesh definitions in the **Mesh** input script look like this:

```
GLOBAL
  ZMESH
   -5.0 -3.5  0.2
   -3.5  3.5  0.1
    3.5  5.0  0.2
  END
  RMESH
    0.0  4.5  0.1
    4.5  7.5  0.2
  END
END
```

The axial specification states that the initial triangular element base (before smoothing and fitting) is 0.2 cm in the zones -5.0 cm $\leq z \leq$ -3.5 cm and 3.5 cm $\leq z \leq$ 5.0 and 0.1 cm near the coil. Figure 26 shows the result. Note that **Mesh** has fitted the coil boundaries exactly and made smooth transitions between regions of different element sizes.

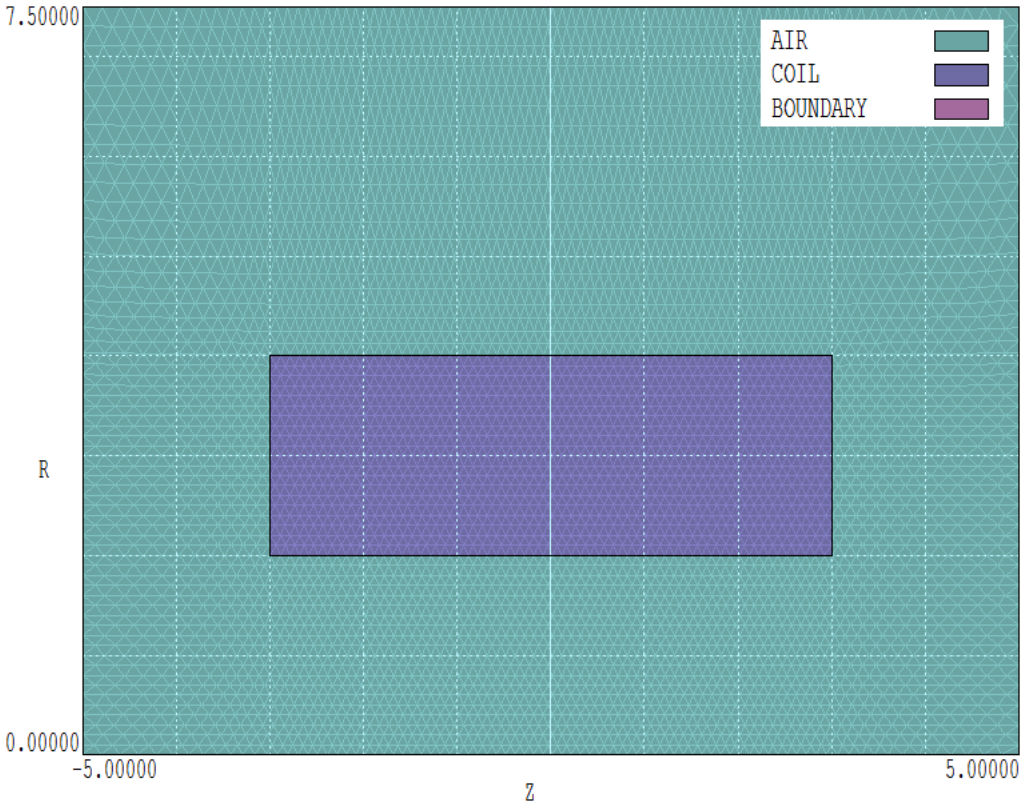


Figure 26: Variable resolution mesh for the magnet coil solution.

To make useful comparisons of the numerical results, we need a baseline. The theoretical expression[5] for the on-axis field at the midplane of a solenoid ($z = 0.0$ cm, $r = 0.0$ cm) with finite length and radial thickness is:

$$\mathbf{B} = \frac{\mu_0 N i}{2(r_2 - r_1)} \ln \left[\frac{\sqrt{r_2^2 + (l/2)^2} + r_2}{\sqrt{r_1^2 + (l/2)^2} + r_1} \right]. \tag{5}$$

For the values $N_i = 2500.0$, $r_1 = 0.02$ m, $r_2 = 0.04$ m and $l = 0.06$ m, Eq. 5 yields the value $B_z(0,0) = 0.037186$ tesla.

For the study, we will expand the boundaries (keeping $r_{max} = 1.5 z_{max}$) and compare the value of $B_z(0,0)$ to the infinite-space result. We will use the following nine values for z_{max}: 5.0 cm, 6.0 cm, 7.0 cm, 8.0 cm, 9.0 cm, 10.0 cm, 12.50 cm, 15.0 cm and 20.0 cm. We could do each calculation interactively: create nine mesh scripts, run and analyze nine **PerMag** solutions. That's the hard way. Field Precision programs offer a useful option for extended calculations. Batch files and external programs (*e.g.* python scripts) can not only run the technical programs, but they can also control how the program interprets variable quantities in the input script. In the present application, the **Mesh** input script is modified to the following form:

[5] `http://www.netdenizen.com/emagnet/solenoids/solenoidonaxis.htm`.

```
*  ------------------------------------------------------
GLOBAL
 ZMESH
   %1  -3.5  0.2
  -3.5  3.5  0.1
   3.5  %2   0.2
 END
 RMESH
   0.0  4.5  0.1
   4.5  %3   0.2
 END
END
*  ------------------------------------------------------
REGION  FILL AIR
 L     %1    0.00000    %2    0.00000
 L     %2    0.00000    %2    %3
 L     %2    %3         %1    %3
 L     %1    %3         %1    0.00000
END
*  ------------------------------------------------------
REGION  FILL COIL
 L     -3.00000    2.00000    3.00000    2.00000
 L      3.00000    2.00000    3.00000    4.00000
 L      3.00000    4.00000   -3.00000    4.00000
 L     -3.00000    4.00000   -3.00000    2.00000
END
*  ------------------------------------------------------
REGION BOUNDARY
 L     %1    0.00000    %2    0.00000
 L     %2    0.00000    %2    %3
 L     %2    %3         %1    %3
 L     %1    %3         %1    0.00000
END
*  ------------------------------------------------------
ENDFILE
```

Note the symbolic representation of the boundary limits, a convention familiar to users of Windows batch files. The symbol %1 stands for z_{min}, %2 for z_{max} and %3 for r_{max}.

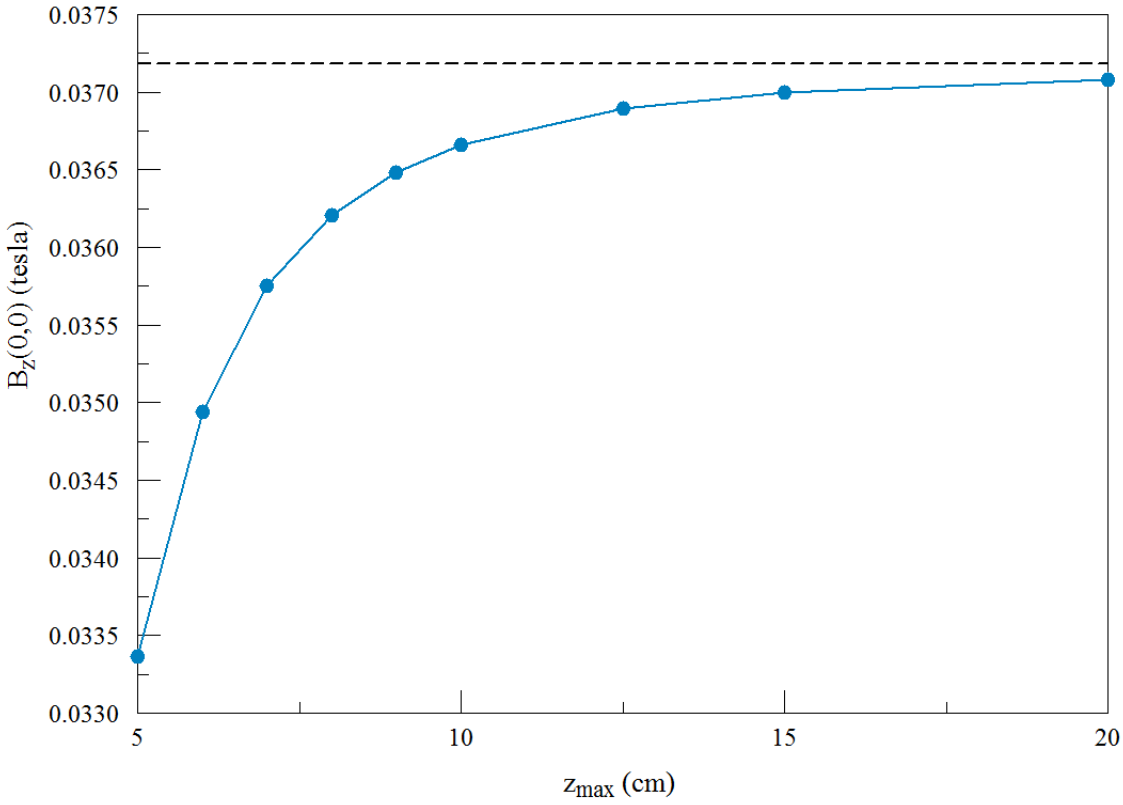


Figure 27: Results of the solution set. Dashed line shows the theoretical free-space value.

We prepare a Windows batch file with the following content:

```
START /B /WAIT C:\fieldp_pro\tricomp\mesh.exe C:\BatchControl  -5.00   5.00   7.50
START /B /WAIT C:\fieldp_pro\tricomp\permag.exe C:\BatchControl.PIN
START /B /WAIT C:\fieldp_pro\tricomp\permag.exe C:\BatchControl.SCR
START /B /WAIT C:\fieldp_pro\tricomp\mesh.exe C:\BatchControl  -6.00   6.00   9.00
START /B /WAIT C:\fieldp_pro\tricomp\permag.exe C:\BatchControl.PIN
START /B /WAIT C:\fieldp_pro\tricomp\permag.exe C:\BatchControl.SCR
START /B /WAIT C:\fieldp_pro\tricomp\mesh.exe C:\BatchControl  -7.00   7.00  10.50
START /B /WAIT C:\fieldp_pro\tricomp\permag.exe C:\BatchControl.PIN
START /B /WAIT C:\fieldp_pro\tricomp\permag.exe C:\BatchControl.SCR
START /B /WAIT C:\fieldp_pro\tricomp\mesh.exe C:\BatchControl  -8.00   8.00  12.00
START /B /WAIT C:\fieldp_pro\tricomp\permag.exe C:\BatchControl.PIN
START /B /WAIT C:\fieldp_pro\tricomp\permag.exe C:\BatchControl.SCR
...
```

The file seems verbose, but it is mainly copy-and-paste from a template prepared with the *Create task* button of **FPController**. The interesting lines are those that call **Mesh**. The first pass parameter is the prefix of the input script listed above. The three additional string parameters give numerical values for the variables %1, %2 and %3. There are multiple solutions with expanding boundaries with the constraint $r_{max} = 1.5z_{max}$. The two commands that follow run **PerMag** with the modified mesh and then execute the analysis script `BatchControl.SCR`. This file has the following content:

```
INPUT BatchControl.POU
OUTPUT BatchControl.DAT Append
POINT 0.0 0.0
ENDFILE
```

The *Append* specification in the second command ensures that all the data will be added in sequence to a single output file.

The full data set is generated in about two seconds by executing the batch file. The data are available as text entries in `BatchControl.DAT`. Figure 27 shows a plot of the results. The dashed line is the theoretical result from Eq. 5. The difference from the infinite-space result is about 10.4% for the close boundaries ($z_{max} = 5.0$ cm) and about 0.4% for the large boundaries ($z_{max} = 20.0$ cm). The next chapter discusses the role of steel in magnet design. In particular, we will improve the example solenoid, providing external shielding and minimizing the drive power to achieve a given internal field.

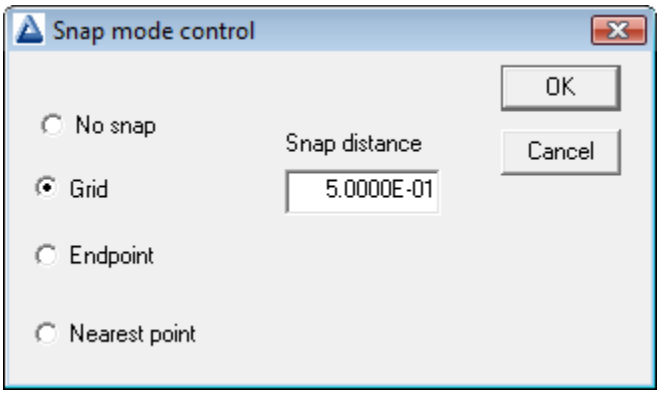


Figure 28: Set snap mode dialog of the **Mesh** drawing editor.

8 Magnetostatic solution: the role of steel

To complete our study of the solenoid coil, we'll proceed to a practical design by adding a magnet-steel shield. Here, the term *magnet-steel* designates a soft material with high relative permeability. The term *soft* means that the steel has a narrow hysteresis curve and has negligible permanent magnetization. Magnet-steel serves three functions in an electromagnet:

1. High-μ materials act as conductors of magnetic flux with little expenditure of energy. The use of atomic currents of materials to carry the return flux of a magnet means that the real currents in the drive coil can be reduced.

2. Ferromagnetic materials act as shields. Return flux prefers to flow through the steel, reducing the fringing fields of the magnet.

3. The magnetic flux density **B** is constrained to point almost normal to the surface of a material with $\mu_r \gg 1.0$. Some control may be exerted over field variations by shaping the surfaces of the steel.

The example will demonstrate these effects. The underlying assumption is that the fields generated by drive coils are low enough so that the ferromagnetic material is not driven into saturation. The next article discusses the nature of saturation effects and how to model them.

We'll start with the magnetic solution for a bare cylindrical coil discussed in the previous chapter with boundaries $z_{min} = -6.0$ cm, $z_{max} = 6.0$ cm and $r_{max} = 9.0$ cm. We'll add an additional region to the input script `BareCoil.MIN` to represent the external iron shield using the **Mesh** *Drawing Editor*. Run `tc.exe`, set the *Data folder* to the working location and launch **Mesh**. Use the command *File/Load/Load script (MIN)* and choose `BareCoil.MIN`. Then, use the command *Edit script/Edit script (graphics)* to open the drawing editor. The vectors for the three regions are displayed and current region is set to *3*. Click the *Start next region* tool. We will add vectors to represent the outline of the shield to *Region 4*. Click the *Set snap mode* tool to open the dialog of Fig. 28. For convenience, we specify that entered points will snap to the drawing coordinate system with a resolution of 0.5 cm.

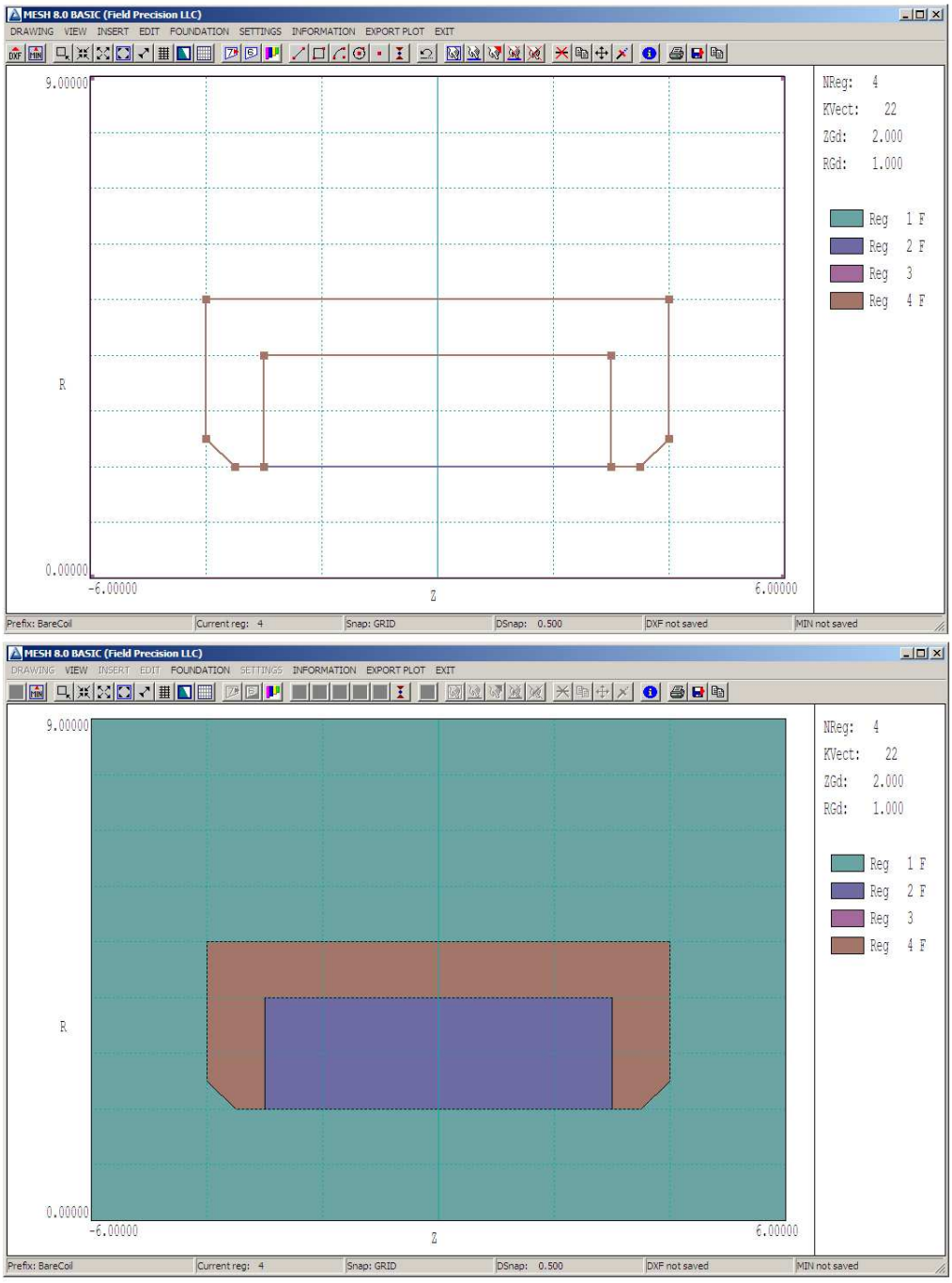


Figure 29: Top: adding a region in the **Mesh** *Drawing Editor*. Bottom: checking the fill status.

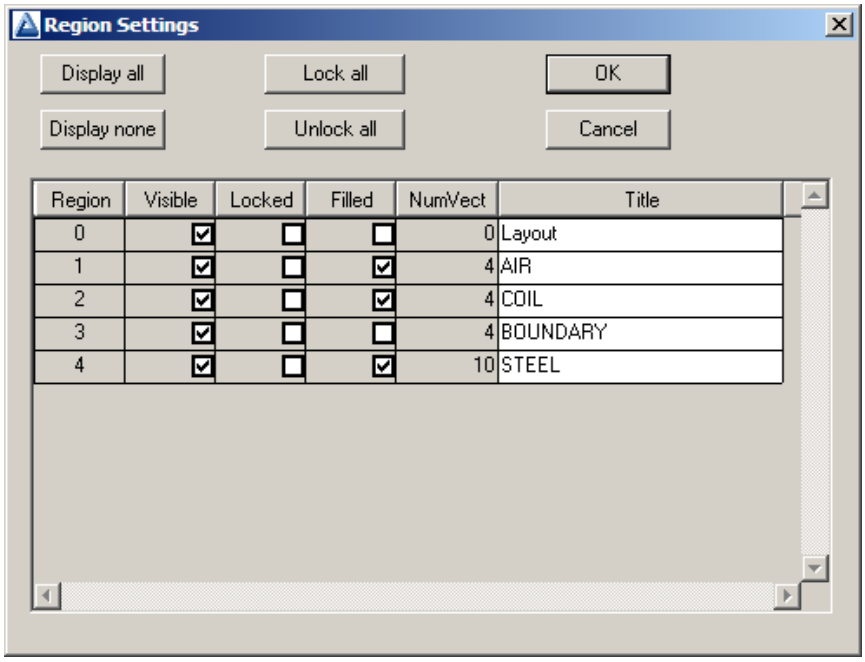


Figure 30: Region properties dialog in the **Mesh** *Drawing Editor*.

Click on the *Line* tool to enter a series of vectors that outline the shape shown in Fig. 29 (brown lines). In the line entry mode, move the cursor to snapped locations and left-click on the start and end points of each of the ten vectors. Be sure that they all connect and define a closed shape. Snap mode helps to ensure that the end point of one vector connects exactly to the start point of the next. When you have finished the last vector, right-click the mouse to exit the line entry mode.

Choose the command *Settings/Region properties* to open the dialog of Fig.30. Give the new region the name *Steel* and check the *Filled* box (we want to assign a high value of μ_r to all the enclosed elements). Exit the dialog. To confirm that the vectors of the new region constitute a connected and closed set, click the *Toggle fill display* command. The lower display of Fig. 29 shows valid filled regions.

Finally, it is a good practice in electrostatic and magnetostatic solutions to group regions with fixed boundary conditions (electrostatic potential or vector potential) at the end of the **Mesh** script. Choose the command *Settings/Region order*. Check the box for *Steel* and then click the *Move UP* button. Click *OK* to exit the dialog. To conclude. click the *Export MIN* tool and save the revised data as the file `SteelShield.MIN`. Exit the drawing editor. Click the *Load script (MIN)* tool and load the new script. Process the mesh and then click the *Save mesh (MOU)* tool to create the file `SteelShield.MOU`.

Run **PerMag**, click the *1* tool and choose the new mesh file. The dialog is similar to the previous example, except for the new region. Fill in the values as shown in Fig. 31. Save the file as `SteelShield.PIN` and then run a solution. To make a comparison, we need a solution without the steel. Here's a quick way to create it. Choose the command *File/Edit files* and pick `SteelShield.PIN`. Comment out the specification for high μ_r (put an asterisk at the beginning of the line) and replace it with the value for air:

```
* Mu(3) =   5.0000E+02
Mu(3) =   1.00
```

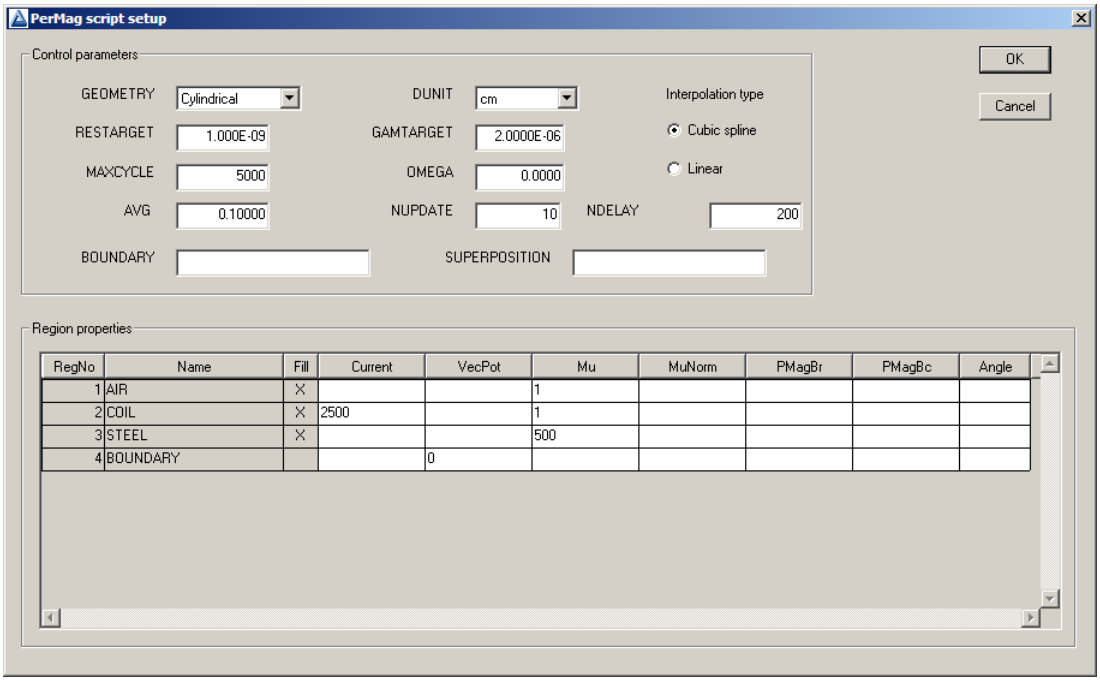


Figure 31: Dialog to set up the **PerMag** solution.

Save the result as `NoShield.PIN` and exit the editor. Then generate an additional **PerMag** solution.

We can find out a lot about the effect of steel just by looking at a plot of lines of magnetic flux density **B** (Fig. 32). With no steel shield, the lines spread out over the entire external region and the solution is strongly influenced by the boundaries. With the shield, the return flux lines are conducted through the steel. In this case, fringing fields are small and the boundaries have almost no effect on the solution. As expected, lines of **B** entering and exiting the steel are normal to the surface. The shield also compresses lines axially and $|\mathbf{B}|$ is more uniform within the coil.

For a quantitative comparison, we can inspect scans of magnetic flux density along the axis, $B_z(z, 0)$. Prepare and run the following analysis script:

```
NScan = 100
Output Shield_Analysis.DAT
Input NoShield.POU
Scan -6.0  0.0  6.0  0.0
Input SteelShield.POU
Scan -6.0  0.0  6.0  0.0
EndFile
```

Fig. 33 shows plots of computed values. The shield does improve axial containment of magnetic flux. A significant result is that field magnitude inside the coil (*e.g.*, the working volume of a magnetic lens) is 38% higher. Note that the two solutions have the same coil cross section and NI product. Alternatively, suppose the goal is to achieve a given central value $B_z(0, 0)$. The result of Fig. 33 indicates that the required NI product with the shield is only 73% that for the air coil, so the drive power would be cut almost in half. This effect arises because the coil need not supply field energy to support the return flux.

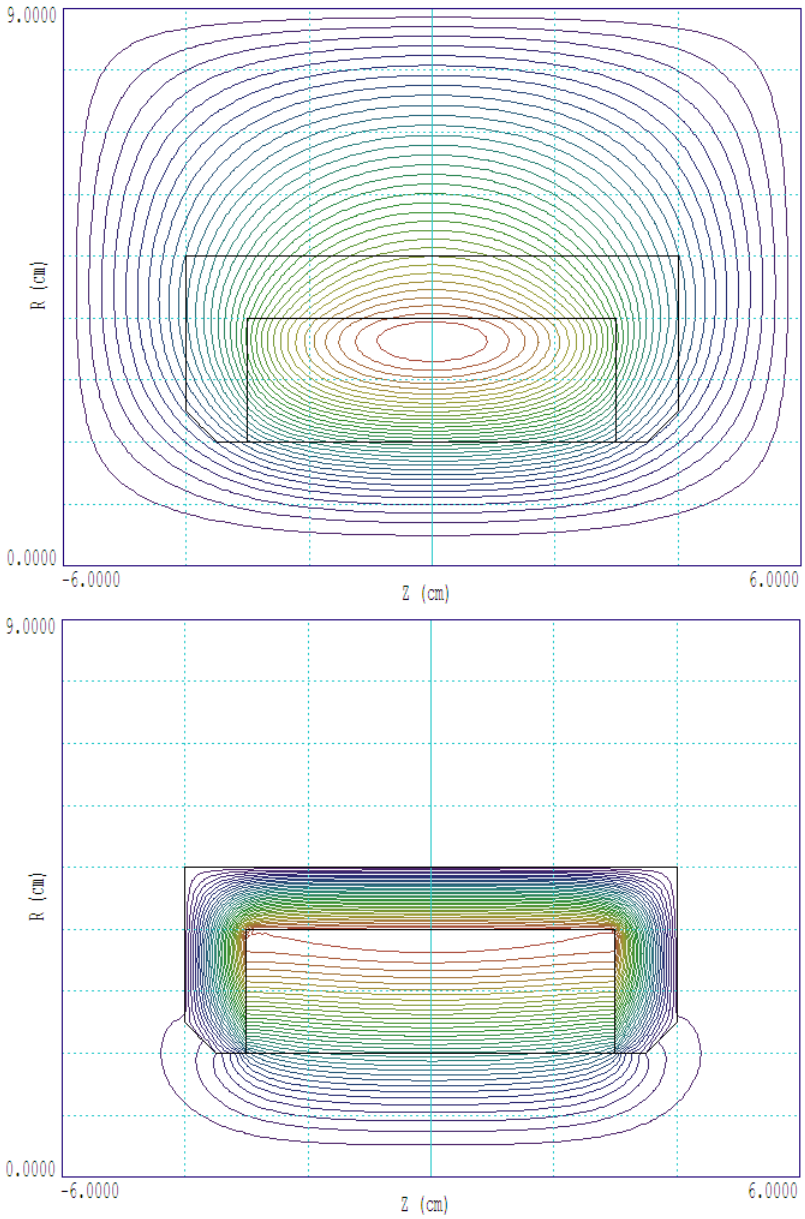


Figure 32: Lines of magnetic flux density **B**. Top: No shield. Bottom: With shield.

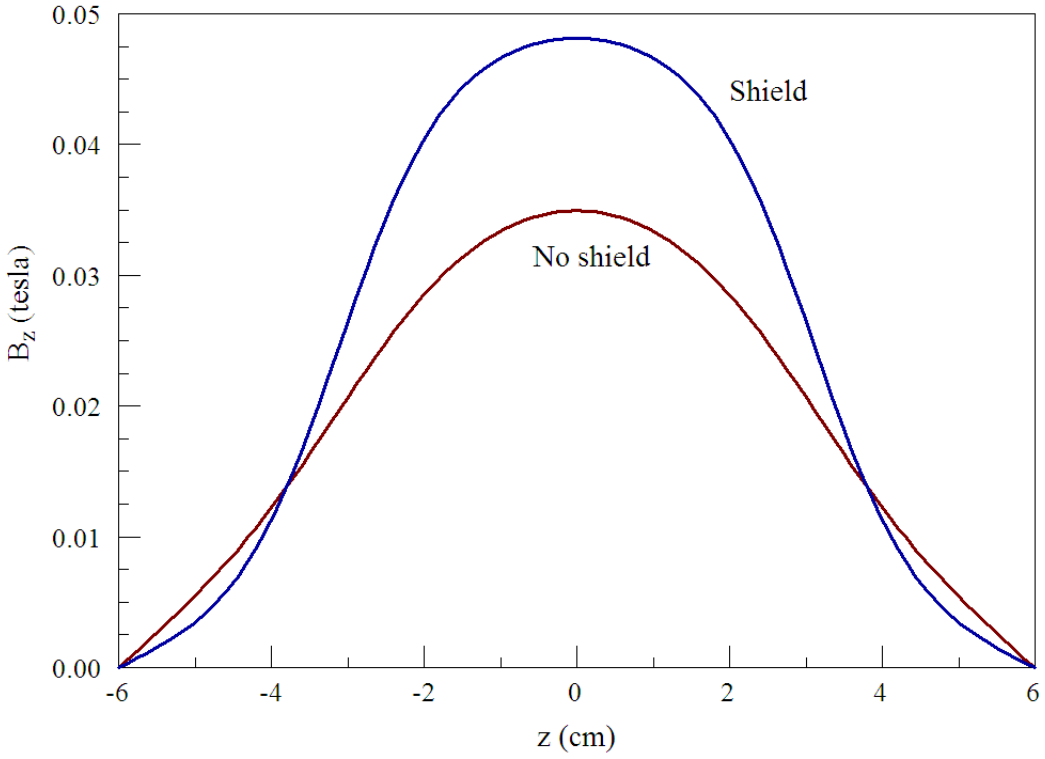


Figure 33: Scans of $B_z(z, 0)$ with and without the shield.

The condition of a fixed value of μ_r applies if the atomic currents in the iron are proportional to the drive currents in the coil (*i.e.*, the hysteresis curve approximates a straight line). Sometimes, the hysteresis curve may have a more complex variation. Even more important, there is a maximum value of atomic current in the material equivalent to alignment of all magnetic domains. At some value of coil current, the proportionality can no longer hold. The effect is called *saturation* of the magnetic material. The next chapter discusses how the non-linear effect of saturation is represented in **PerMag**.

9 Magnetostatic solution: when steel gets complicated

Complex material behavior is the main reason why magnetostatic solutions are generally more difficult than electrostatic solutions. Usually, dielectrics can be characterized by a single value of the relative dielectric constant ϵ_r up to the point where they break down. On the other hand, magnetic materials do unusual things even at normal values of flux density $\mathbf{B}$:

> Magnet steels may exhibit variations of μ_r, with a drop to $\mu_r = 1.0$ at higher field levels (*saturation*).
>
> In permanent magnets, the domains are locked in place, almost independent of the applied field.

We'll talk about how to model permanent magnets in the next chapter. Here, we'll concentrate on non-linear magnetic materials.

First, some definitions. The quantity $\mathbf{B}_0$ is the magnetic flux density at a point in space created by coils or permanent magnets. We'll call it the *applied flux density*. In the absence of steel, the total flux density is given by $\mathbf{B} = \mathbf{B}_0$. In the **PerMag** program, the relative magnetic permeability of isotropic materials is defined as $\mu_r = |\mathbf{B}|/|\mathbf{B}_0|$. Therefore, $\mu_r = 1.0$ in air or vacuum. In steel, the alignment of domains creates a material flux density that adds to the applied flux density. Therefore, μ_r is greater than 1.0. As we saw in the previous article, the utility of steel follows from that fact that $\mu_r \gg 1.0$.

Magnetic materials are characterized by curves of the form $|\mathbf{B}|$ versus $|\mathbf{B}_0|$ (or $|\mathbf{B}|$ versus $|\mathbf{H}|$, where $\mathbf{H} = \mathbf{B_0}/\mu_0$). If the variation is a straight line, then μ_r = constant and we say that the material is *linear*. Magnetic materials may exhibit linear behavior at low fields, but they always become non-linear at high fields (a few tesla). Equivalently, we can characterize materials with a plot of $|\mathbf{B}|$ versus μ_r. Figure 34 shows such a plot for steel 1020, a common material for magnet cores. At low $|\mathbf{B}|$, the value of μ_r is much greater than unity and changes considerably (*i.e.*, the $|\mathbf{B}|$ versus $|\mathbf{B}_0|$ curve is not a straight line). Physically, the curve implies that it takes some pushing to start aligning domains but things get easier when they begin to come around.

The notable feature of the curve is that μ_r approaches 1.0 when all the domains are aligned. In this case, the material contribution to $\mathbf{B}$ does not get higher as $\mathbf{B}_0$ increases, so that $\mathbf{B} \to \mathbf{B}_0$. The saturation flux density corresponds to the point where all domains become aligned. For steel 1020, the value is $B_s = 1.75$ tesla. **PerMag** can perform self-consistent calculations for non-linear materials using $|\mathbf{B}|$ versus μ_r data[6]. **PerMag** simultaneously adjusts the value of μ_r in elements based on the present value of $|\mathbf{B}|$ as it performs the iterative matrix solution of the finite-element equations.

Let's work on some calculations. The first question is whether the variations of μ_r below saturation will make a significant difference in the calculation. In other words, do we need to worry about getting the $|\mathbf{B}|$ versus μ_r curve exactly right? To illustrate, we'll use the example of the H magnet illustrated in Fig. 35. It has a planar rather than cylindrical geometry. Here, there are variations in x-y but the steel core and coils extend an infinite distance in z. Such a

[6]The values of Fig. 34 are used as **PerMag** input for the calculations of this chapter

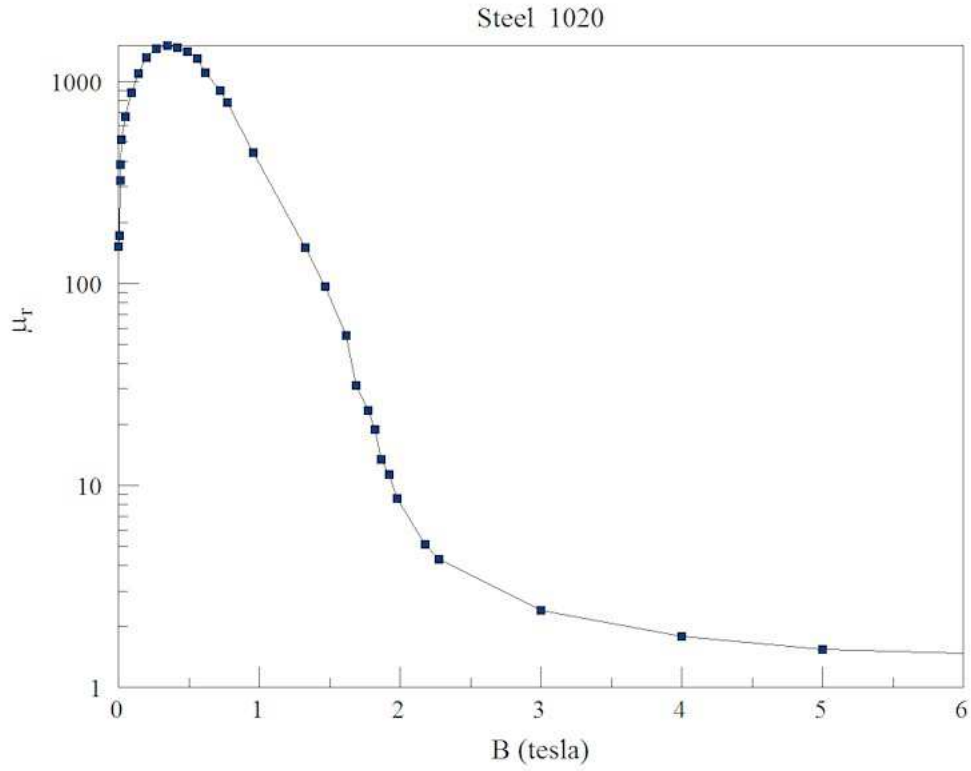


Figure 34: Plot of relative magnetic permeability versus |**B**| for Steel 1020.

calculation is often a good approximation to the central section of a long dipole magnet. Both sets of coils produce a upward-directed flux that is conducted by the steel to the air gap, the working volume.

To start, we'll address the question: how does the value of μ_r in the core affect the field distribution? We'll use the command-line parameter method discussed in Chap. 7, assigning fixed μ_r values over the range 5.0 to 10,000 and comparing values |**B**| in the air gap. Before beginning, its useful to recognize that the solution of Fig. 35 involves more work than is necessary. The fields in the four quadrants are mirror images. As an alternative, we could find the solution only in the first quadrant by applying appropriate symmetry conditions along the boundaries $x = 0.0$ and $y = 0.0$. Lines of **B** are normal to the boundary at $y = 0.0$. Here, we could apply the Neumann condition (the natural condition for a finite-element solution). Lines of **B** are parallel to the other three boundaries, implemented by the Dirichlet condition $A_z = 0.0$.

Figure 36 shows a plot of B_y in the magnet air gap at the midplane (0.0,0.0) and edge (2.0,0.0) of the magnet gap as a function of μ_r. The relative magnetic permeability has a strong influence at low values but little effect for large values. We can understand what's happening by inspecting Fig. 37, a plot of lines of **B**. The top illustration shows a solution with high μ_r. In this case, the low-reluctance core carries most of the flux, which flows up between the coils and returns across the air gap. A small fraction takes a short-cut around the outside of the outer coil (*leakage flux*). The reluctance of the air gap causes the flux to spread out to increase the cross-section area (*fringing flux*). Note that the lines of **B** entering and exiting the core are almost normal to the surface. In contrast, the lower illustration shows the case with low μ_r. In this case, the core has a high reluctance, increasing the leakage flux. A reduced fraction

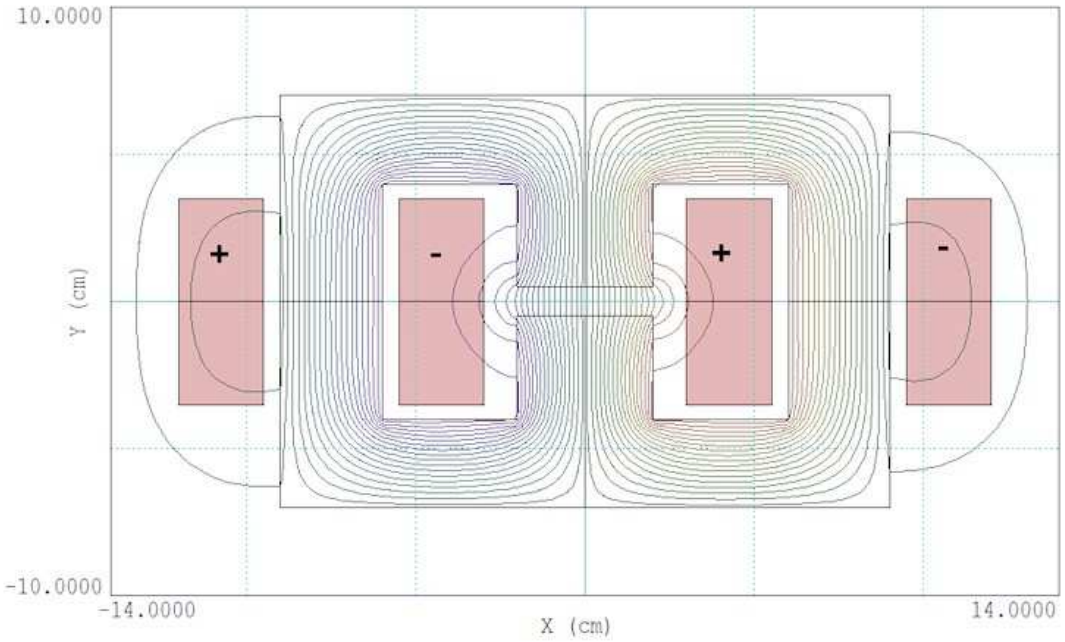


Figure 35: Geometry and calculated lines of **B** in the cross section of an H magnet.

of the flux is transported to the air gap, hence the reduced value of B_y. Finally, we'll consider why there is little change in the solution when μ_r changes by a factor of 10 between 1000 and 100000. The important quantity in a finite-element magnetic field calculations is not μ_r, but actually $\gamma = 1.0/\mu_r$. A value $\gamma = 1.0$ represents air, and a value $\gamma \to 0.0$ is characteristic of unsaturated steel or iron. Both 1/1000 and 1/10000 are close to zero, so the change in μ_r makes little change in the solution.

We'll conclude with a full non-linear calculation for the H magnet. Table 2 shows the script `HMagnet.PIN` that controls the **PerMag** calculation. *Region 2* is the steel core. Instead of assigning a fixed value, the code reads the table of information plotted in Fig. 34. The coils are set up for two calculations at high and low current. The values of drive current were chosen for solutions such that the core steel is well below and well above saturation.

Because the two processes of the material adjustment and the iterative matrix solution are performed simultaneously, program controls must be optimized to ensure convergence. Some experimentation may be necessary. Consider the following commands:

```
MaxCycle = 8000
Avg = 0.05
Update = 20 500
```

The quantity *MaxCycle* is the maximum number of matrix iterations, set to a fairly high value. The quantity *Avg* controls averaging of μ_r values between cycles. The low value is used to avoid oscillations of μ_r. Finally, the *Update* command states that μ_r should be recalculated every 20 iteration cycles and that the program should wait 500 cycles before corrections to ensure a good starting solution.

Figure 38 shows the variation of μ_r at low and high current. At low current (upper solution), the relative permeability exceeds 1000 over most of the core volume. The low values at the top reflect the fact that $|\mathbf{B}|$ is small and the elements are on the left-hand side of the curve of

Table 2: File `HMagnet.PIN`

```
Mesh = HMagnet
Geometry = Rect
DUnit = 1.0000E+02
ResTarget = 1.0000E-09
MaxCycle = 8000
Avg = 0.05
Update = 20 500

* Region 1: AIR
Mu(1) =   1.0000E+00

* Region 2: COREUPRT
Mu(2,Table) = steel1020_permag.dat

* Region 3: COILUPRTIN
Mu(3) =   1.0000E+00
* Low current
* Current(3) =   1000.0
* High current
Current(3) =   10000.0

* Region 4: COILUPRTOUT
Mu(4) =   1.0000E+00
* Low current
* Current(4) =  -1000.0
* High current
Current(4) =  -10000.0
* Region 5: BOUNDARY
VecPot(5) =   0.0000E+00

EndFile
```

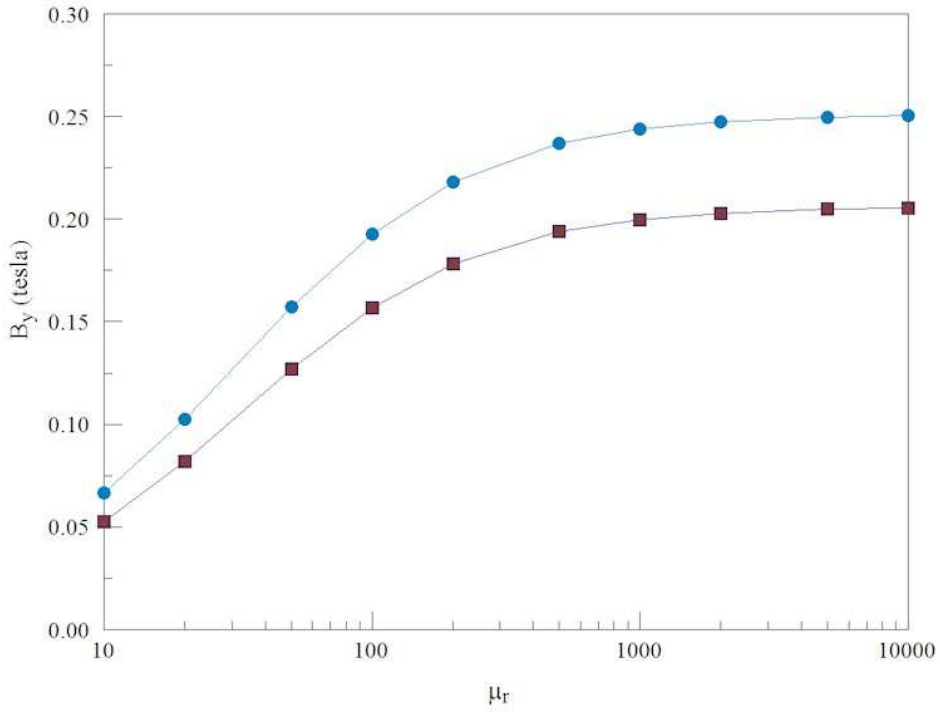


Figure 36: Variation of $B_y(0,0)$ [blue] and $B_y(2,0)$ [red] in the H magnet as a function of relative magnetic permeability in the steel core.

Fig. 34. An inspection of lines of B in Fig. 39 (top) shows that they are contained mainly in the core. Lines outside the core are normal to the surface. The midplane field is $B_y(0,0) = 0.2459$ tesla. At high current (Fig. 39 bottom), many regions of the core are driven into saturation, particularly the vertical piece adjacent to the air gap. The figure shows enhanced fringing flux near the air gap. The central field value is $B_y(0.0) = 1.484$, only 6.03 times the field at one tenth the drive current. Note that the field lines near the air gap are not normal to the core surfaces, affecting the profile of **B** across the gap.

In summary, we addressed the following issues in this chapter:

Typical variations of μ_r for soft magnetic materials.

Physical implications of the magnitude of μ_r, particularly as it affects the reluctance of steel structures in magnets.

Procedures to set up and to interpret non-linear solutions in **PerMag**.

The next chapter will discuss how permanent magnets work and how to build solutions for 2D permanent magnet devices.

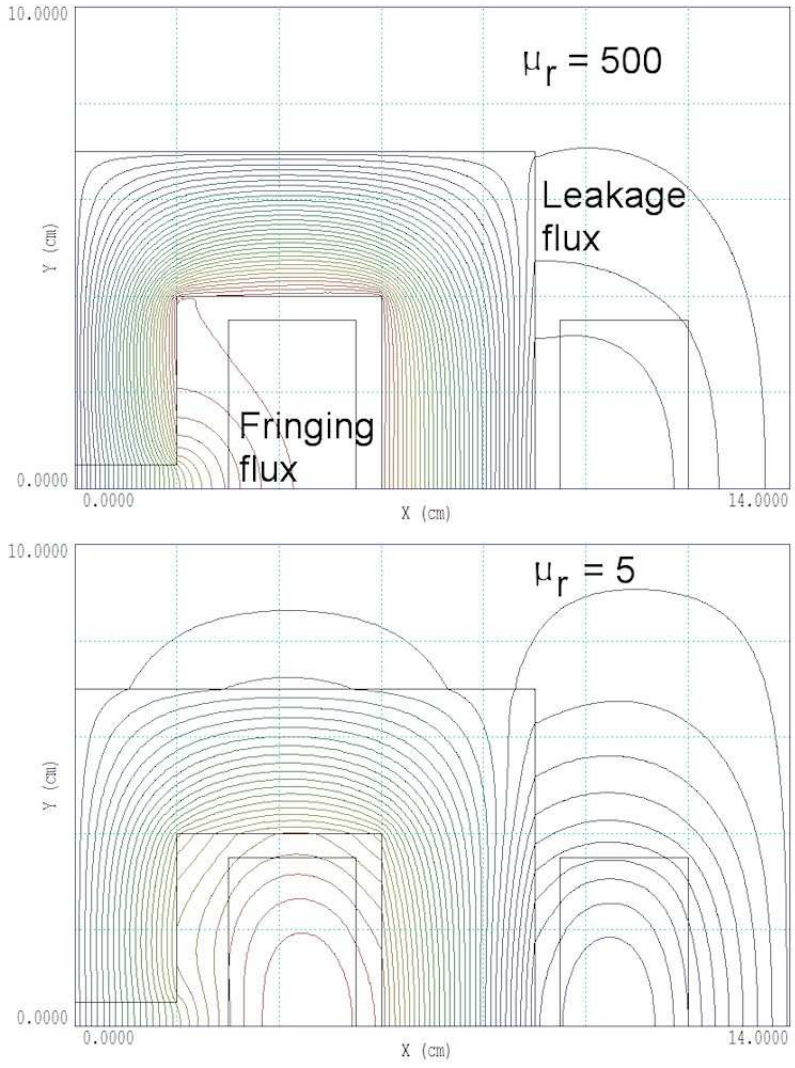


Figure 37: Lines of **B** at high and low values of relative magnetic permeability.

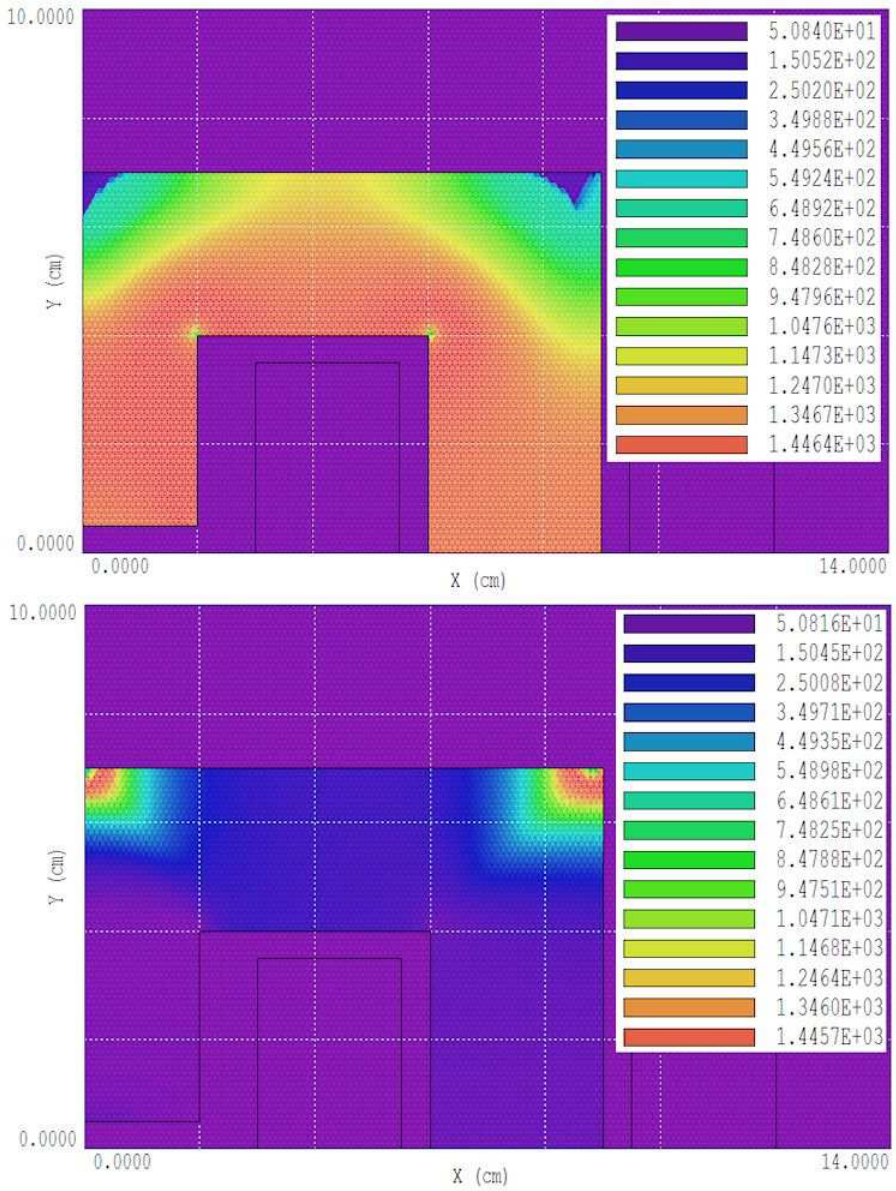


Figure 38: Spatial variation of relative magnetic permeability in the H magnet solution for low (top) and high (bottom) drive currents.

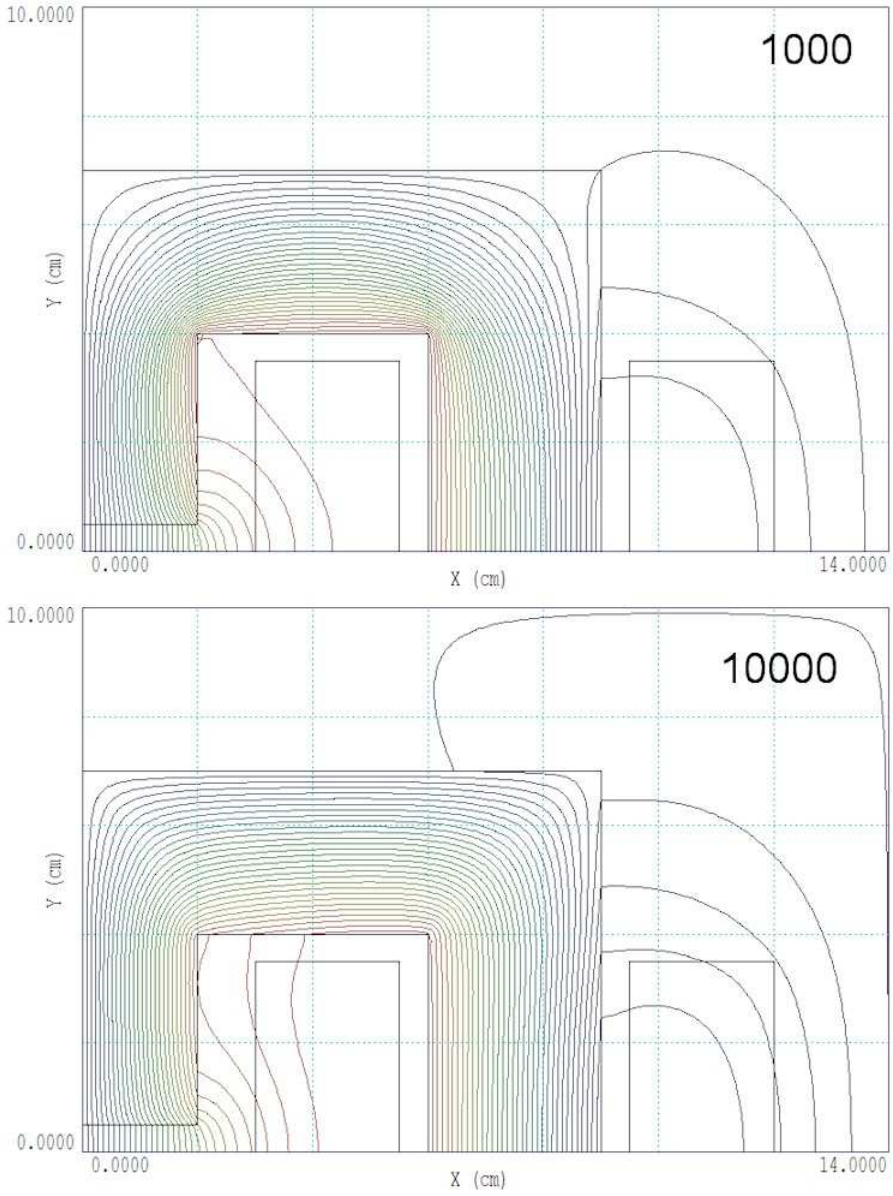


Figure 39: Lines of **B** in the H magnet solution for low (top) and high (bottom) drive currents.

10 Magnetostatic solution: permanent magnets

As the final topic in two-dimensional magnetostatics, we'll consider solutions with permanent magnets. To start, it's useful to review how permanent magnets work. Introductpry texts on electromagnetism sometimes don't treat the topic, and the explanations in many specialized references are overly complex.

The electrons of many atoms carry a circulating current, like a small current loop. Such an atom has a magnetic moment. The magnetic moment is a vector pointing from the center of the loop normal to the current. It points in the direction of the magnetic flux density $\mathbf{B}$ created by the loop. The distinguishing feature of ferromagnetic materials is that the atoms prefer to orient the currents in the same direction (a quantum mechanical effect). A region where all atoms are aligned is called a *domain*. The upper section of Fig. 40 shows the summation of aligned atomic currents in a domain. Currents on the inside cancel – the net result is a surface current in the direction normal to the magnetic moment.

In the natural state of a ferromagnetic material, the orientation of domains is randomized so that there is no macroscopic field outside the material (Fig. 40 lower). To generate such a field would require energy. Let's say that we supplied the energy by placing the material inside a strong magnet coil. In this case, the domains line up and the current of all domains sum up as in the upper section of Fig. 40. The domain currents cancel inside, but there is a net surface current on the object. If we turn off the coil, the domains of a *soft* magnetic material return to a random distribution. On the other hand, suppose we could physically lock the domains in position before we turned off the coil. In such a *hard* material, the object retains its surface current and can generate external fields. The energy for the field was supplied by the magnetizing coil and it was bound in the material. Such a object is called a *permanent magnet*.

The distinguishing feature of modern permanent magnet materials like neodymium-iron and samarium-cobalt is that domain locking is extremely strong. The domains remain lined up, independent of external processes. This property makes it simple to model the materials. Let's review some definitions and facts. The direction of the magnetic moments of the aligned atoms (and domains) is called the *magnetization direction*. Suppose we have a permanent magnet that is long along the direction of magnetization and self-connected (*e.g.* a large torus). In this case, there is no external field and the flux density inside the material is generated entirely by the surface currents. This intrinsic flux density is called the *remanence flux*, B_r, the most important quantity for characterizing a permanent magnet. A typical value for neodymium iron is $B_r = 1.6$ tesla.

We can express the surface current density in terms of the remanence flux. Take $\mathbf{B}_r$ as a vector pointing along the direction of magnetization and let $\mathbf{n}_s$ be a vector normal to the surface of the permanent magnet. The surface current density is given by

$$\mathbf{J}_s = \left(\frac{1}{\mu_0}\right) \mathbf{B_r} \times \mathbf{n}_s \quad \text{(A/m)}. \tag{6}$$

We can use **PerMag** to confirm this physical interpretation. The top section of Fig. 41 shows a calculation in cylindrical geometry for an annular permanent magnet in space (*i.e.*, no iron, coils or other permanent magnets). The remanance field is $B_r = 1.5$ tesla and the direction of magnetization is along z. The plot shows lines of $\mathbf{B}$. According to the interpretation of Fig. 40,

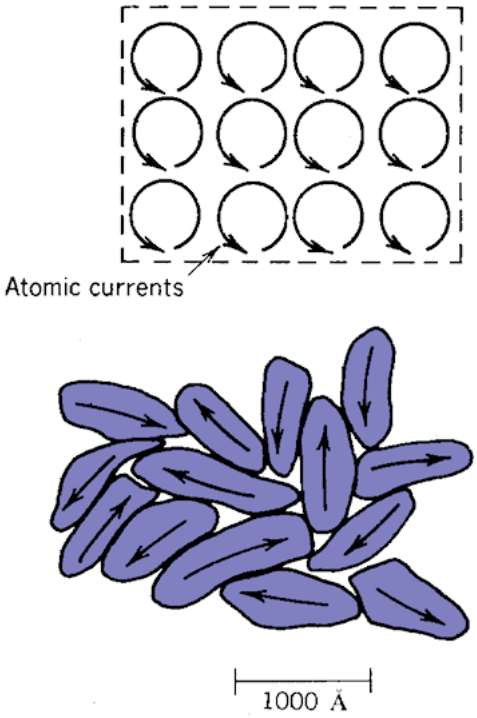


Figure 40: Top: alignment of atomic currents in a domain. Bottom: alignment of domains in unmagnetized material.

we would get the same results by replacing the permanent magnet with thin surface current layers. Equation 6 implies that there should be no current on the ends and uniform current density $J_s = 1.193$ MA/m on the inner and outer radial surfaces. In the second model (bottom section of Fig. 41), the permanent magnet is replaced by two thin solenoid coils (length = 0.08 m, thickness = 0.00125 m). The total current of the outer coil is $(1.193 \times 10^6)(0.08) = 95470.0$ A and the current on the inner coil is -95470.0 A. The lower section of Fig. 41 shows that the calculated lines of $\mathbf{B}$ are indistinguishable from the permanent magnet calculation. For a quantitative check, we can compare scans of B_z along the axis. Figure 42 shows the results. There small difference is a result of the finite thickness of the surface current layer.

If permanent magnets are that simple, where could confusion arise? Unfortunately, things are more challenging when we talk about older materials like Alnico. Such materials are characterized by a *demagnetization curve*. Typically, the curve is a plot of $|\mathbf{B}|$ inside the permanent magnet versus $|\mathbf{H}|$. Here, $|\mathbf{H}|$ is the applied magnetic field that arises from coils and other permanent magnets. It is easier to see the meaning of the curve if we make the plot in terms of the applied magnetic flux density, $\mathbf{B}_0 = \mu_0 \mathbf{H}$. Applied fields have almost no effect on a modern material. Therefore, the total flux density inside the material is simply

$$\mathbf{B} = \mathbf{B}_r - \mathbf{B}_0. \tag{7}$$

Figure 43*a* shows a plot. The value of applied magnetic field when $B = 0.0$ is called the *coercive force* H_c. For modern materials, the coercive force is

$$H_c = -\frac{B_r}{\mu_0}. \tag{8}$$

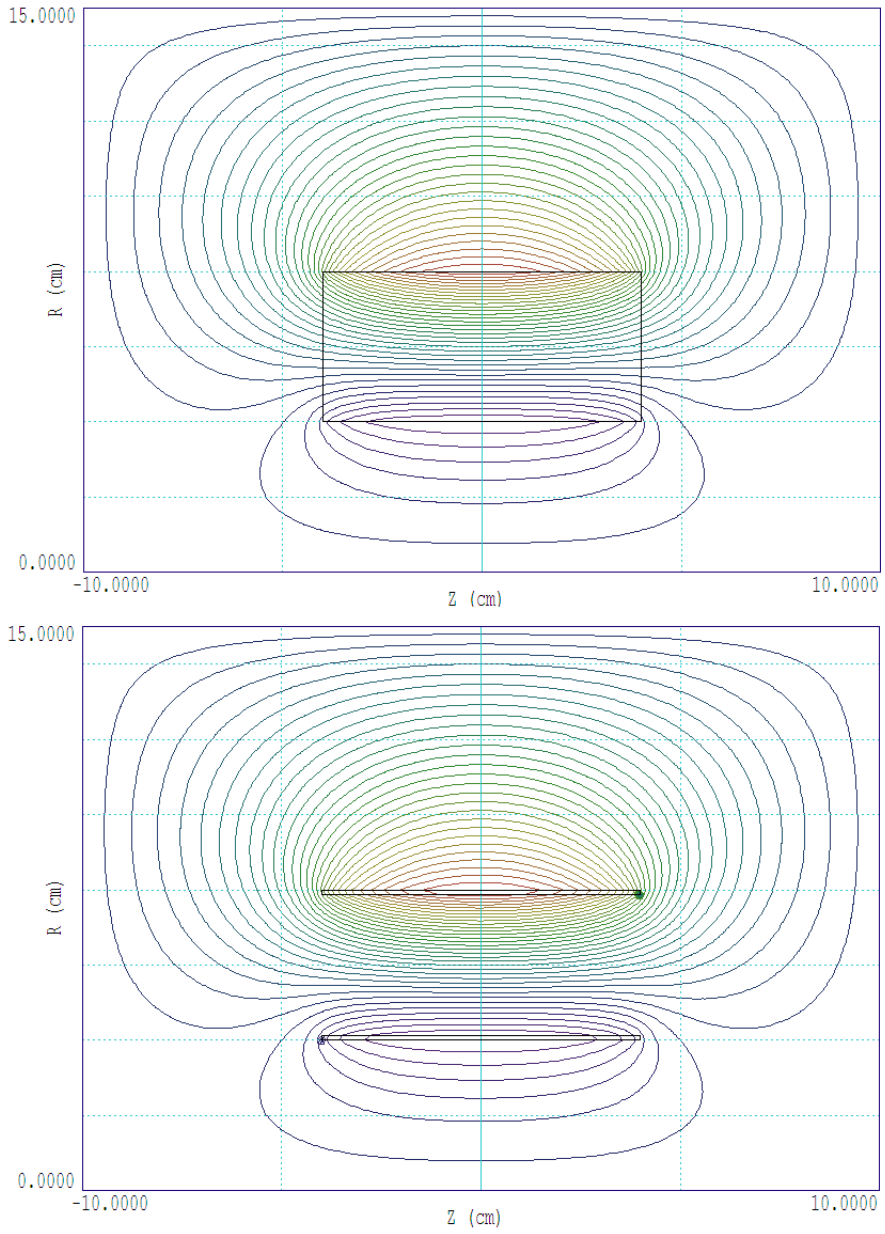


Figure 41: Lines of magnetic flux density $|\mathbf{B}|$ for an annular permanent magnet (top) versus equivalent surface current layers (bottom).

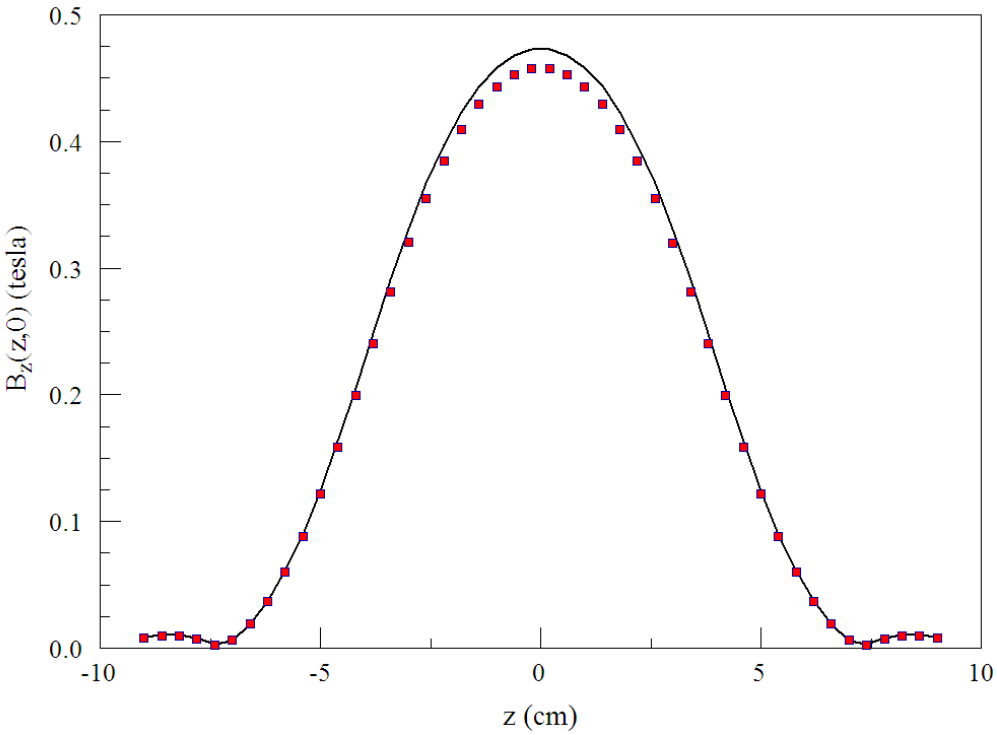


Figure 42: Scans of $B_y(x,0)$ for an annular permanent magnet (line) versus equivalent surface current layers (symbols).

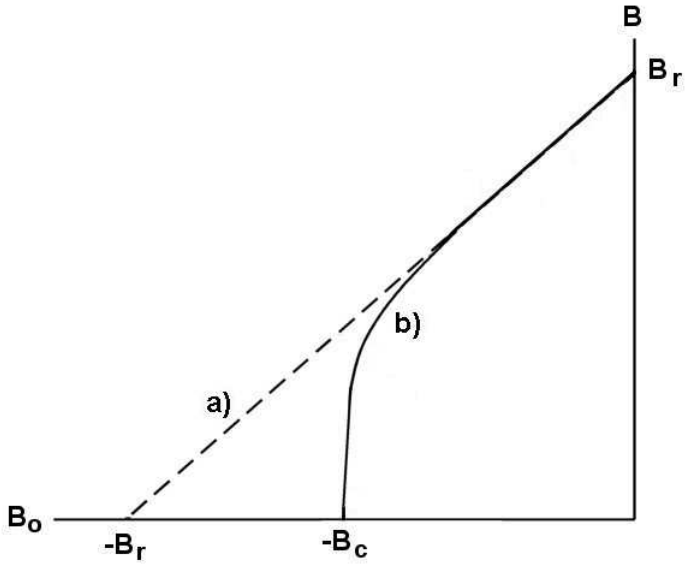


Figure 43: Demagnetization curves for modern (a) and older (b) permanent magnet materials.

It is clear that neither the demagnetization curve or the coercive force are particularly meaningful for materials like NdFeB. On the other hand, the concepts are useful for older materials. Here, the domains are not tightly locked. Putting such a magnet in a device with an air gap could cause degradation of the alignment so that the total flux density falls below the ideal curve (Fig. 43*b*). **PerMag** can solve such problems, but it is important to clarify what the solutions mean. Suppose you bought an Alnico magnet magnetized at the factory and shipped with an iron flux return clamp (*i.e.*, $B = B_r$). If you could move it instantaneously to the application device, then the operating point calculated by **PerMag** would apply. On the other hand, if you removed the clamp so the permanent magnet was exposed to a large air gap and possibly dropped it on the floor, then the material would have undergone an irreversible degradation and the fields generated will be lower than predicted. The best way to get the predicted performance based on the demagnetization curve is to magnetize the material *in situ*. One big advantage of modern materials (beside their strength) is that they achieve the predicted performance, even if they are magnetized at the factory, shipped from China and moved to the assembly.

We'll conclude with an application calculation, a bending magnet for an ion spectrometer. It is a long assembly, so we will do a preliminary 2D calculation in a cross section. The top section of Fig. 44 shows half the geometry (there is a symmetry plane with Neumann boundaries at $y = 0.0$ cm). The goal is to create a dipole field B_y in the air gap to bend ions in the x direction. The arrow shows the magnetization direction of the permanent magnet (NdFeB with $B_r = 1.6$ tesla). Two features ensure the maximum air-gap field for the given magnet surface currents:

A steel core carries the return flux.

The permanent magnet is placed close to the gap.

The figure shows the calculated lines of **B**. Figure 45 shows a scan of B_y along x across the air gap. The field is strong but non-uniform, probably of little use in a spectrometer. To correct things, we can take advantage of one of the properties of soft steel, field shaping. Consider the effect of adding a steel layer adjacent to the gap. The lower section of Fig. 44 shows the change in lines of **B**. The steel shifts flux away from the center to the edges of the gap. Figure 45 shows the effect on the field scan. With some sacrifice in the magnitude of the flux, the steel shaper gives a working volume of approximately uniform field.

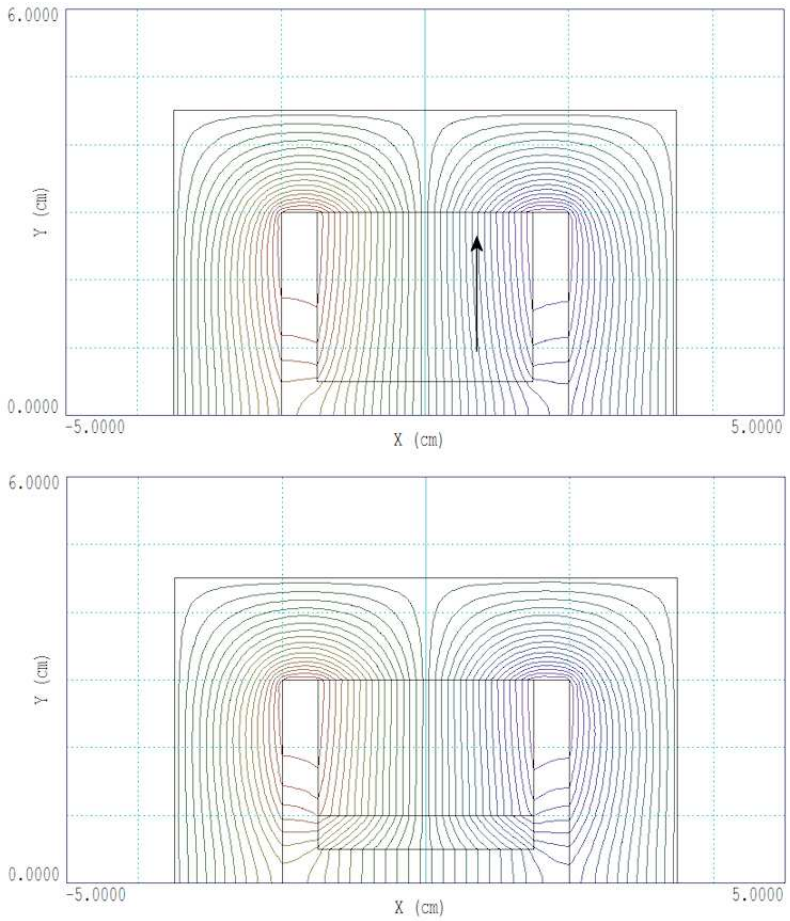


Figure 44: Application example, permanent-magnet ion spectrometer. Top: bare magnet. Bottom: magnet with steel insert for field shaping.

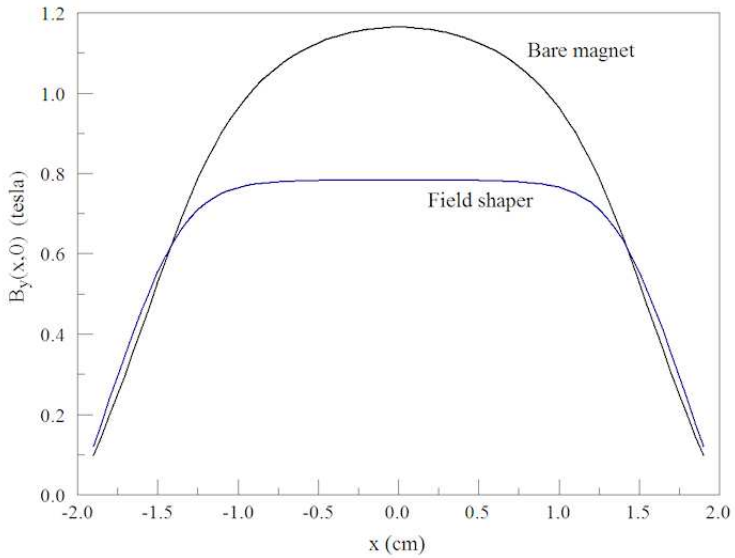


Figure 45: Spectrometer application, scan of $B_y(x, 0)$ with and without the steel insert.

11 3D electrostatic example: STL input

We'll assume that **HiPhi** has been installed, adding new active buttons in **FPController**. To introduce a 3D electrostatic calculation, we'll step through a complete field solution for an electron gun designed at the Lawrence Livermore National Laboratory to generate sheet beams. We'll use prepared input files – following chapters will help you prepare you own inputs.

To start, run **FPController**. If necessary, set the *Data folder* to point to a working directory. Copy the following files supplied in the example archive to the working directory:

```
sheetbeam.min
sheetbeam.hin
cathode.stl
focus.stl
output.stl
```

The text file `sheetbeam.min` is a set of instructions to **MetaMesh** defining the shapes and assembly method of the gun parts. The instructions were built in the interactive environment of **Geometer**. The file `sheetbeam.hin` gives **HiPhi** material data and other parameters needed to generate a finite-element solution. The stereolithography files `cathode.stl`, `focus.stl` and `output.stl` were supplied by a laboratory engineer who exported them from a **Solidworks** model. They define shapes that are too complex to create from a summation of simple solids.

As we saw in Chap. 5, the procedure for 2D shapes is relatively simple. Objects are created from a boundary outline of line and arc vectors. The outlines may be derived from or exported to `DXF` files. In contrast, shapes encountered in the 3D world may be much more complex. A flexible approach is essential to make the task manageable. The following principles underlie the operation of **Geometer** and **MetaMesh**. Their implications will become clear as we move through the example:

> Objects (like electrodes and dielectrics) are constructed from one or more *Parts*.
>
> Physical properties (like potential and relative dielectric constant) are assigned to *Regions*. Each *Part* belongs to a *Region*.
>
> An object may be constructed from several parts associated with the same region. For example, a grounded electrode may be composed of two parts (a spherical tip and cylindrical support rod) that belong to a region with fixed potential 0.0 V.
>
> Parts are processed in the order in which they appear in the **MetaMesh** input file. The currently-processed part overwrites any shared volumes with previously-processed parts.

Parts are defined at a standard position and orientation. They may be moved or rotated to a final configuration in the solution space.

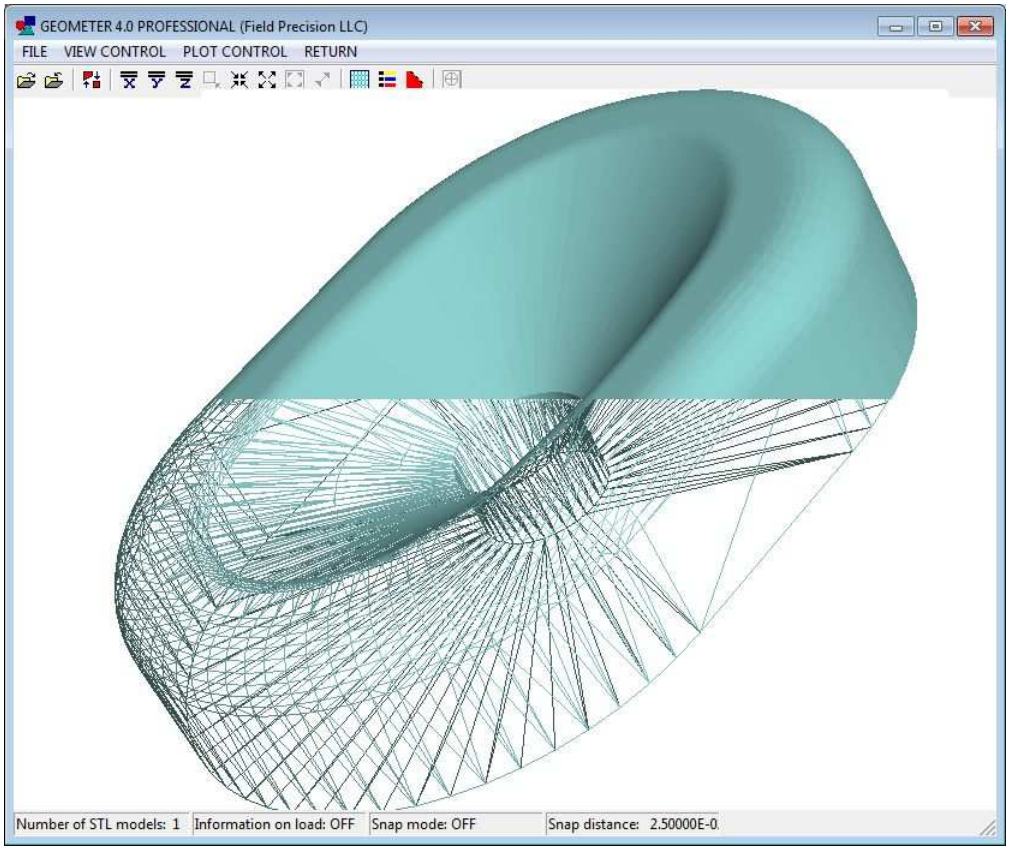


Figure 46: **Geometer** `STL` viewer, showing solid and wireframe views of `focus.stl`.

It's clear that the idea of a *part* is central to the mesh-building process. There are two types:

> Parametric models for simple shapes like cones or spheres. These models are built into **Geometer** and **MetaMesh** and do not require external data.
>
> Arbitrary shapes created with 3D CAD programs like **SolidWorks** and exported as files in the `STL` format. The files must be loaded to be used in **Geometer** or **MetaMesh**.

A mesh may combine both types of parts.

Let's get started with the calculation. First, it's useful to understand the information in `STL` files. Run **Geometer** and click the *STL viewer* command. The program becomes a full-featured viewer that can display the shapes defined by individual `STL` files and their relationships in space. Click *File/Add model* and choose `focus.stl`. Click *View control/Orthogonal/perspective* to enter the 3D mode. This is an interactive environment based on **OpenGL**. Move the mouse cursor to the sides and left click to change the view. You can also move the cursor toward the center and then left or right-click to zoom in or out. The view looks like the right-hand side of Fig. 46. The part is a dish-shaped focusing electrode with a hole for the cathode.

To see the inherent data of the `STL` file, click *Plot control/Model display*. Uncheck the *Solid* box to create a view like the left-hand side of Fig.46. The view shows the set of contiguous triangular facets that define the surface of the part. The `STL` file is simply a list of facets. The facets shown are typical of those created by 3D CAD programs. Although all facets

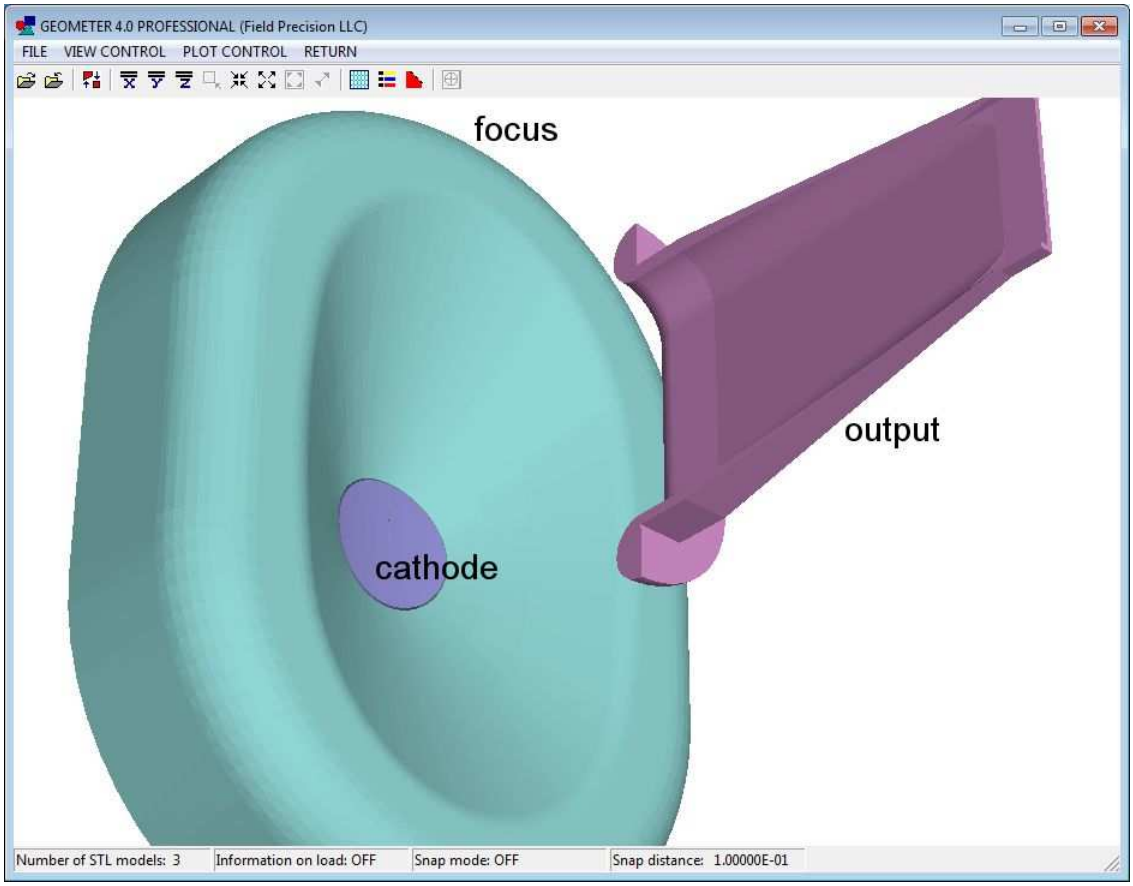


Figure 47: Full set of `STL` shapes for the assembly.

are theoretically correct, they may vary considerably in scale. To determine if an element is inside an `STL` shape, **MetaMesh** must analyze all facets, an intensive activity where parallel processing is a big advantage.

Return to a solid view of the surface and load the other two parts to get a view like that of Fig. 47. The engineer exported the parts from a SolidWorks *assembly*. In this case, the coordinates of facets in the `STL` file are absolute with respect to the assembly space rather than relative to the part. Therefore, the facets not only define the shape of parts, but also their positions and orientations relative to each other[7].

The parts as loaded define the full volumes of the cathode and focus electrode, but only half of the output transport tube. This is not a problem because by symmetry it's sufficient to perform a calculation only in the first quadrant of the x-y plane. **MetaMesh** automatically clips over-sized parts. There is one issue that you can see by shifting to the *Orthogonal* view and clicking *View control/Y normal axis*. The **Solidworks** model corresponds to a beam moving in the -z direction. For transport calculations, we want the beam to move in $+z$ following the standard convention. The modification will be made during **MetaMesh** processing.

Exit **Geometer** and run **MetaMesh**. *Click File/Load MIN file* and choose `sheetgun.min`. Click the *Process* mesh command and wait for the program to complete it's analysis. It takes about 20 seconds on a multi-core computer to generate a mesh with 1.3 million elements. Right-click to close the messages. Choose *Plot3D* to observe the display of Fig. 48. The plot

[7]Note that we could move the parts to different positions by adding shift operations in **Geometer**.

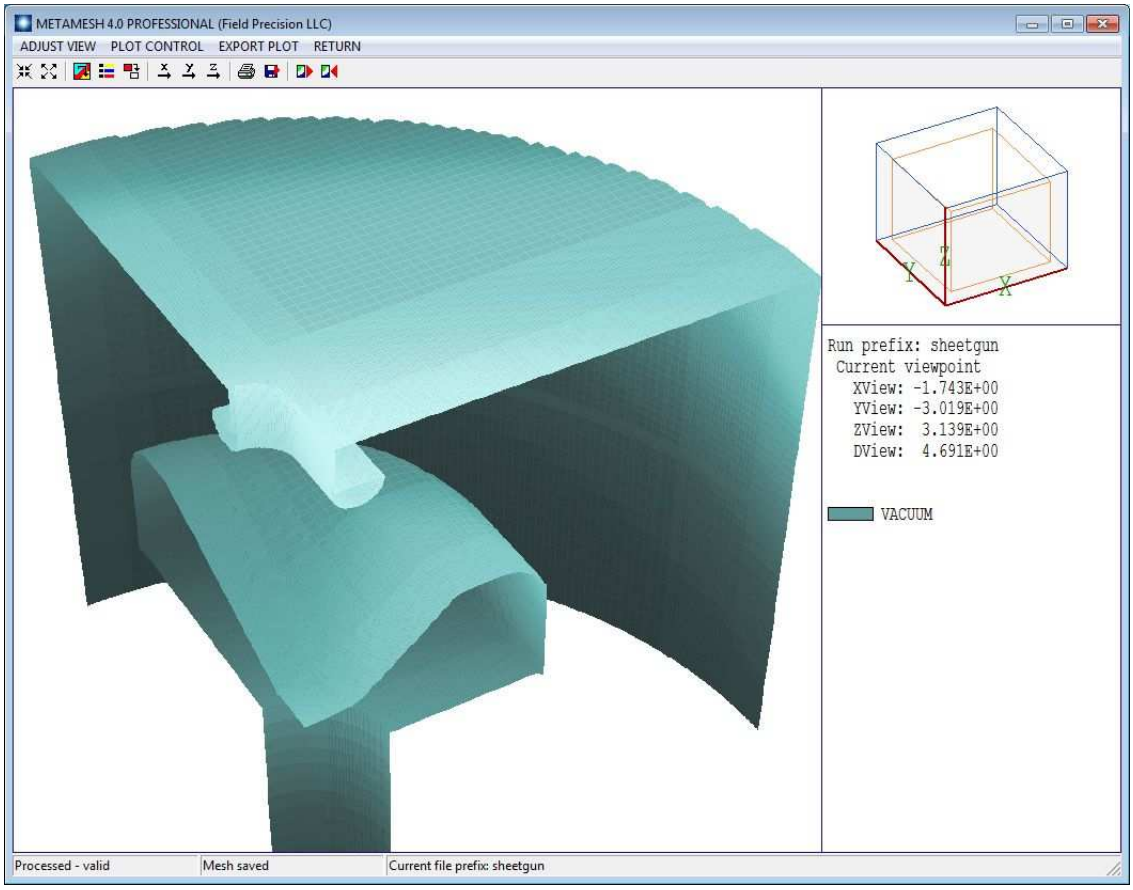


Figure 48: Completed mesh, outline of the vacuum region.

of the surfaces of elements of the vacuum region) indicates a good representation of all the electrodes. The parts defined by `STL` files are present, but much more has happened. The mesh has been created in the first quadrant with fine resolution of the cathode surface, the hexahedron elements conform closely to the theoretical shapes, the beam direction points in $+z$, the assembly is inside a cylindrical vacuum chamber and there is a support rod for the cathode. All will be explained in the next chapter where we take a close look at the **MetaMesh** script.

12 3D electrostatic example: mesh generation and solution

In this chapter, we'll conclude the example 3D electrostatic solution. The previous chapter introduced the concept of mesh *Parts* and *Regions*. The emphasis was on input from 3D CAD programs via `STL` files. Here, we'll consider how **MetaMesh** uses the `STL` data and processes extra information to create the result of Fig. 48. We could go through all the interactive operations in **Geometer** to define the parts, but there's a quicker way to understand the setup. The end result of **Geometer** activities is a **MetaMesh** input script that includes all the information. We'll study the script directly. A following chapter shows how to use **Geometer** to create a solution from scratch.

Run **FPController** and launch **MetaMesh**. We'll assume that the *Data folder* still points to the working directory of the previous chapter. Click *File/Load MIN* file and choose `sheetgun.min`. Then click *File/Edit MIN file* to view the contents of the script. The first part of the *Global* section has detailed specifications for element sizes over multiple zones along the axes. The feature was necessary for accuracy in subsequent beam-emission calculations – the details are not of immediate concern. Instead, we'll concentrate on the part definitions shown in Fig. 49[8].

The parts named *Anode*, *Focus* and *Cathode* (marked in green) use the `STL` models that we previously discussed. The *Type* command of each part section lists the associated `STL` file. The *Fab* (fabrication) command defines fitting controls – default values are usually sufficient. Although the intent is to build the mesh around the assembly positions of the *STL* parts, we do want to reverse the emission direction to $+z$. This is accomplished with the commands:

```
Rotate 180.0 0.0 0.0 XYZ
```

Each command rotates the coordinates of all facet nodes 180° about the x axis relative to the assembly origin. The commands

```
Surface Region VACUUM
```

instruct the code to adjust all element facets of the part adjacent to *Vacuum* elements to conform closely to the defined surface of the `STL` model.

The assembly is located inside a cylindrical vacuum chamber of radius 1.23" that extends from z = 0.0" to the boundary of the solution volume at z = -1.50". For high speed, **HiPhi** uses a structured conformal mesh, where *structured* means that the elements are arranged like the units of an apartment building. The term *conformal* means that the apartments are not necessarily right-angle boxes[9]. The solution volume of a structured mesh is always a box. To create the vacuum chamber, we'll fill the entire volume with elements assigned to the fixed-potential region *Ground* (part *VChamber*) and then carve out a cylindrical volume of elements associated with the *Vacuum* region. The part *ExitPort* is a slot to accommodate the beam exit aperture. The final part (*Support*) is a cylindrical pipe attached to the cathode. Note that this

[8]The text display is from the **ConText** editor using syntax highlighting definitions that we supply with the software.

[9]The general term for a six-sided solid where facets may not be at right angles is a *hexahedron*.

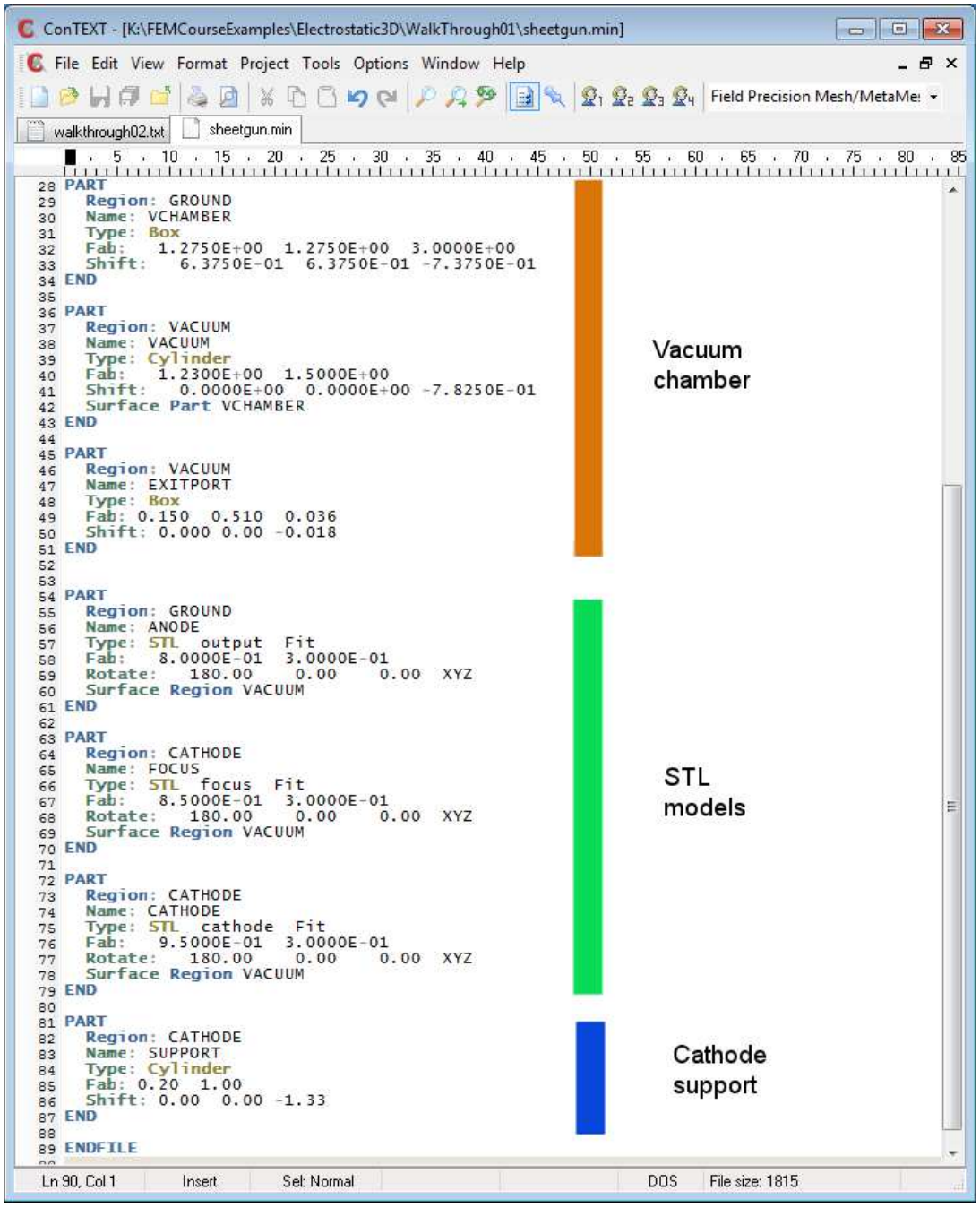


Figure 49: Part sections of the **MetaMesh** input script `sheetgun.min`.

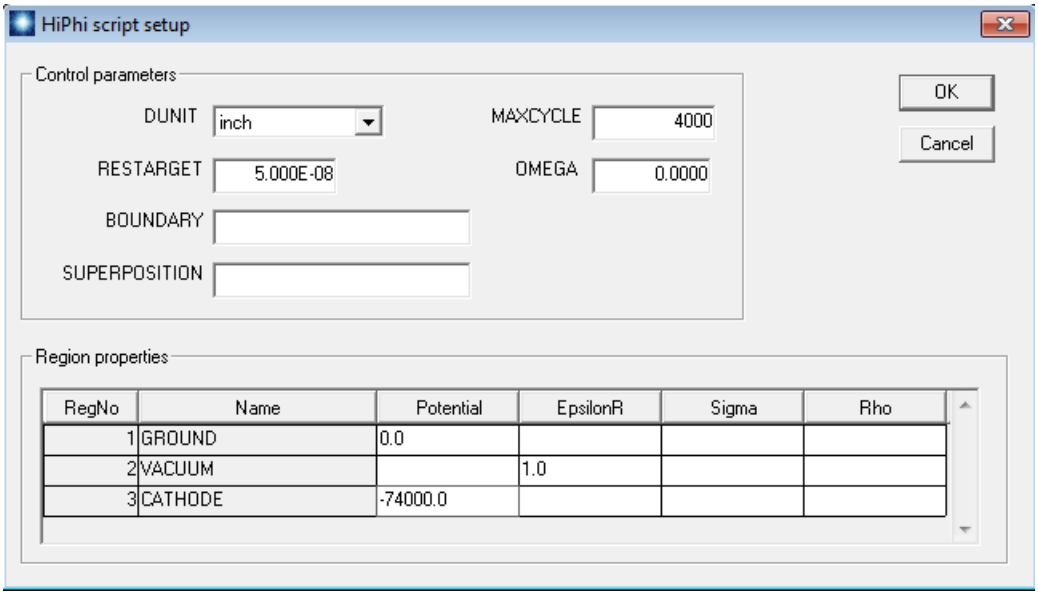


Figure 50: **HiPhi** dialog to set program parameters and the physical properties of regions.

part belongs to the same region as the cathode and the focus electrode. The physical property of the region in the **HiPhi** solution is the fixed potential (-74.0 kV).

Exit the text editor and click the *Process mesh* command. **MetaMesh** reports progress of the mesh creation. When the program is finished, right-click to exit the report and use the *File/Save mesh* command to create the file `sheetgun.mdf` (**M**esh **D**efinition **F**ile).

We can now proceed to the **HiPhi** solution. Run the program, click *Setup* and choose `sheetgun.mdf` to open the dialog of Fig. 50. Fill in the values as shown. Click *OK* and save the results as `sheetgun.hin`. The resulting file has the contents:

```
Mesh = sheetgun
DUnit =   3.9370E+01
ResTarget =   5.0000E-08
MaxCycle =    4000
Parallel = 4

* Region 1: GROUND
Potential(1) =   0.0000E+00

* Region 2: VACUUM
Epsi(2) =   1.0000E+00

* Region 3: CATHODE
Potential(3) =  -7.4000E+04

EndFile
```

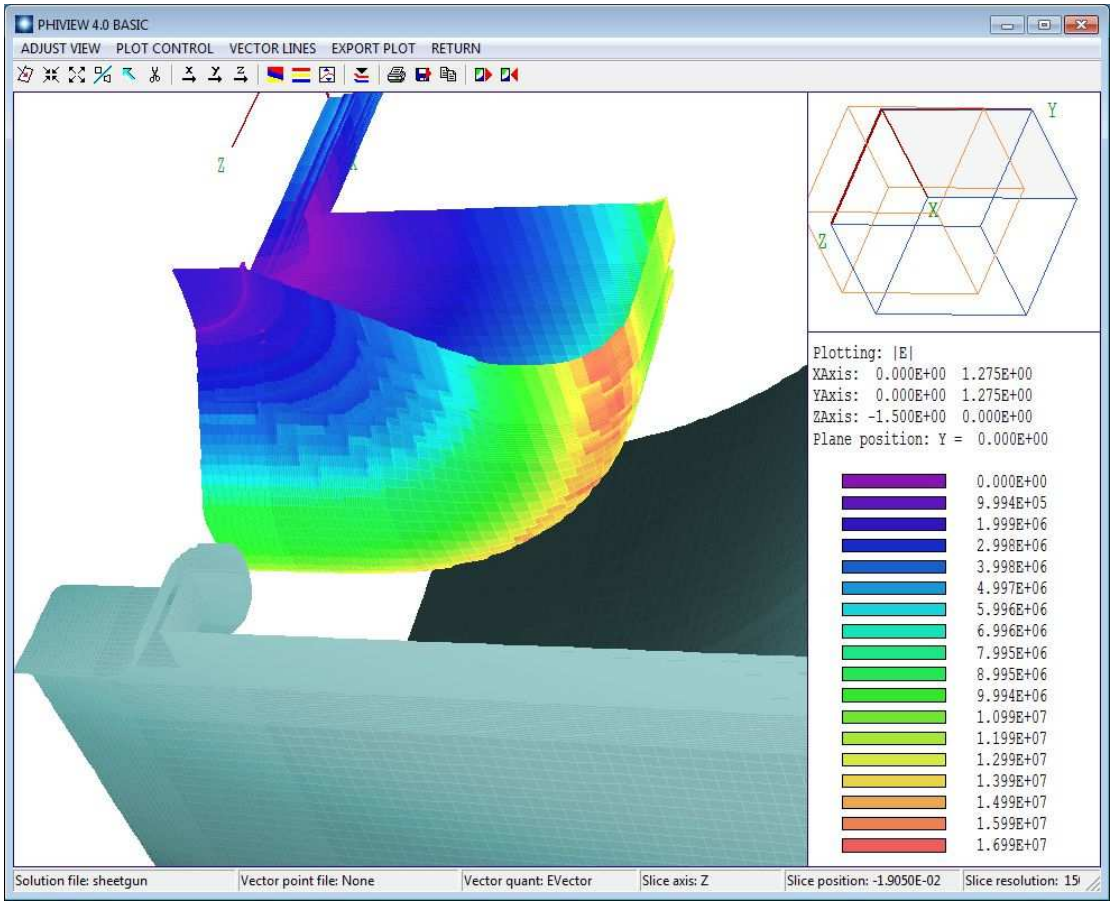


Figure 51: 3D plot of the assembly showing levels of $|\mathbf{E}|$ on the surface of the focus electrode.

The clean format of the file makes it easy to identify the functions of the commands. If you have the *Pro* version of **HiPhi**, the command

```
Parallel = 4
```

will reduce the run time.

Click the *Run* button and choose `sheetgun.hin`. The program generates a solution recorded as the binary file `sheetgun.hou`. We can get some useful information by checking the text listing files `sheetgun.mls` and `sheetgun.hls` created by **MetaMesh** and **HiPhi**. For example, the **MetaMesh** listing shows that the mesh contains 1,295,952 active elements[10]. The 4000 matrix-inversion cycles in **HiPhi** took 363 seconds to reach a relative residual[11] of 8.6×10^{-7}.

To complete the exercise, run **PhiView** and load `sheetgun.hou`. You can experiment with the options for 2D and 3D plots. Figure 4 shows an example, an overview of the levels of $|\mathbf{E}|$ on the focus-electrode surface. The red areas correspond to $|\mathbf{E}| = 16.7$ MV/m.

[10]The term active means that the elements are not part of a fixed-potential region.

[11]The relative residual, a measure of the accuracy of the iterative solution, should be much smaller than unity.

13 3D electrostatic application: getting started

In this chapter we'll advance to an application study, a concept for a capacitive position sensor. Rather than using prepared input files, we'll build the entire solution from scratch. This chapter covers the initial steps in creating a 3D assembly with **Geometer**. Following chapters will cover mesh generation, the finite-element solution and analysis techniques. The example emphasizes the internal parametric models of **Geometer** and **MetaMesh** rather than the `STL` models discussed in Chap. 11.

Figure 52 shows a view of the detector geometry. A dielectric object of known shape and size drops through the assembly along the y direction. A shaped drive electrode with an applied AC voltage creates a time-dependent electric field that varies in the x direction. Changes in the mutual capacitance between the drive electrode and detector are sensed with a bridge network. The function of the detector is to confirm the presence of the object and its approximate position in x. The purpose of our calculation is find numerical values for the relative changes in mutual capacitance with the size and position of the object to gauge the accuracy requirements for the detector circuit.

Figure 53 gives an alternate 3D view of the assembly. For reference, we will use the following dimensions:

> The solution volume covers the region -2.50 cm $\leq x, y \leq$ 2.50 cm and -0.25 cm $\leq z \leq$ 3.00 cm.
>
> A metal case of thickness 0.25 cm encloses all sides of assembly except for a cut to house the detector and the upper boundary in z. We'll discuss the physical meaning of this open boundary in a following article.
>
> The test object is a sphere of radius 0.375 cm.
>
> The detector is a rectangular plate with dimensions $W_x = 1.75$ cm, $W_y = 3.25$ cm and $W_z = 0.25$ cm.
>
> The cutout for the detector has dimensions $W_x = 2.00$ cm and $W_y = 3.50$ cm.
>
> The drive electrode is an extrusion of length $W_y = 4.0$ cm with a cross-section in the z-x plane defined by an outline of lines and arcs.

To start, decide on a working directory for the input/output files. Run the **FPController** and point the *Data folder* to the directory. Run **Geometer** and click *File/New script*. In the opening dialog, enter the values shown in Fig. 54 to define the dimensions of the solution volume. When you click *OK*, **Geometer** automatically creates the first part[12], a box that fills the solution volume. By default, the part is named *SolutionVolume* and is assigned to *Region 1*. The program displays a default orthogonal view in the x-y plane. The program turns off the visibility of the first part because it would obscure the view of additional ones.

[12] see Chap. 11 for the definitions of parts and regions in **Geometer**

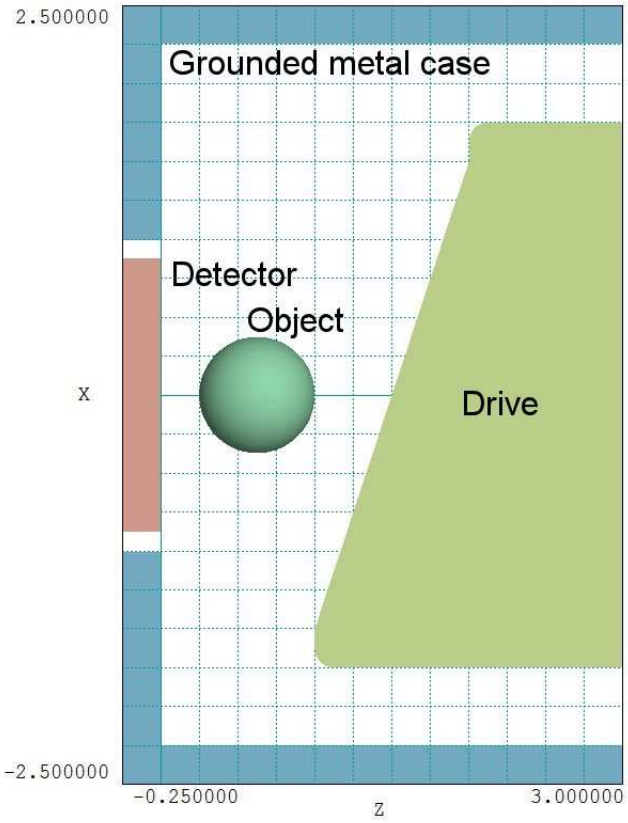


Figure 52: Detector geometry in the plane $y = 0.0$ cm.

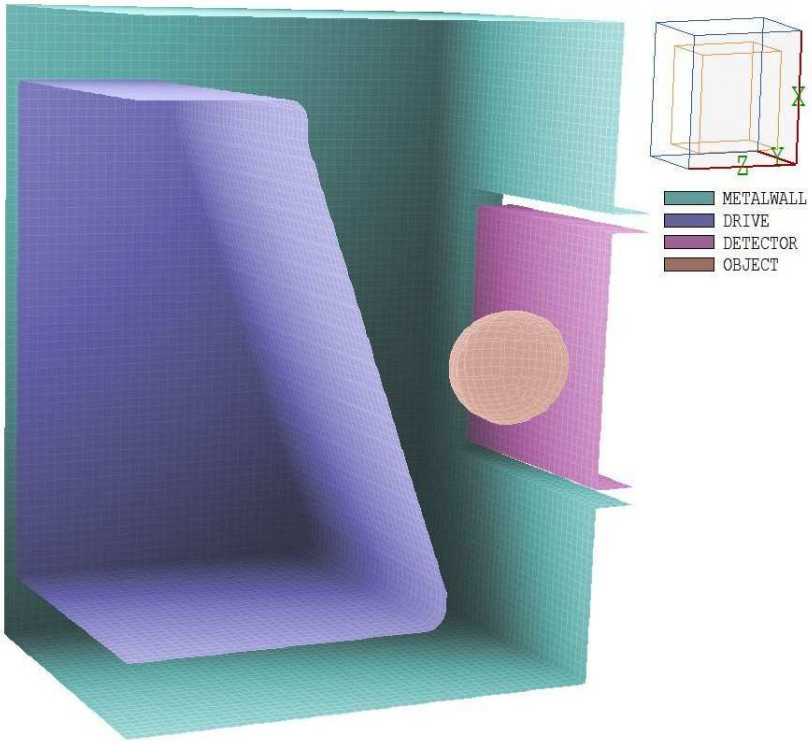


Figure 53: 3D view of the detector assembly.

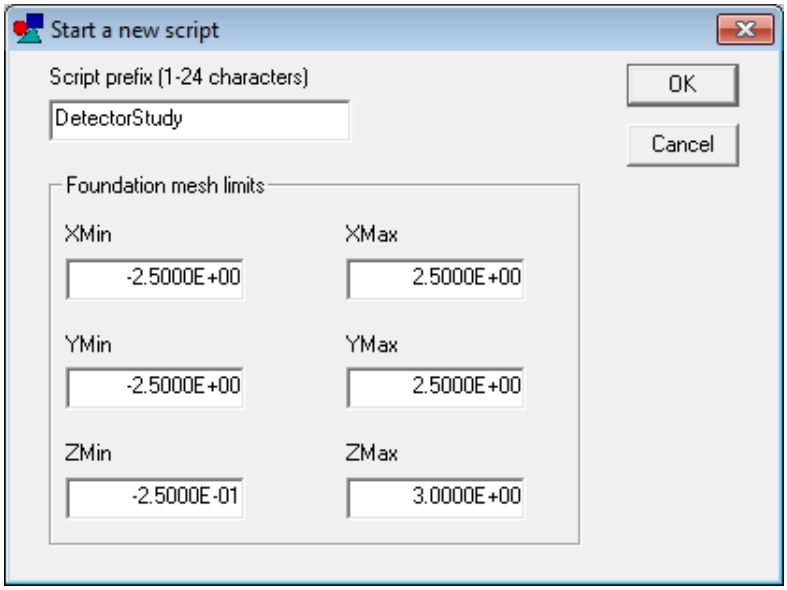


Figure 54: Dialog to start a new assembly in **Geometer**.

At this point, we need to make a strategy decision. We could associate the first part with air and add five parts to build up the metal case wall. On the other hand, it is easier to start with a block of metal that fills the solution volume and then to carve out an internal air volume. We'll take this approach. To start, let's define the physical regions that will be needed for the electrostatic solution and give them names that suggest their functions. Click *Edit/Edit region names* to bring up the dialog of Fig. 55. Replace the default region names with the ones shown and click *OK*.

Click the command *Edit/Edit part*, select *SolutionVolume* (the only one available) and click *OK*. The dialog on the left-hand side of Fig. 56 (left) shows the properties of the parametric model. The part is a *Box* that fills the solution volume. The box length in z is $W_z = 3.25$ cm. The part has been shifted a distance 1.375 cm so that it fills the region -0.25 cm $\leq z \leq$ 3.00 cm. Note that the name of the associated region has been changed to *MetalWall*. Change the default part name to *MetalWall* and then click *OK* to exit.

Next, we shall add a part that carves out the air volume. Choose *Edit/Add part* to open the model parameter dialog. Under *Part type*, choose *Box* from the popup menu. Set the *Part name* to *Air* and pick the *Region Air*. To leave metal walls of thickness 0.25 cm, the box should have lengths $L_x = L_y = 4.50$ cm and $L_z = 3.00$ cm. We need to set $Z_{shift} = 1.50$ cm to fill in the region from 0.0 cm $\leq z \leq$ 3.0 cm. The right-hand side of Fig. 56 shows the final state of the dialog. When you exit, the boundary of the air region appears in the plot. Click *View/Y normal axis* to check that the length and displacement of the box are correct. At this point, it's a good idea save the work. Click *File/Save script* and accept the name `DetectorStudy.MIN`. The text file is in a format recognized as input to **MetaMesh.**

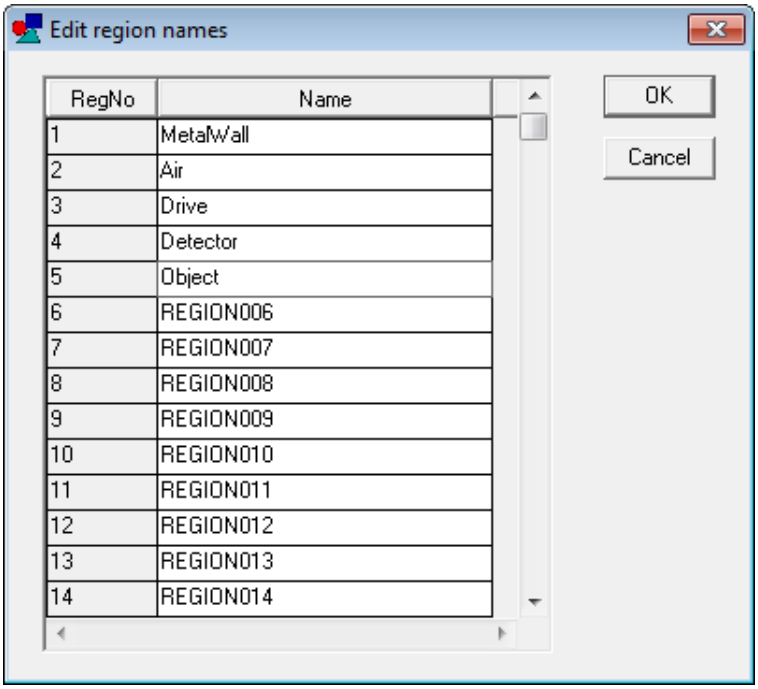


Figure 55: Dialog to define region names.

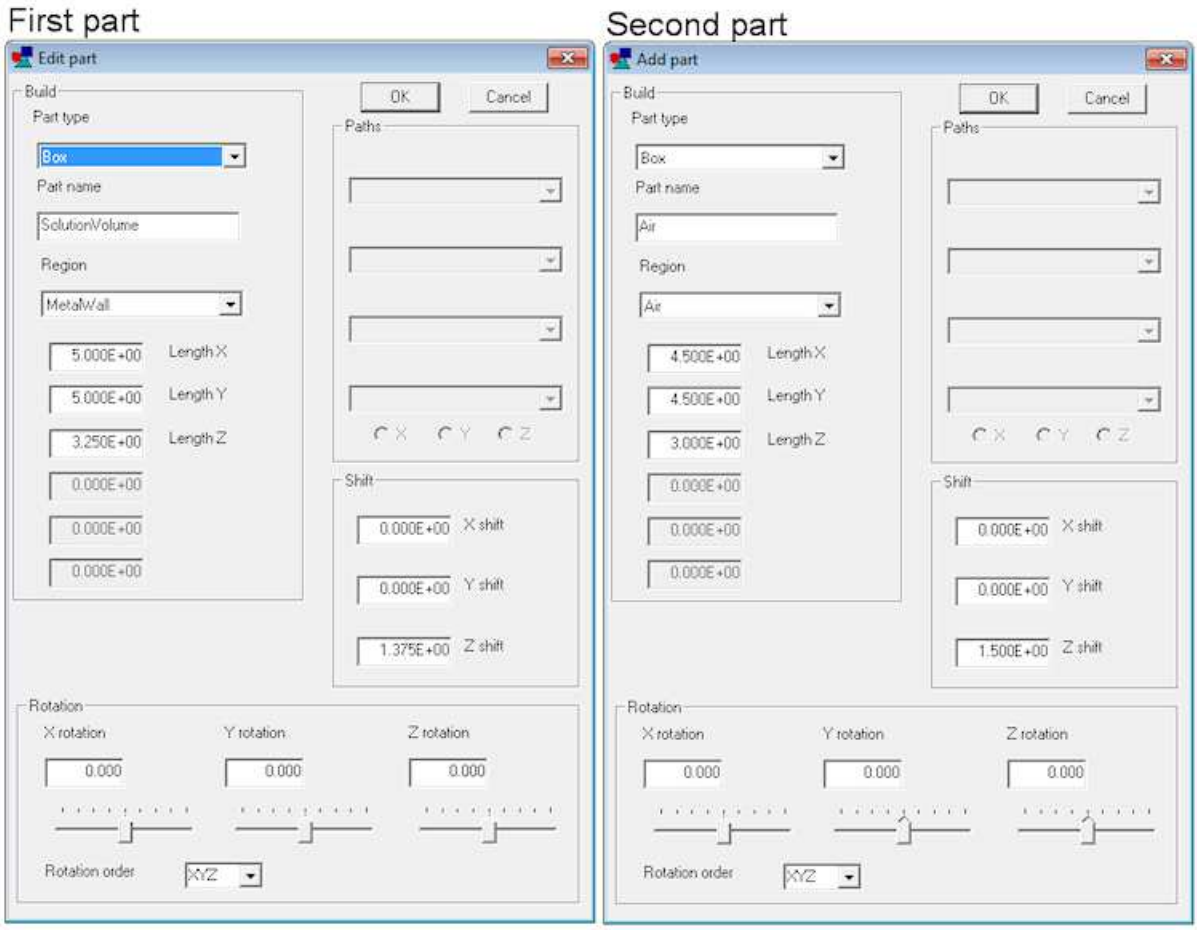


Figure 56: *Edit part* dialogs for the first (metal wall) and second (air volume) parts.

We need to include a slot in the wall to house the detector. Click on *Edit/Add part* and fill in the dialog with the following values

```
Type: Box
Part name: DetectorWell
Region: Air
Lx = 2.00
Ly = 3.50
Lz = 0.25
ZShift = -0.125
```

If everything is correct, the plot looks like Fig. 57. Go ahead and add the detector and the test object using the following parameters

```
Type: Box
Part name: Detector
Region: Detector
Lx = 1.75
Ly = 3.25
Lz = 0.25
ZShift = -0.125

Type: Sphere
Part name: Object
Region: Object
Radius = 0.375
ZShift = 0.625
```

The spherical object is centered at $x = y = 0.0$ cm with center 0.625 cm from the surface of the detector.

Viewing the state of the assembly requires some effort and experience. The surfaces of selected parts are plotted using **OpenGL**. When an encompassing part (like the metal wall) is included, it obscures internal parts. The following options are useful to display internal parts:

Turn off the visibility of encompassing parts.

Use a wireframe display for the encompassing part.

Assign clipping planes to the part display.

The **Geometer** instruction manual gives a complete description of plot controls. To illustrate the current state of the setup, Fig. 58 shows a perspective view of the detector and object regions with the air region shown as a wireframe.

We need only define the drive electrode to complete the assembly. This part is an *extrusion*, a shape that extends a given distance along one direction with an arbitrary shape in the plane normal to that direction. In comparison, a turning is an arbitrary shape rotated about an axis of symmetry. In both cases, the shape is defined by a set of connected line and arc vectors. Extrusions and turnings are highly useful and versatile models that merit their own discussion. We'll cover the topic in the next chapter.

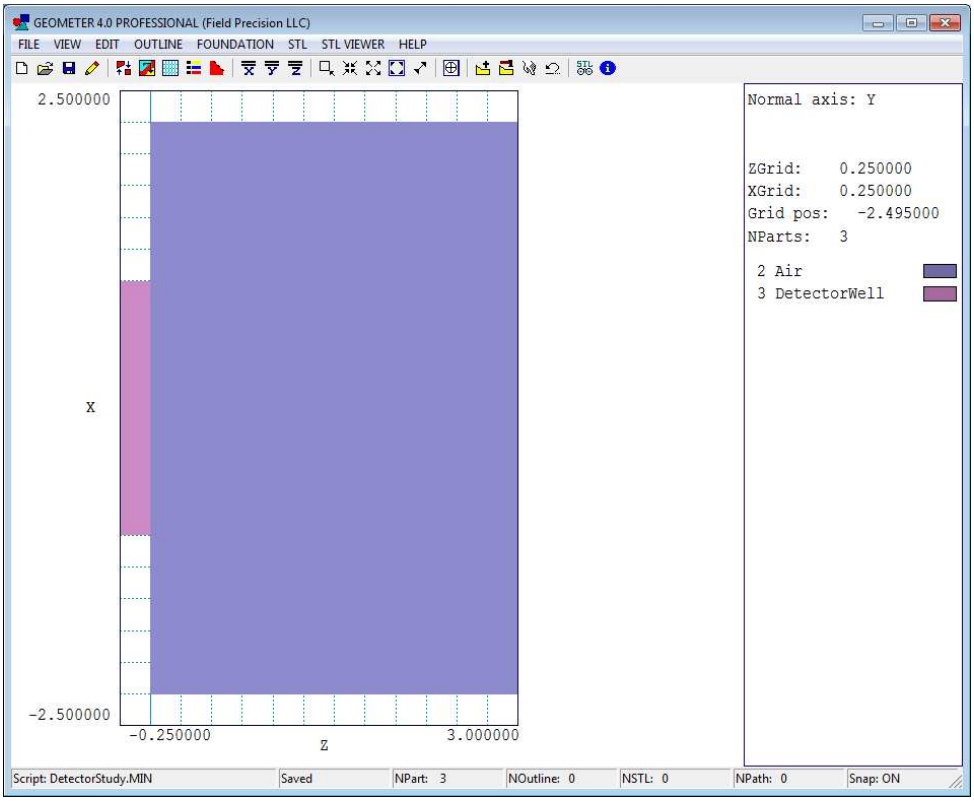


Figure 57: State of the assembly with the metal wall, air volume and detector slot.

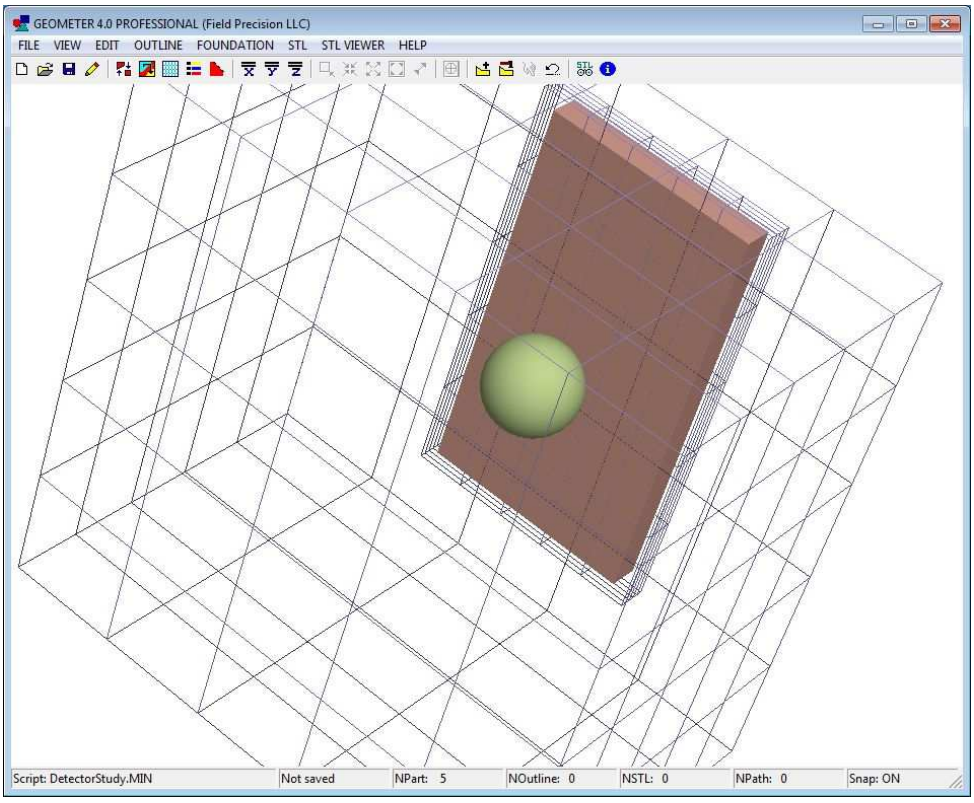


Figure 58: Perspective view of the assembly by regions with the air region shown as a wireframe.

14 3D electrostatic application: extrusions

In this chapter we'll finish building the geometry of the capacitive detector and then generate a mesh. The remaining part is the drive electrode, an *extrusion*. Extrusions are the type of objects that you fabricate with a milling machine. They have arbitrary cross-section shapes that extend a specified distance along one direction.

A set of connected line and arc vectors defines the cross section of an extrusion. We call the set an *outline*. We already encountered outlines when discussing the 2D **Mesh** program in Chap. 3. The outlines in **Geometer** and **MetaMesh** have formats identical to those in **Mesh** – in fact, you can interchange them between the programs. One way to add an outline in **Geometer** is to construct it in the **Mesh** *Drawing Editor* (a full-featured CAD program) and then copy-and-paste the vectors. **Geometer** also has a built-in *Outline Editor*. It has many of the features of the *Drawing Editor*, but is designed for work on one outline at a time. In this article, we'll concentrate on building outlines within **Geometer**.

First, a review of some concepts. Outlines are data sets available within **Geometer** that may (or may not) be used to define the cross section of one or more extrusions or turnings. In other words, outlines in the program exist independently of specific parts[13]. There are three ways to enter outlines in **Geometer** to make them available for part definitions:

Copy-and-paste them from a **Mesh** script (*New outline Text* command).

Load a **MetaMesh** input script (*Load script* command). The outlines of extrusions and turnings are stored and made available to construct additional parts.

Draw them in the *Outline Editor* (*New outline Graphics* command).

We'll concentrate on the third method. For reference, Fig. 59 shows the dimensions of the drive-electrode cross section. Although the following discussion refers to menu commands, there are entries in the toolbar for most of them.

Run **Geometer** and click *File/Load script*. Choose `CapDetector.MIN`, the file that contains the work from the previous chapter (the metal box and detector). Click *Outline* to open the *Outline Editor*. Choose *New outline (Graphics)* to open the dialog of Fig. 60. Supply the name *DriveElectrode*. The extrusion extends in the y direction in the assembly, so for convenience we will create cross-section vectors in the z-x plane. Check the *ExtY* radio button. For the *View limits*, enter the extreme dimensions of the outline as shown. Note that in a right-handed coordinate system normal to y, the z axis corresponds to the horizontal direction and the x axis to the vertical. When you click *OK*, **Geometer** opens the drawing editor with a view large enough to encompass the outline (Fig. 61).

We'll enter the shape of Fig. 59 as a set of lines and then add the two fillets. Outline vectors must connect exactly, so it's essential to use snap mode. By default, the program is set to *Grid snap*. There is an invisible grid of snap points beneath the displayed grid. The status line shows that the current snap grid distances are 0.1. We need to change the intervals so they are appropriate to the dimensions of Fig. 59. Click *Draw/Grid control*. In the dialog, uncheck *Automatic* intervals, set *XGrid* and *YGrid* equal to 0.25 and click *OK*.

[13] **Geometer** can store up to 80 outlines, each with up to 50 line and arc vectors.

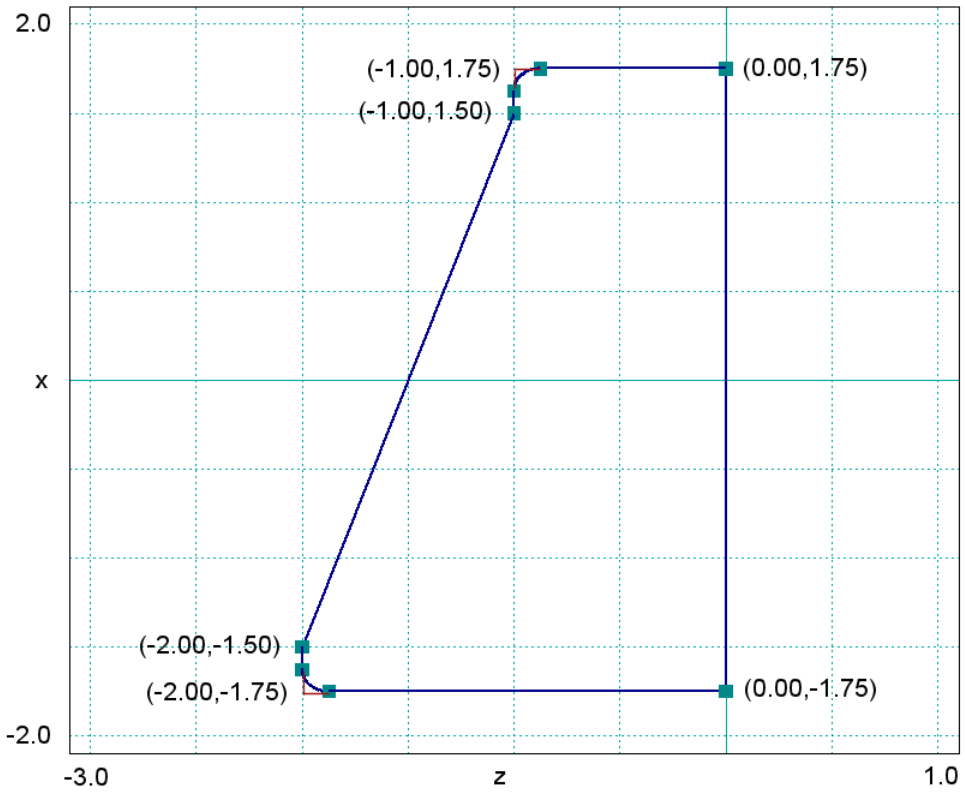


Figure 59: Geometry of the drive electrode cross section.

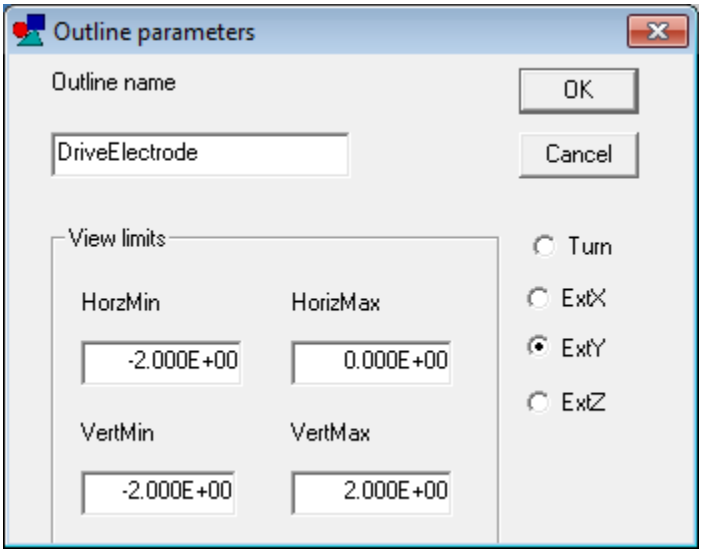


Figure 60: Dialog to start a new outline.

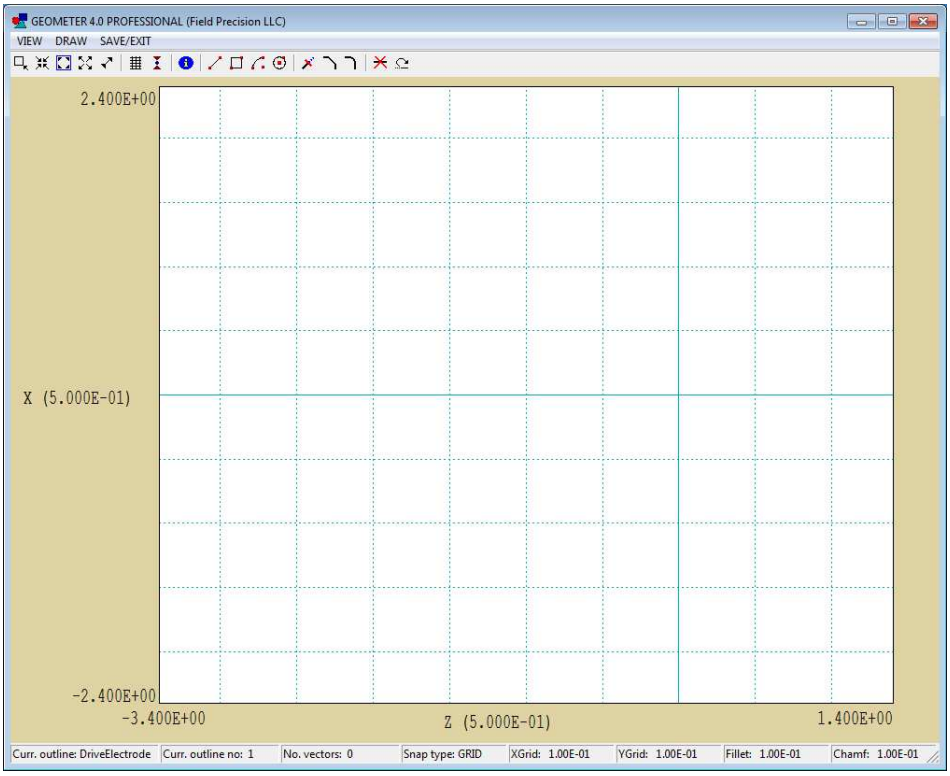


Figure 61: Initial appearance of the *Outline Editor*.

To make a line, pick *Draw/Insert line*. Move the cursor to the correct start position and click the left button. Then move to the end position and click the left button again. You can continue making lines by clicking start and end points. When you are finished, click the right button to exit line entry mode. Figure 62 shows the rough outline. We'll add fillets of radius 0.125 to finish the work. Click *Draw/Fillet/chamfer width* and enter the value 0.125 in the dialog. Then choose *Draw/Fillet*. Move the mouse cursor over one of the lines and click the left button to highlight the vector. Then move over the intersecting line and left-click. **Geometer** completes the fillet, modifying old vectors, adding new ones and reordering the set.

The shape is now complete. Click *Save/Exit*, choose *Save outline* and exit. There is now one outline available to define parts in **Geometer**. Currently, the outline exists only in the program memory and will be lost unless you add it to a part in the **MetaMesh** script. It's a good idea to save the outline as a file that you can import into other **Geometer** sessions. In the main *Outline* menu, click *Save outline* and choose an appropriate name and location. The file is in text format and can be inspected or modified with an editor. Here is the content:

```
* ExtY
L  0.0000E+00   -1.7500E+00  0.0000E+00  1.7500E+00
L  0.0000E+00    1.7500E+00 -8.7500E-01  1.7500E+00
A -8.7500E-01    1.7500E+00 -1.0000E+00  1.6250E+00  -8.7500E-01 1.6250E+00
L -1.0000E+00    1.6250E+00 -1.0000E+00  1.5000E+00
L -1.0000E+00    1.5000E+00 -2.0000E+00 -1.5000E+00
L -2.0000E+00   -1.5000E+00 -2.0000E+00 -1.6250E+00
A -2.0000E+00   -1.6250E+00 -1.8750E+00 -1.7500E+00  -1.8750E+00 -1.6250E+00
L -1.8750E+00   -1.7500E+00  0.0000E+00 -1.7500E+00
END
```

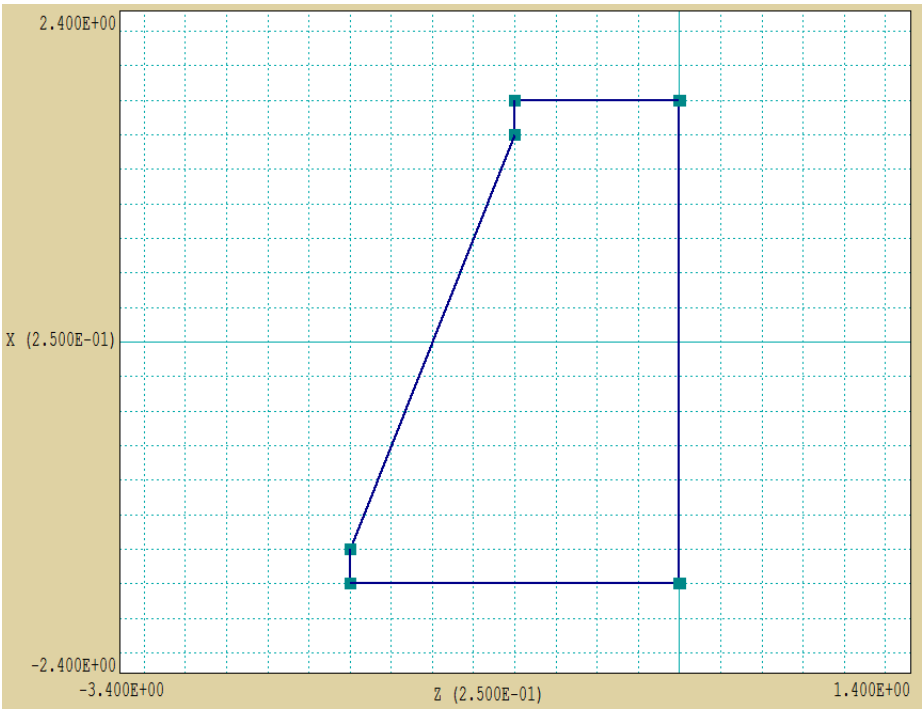


Figure 62: Rough outline before adding fillets.

Exit the *Outline Editor*. To conclude, we'll add the final part incorporating the outline. Pick *Edit/Add part*. In the dialog (Fig. 63), set the *Type* to *Extrusion*. Note that the *Outline* popup menu at the upper-right becomes active. There is only one choice (*DriveElectrode*). Highlight it and press *Enter*. The extrusion axis radio buttons beneath change to *ExtY*. For the model, there is only one fabrication parameter, the *Height*. Set it to 4.0. Finally, the top of the electrode that we drew at $z = 0.00$ cm for convenience should coincide with the top boundary of the solution volume ($z = 3.00$ cm). Accordingly, set $Z_{shift} = 3.00$ cm. When all values have been set to correspond to those of Fig. 63, click *OK*. After inspecting plots to check validity, save your work as the **MetaMesh** input file `CapDetector.MIN`.

Run **MetaMesh** and load the file you just created. To start, let's take a look at it. Choose *File/Edit MIN file* to display the contents. The top section lists the properties of the foundation mesh, the basic elements to construct the geometry. **Geometer** has made default choices for the element sizes. We'll change them so that elements are matched to objects in the assembly. By *matched*, I mean that the initial boundaries between elements are coincident with the boundaries of many of the objects. Change the foundation-mesh specifications to:

```
XMesh
  -2.500   2.500   0.0625
End
YMesh
  -2.500   2.500   0.0625
End
ZMesh
  -0.250   3.000   0.0625
End
```

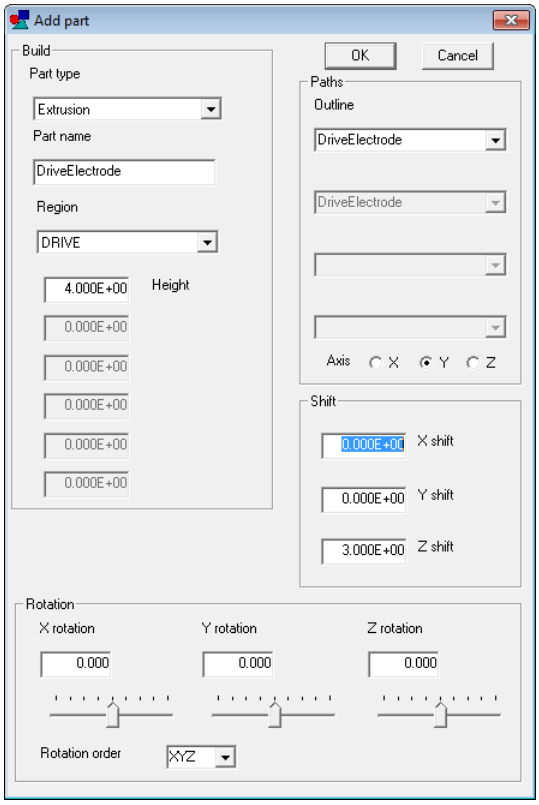


Figure 63: Dialog to define properties of the drive electrode part.

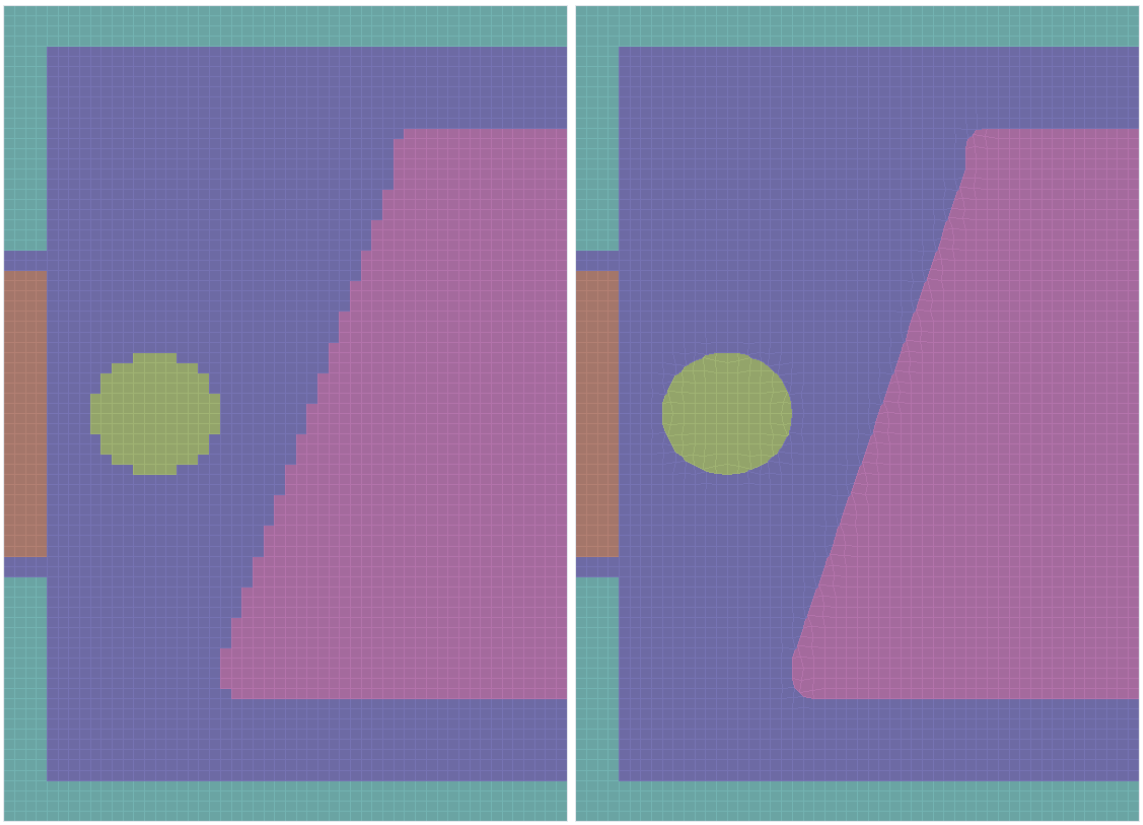


Figure 64: Comparison of meshes without and with region surface fitting.

Save the file and exit the editor. Process the mesh, go to the 2D plot window and choose a plot normal to the y axis. The result looks like the left-hand side of Fig. 64. The metal wall and detector are well represented because of our judicious choice of element sizes. On the other hand, the drive electrode and object have a stair-step shape. To get accurate field interpolations near the surface, we need to tell **MetaMesh** to apply conformal fitting (*i.e.*, shift elements near the surface from boxes to generalized hexahedrons with facets that lie on the object boundaries).

Open the `MIN` file in the text editor again and add *Surface* commands for the *Drive* and *Object* parts as shown in Table 3. The command

```
Surface Region Air
```

instructs the code to identify all elements of the part that share facets with the *Air* region and to change their shape so that they conform to the model surface. Save the file, exit the editor, reload the `MIN` file and process it. The right-hand side of Fig. 64 shows the improved result. Click *File/Save mesh* to create `CapDetector.MDF`.

The task of geometry definition is complete. Modifications to an assembly are generally much easier than the initial creation. You can reload the `MIN` file into **Geometer** and quickly change the shape, position and orientation of parts. You could also edit the `MIN` file directly to move parts around. We'll make use of this feature to determine the variation of mutual capacitance between the drive and detector electrodes as the position of the object changes. In the next chapter, we'll discuss the finite-element solution and useful analysis techniques.

Table 3: *Drive* and *Object* part specifications in `CapDetector.MIN`

```
PART
  Region: Drive
  Name: Drive
  Type: Extrusion Y
    L    0.0000E+00  1.7500E+00 -8.7500E-01  1.7500E+00
    A   -8.7500E-01  1.7500E+00 -1.0000E+00  1.6250E+00 -8.7500E-01  1.6250E+00
    L   -1.0000E+00  1.6250E+00 -1.0000E+00  1.5000E+00
    L   -1.0000E+00  1.5000E+00 -2.0000E+00 -1.5000E+00
    L   -2.0000E+00 -1.5000E+00 -2.0000E+00 -1.6250E+00
    A   -2.0000E+00 -1.6250E+00 -1.8750E+00 -1.7500E+00 -1.8750E+00 -1.6250E+00
    L   -1.8750E+00 -1.7500E+00  0.0000E+00 -1.7500E+00
    L    0.0000E+00 -1.7500E+00  0.0000E+00  1.7500E+00
  End
  Fab:   4.00000E+00
  Shift:   0.00000E+00  0.00000E+00  3.00000E+00
  Surface Region Air
END

PART
  Region: Object
  Name: Object
  Type: Sphere
  Fab:   3.75000E-01
  Shift:   0.00000E+00  0.00000E+00  6.25000E-01
  Surface Region Air
END
```

15 3D electrostatic application: mutual capacitance

In this chapter, we'll conclude the application example by calculating the electric field and discussing two useful techniques:

Determining mutual capacitances in a system with multiple electrodes.

Setting up **HiPhi** solutions for automatic operation in the background.

Run **HiPhi**, click *Setup* and load `CapDetector.MDF`. We discussed the *Setup* dialog in Chap. 12. Its function is to create the **HiPhi** input script, `CapDetector.HIN`, that controls the field calculation. Here, we'll concentrate on the contents of the file (Fig. 65). The values at the top are control parameters (mostly defaults). The *DUnit* command states that the coordinates in the **MetaMesh** file are given in cm. The remaining commands specify the identity and physical properties of regions. The value of the drive voltage, 1.0 V, is a convenient choice for capacitance calculations. Note that there are two entries for the detected *Object* (*Region 5*), one of which is commented out. To begin, we will calculate the field without the object, equivalent to setting $\epsilon_r = 1.0$. Click *Run* in **HiPhi** and pick `CapDetector.HIN` to generate the solution file `CapDetector.HOU`.

Run **PhiView** and load the solution file. To begin, we'll make a plot of $|\mathbf{E}|$ along x at the position of the object (y = 0.000 cm, z = 0.625 cm) to identify the region of linear field variation. Click *Slice plots* and choose *Slice normal to y*. Click *Plot control/Plot style* and set the style to *Element*. Click *Plot control/Plot quantity* and choose $|\mathbf{E}|$. The result is the top plot of Fig. 66. Make sure *Analysis/Scan plot quantity* is set to E_z. Choose the command *Analysis/Line scan*. Although you could specify the endpoints of the scan with the mouse, in this case we want to be sure the line is at the horizontal position of the object center. Press the *F1* key to type values for the start position: $z = 0.625$ cm, $x = -2.25$ cm. Click *OK* and press *F1* again to type the end position: $z = 0.625$ cm, $x = 2.25$ cm. **PhiView** calculates values and displays the graph shown at the bottom of Fig. 66. The region of approximately linear variation covers the range -1.25 cm $\leq x \leq$ 1.25 cm.

We will proceed with the calculation of mutual capacitance between the drive and detector electrodes, advancing to a higher level of automatic analysis at each stage. For the discussions, the three electrodes in the solution have been numbered in Fig. 66. We'll start with single operations that you control in the interactive environment of **PhiView**, appropriate for one or a few calculations.

The mutual capacitance between electrodes 1 and 2 is given by $C_{12} = Q_2/(V_1 - V_2)$. The charge induced on the detector is given by the surface integral:

$$Q_2 = \int\int dS_2 \; \epsilon_r \epsilon_0 E_{norm}, \tag{9}$$

where E_{norm} is the electric field normal to the surface and ϵ_r is the relative dielectric constant of the medium surrounding the detector. (In this case, $\epsilon_r = 1.0$.) The **PhiView** configuration file `phiview_dielectric.cfg` contains this definition:

```
SURFACE
  Charge = &Ex $Epsi0 * &Q[2] *;&Ey $Epsi0 * &Q[2] *;&Ez $Epsi0 * &Q[2] *
END
```

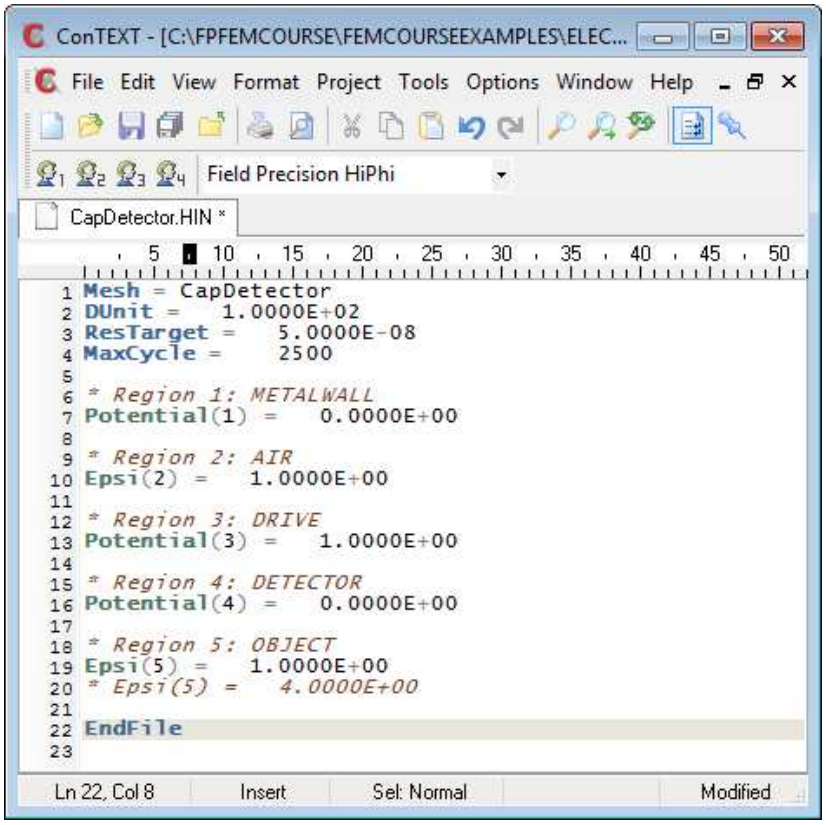


Figure 65: File `CapDetector.HIN` viewed in **ConText**.

Here, `&Ex` is a calculated component of electric field and `&Q[2]` is the relative dielectric constant. **PhiView** takes automatic surface integrals over regions by evaluating the quantities of Eq. 9 just outside the facets of the region boundary. The result is the quantity Q_2 in coulombs. Because we picked an applied voltage of 1.0 V, the result can also be interpreted directly as the mutual capacitance.

Return to the main **PhiView** menu and click *Analysis/Surface integral* to bring up the dialog of Fig. 67. Because **PhiView** can take integrals over the combined surface of multiple regions, we need to specify which regions are inside the surface and which ones are outside. In this case, the detector is inside and all others are outside as shown. When you click *OK*, the program prompts you to create a file `CapDetector.DAT` to record the results. This is a useful feature of numerical work; otherwise, it would be necessary to copy the results from the screen.

To see the results, click *File/Close data file* and then *File/Edit data file*. Here is the entry:

```
Region status
RegNo   Status    Name
==================================0
 1  External   METALWALL
 2  External   AIR
 3  External   DRIVE
 4  Internal   DETECTOR
 5  External   OBJECT
Surface area of region set (m2):   8.187500E-04
Charge:  -3.015445E-13
```

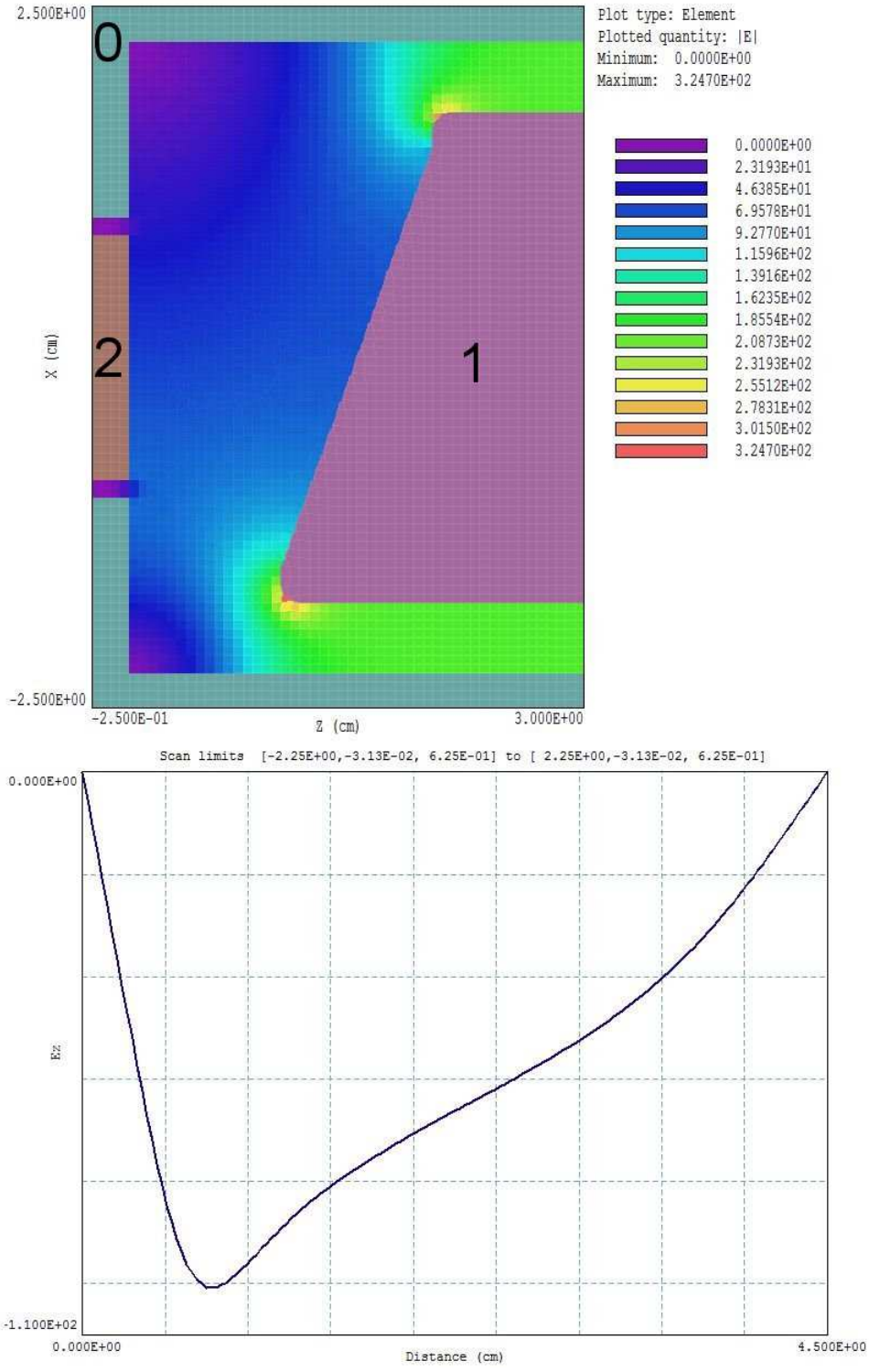


Figure 66: Top: Variation of $|\mathbf{E}|$ in the plane $y = 0.0$ cm. Bottom: Scan of $|\mathbf{E}|$ along x at $y = 0.0$ cm, $z = 0.625$ cm.

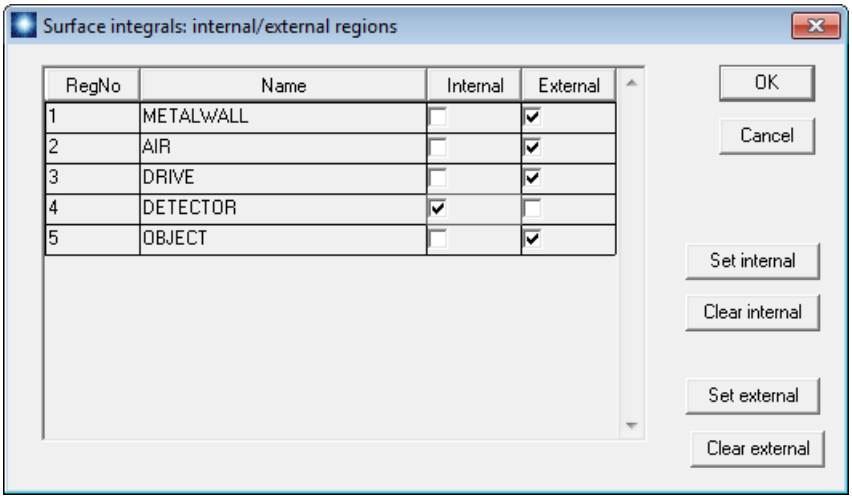


Figure 67: Dialog to control the PhiView surface integral.

The mutual capacitance without the object is 0.30154 pF.

When possible, it's a good idea to make an analytic estimate to make sure we haven't made a fundamental error in the numerical setup. The mutual capacitance is approximately

$$C_{12} = \frac{\epsilon_0 A_2}{D_{12}}. \tag{10}$$

The average distance between the drive and detector is $D_{12} = 0.015$ m. The area of the upper detector surface is $A_2 = (0.0175)(0.0325) = 5.7 \times 10^{-4}$ m^2. Substituting in Eq. 10, the estimated capacitance is $C_{12} = 0.335$ pF, close to the code result.

Open the file `CapDetector.HIN` with an editor and change the object properties to

```
* Region 5: OBJECT
* Epsi(5) =   1.0000E+00
Epsi(5) =   4.0000E+00
```

Recalculate the field with **HiPhi** and find the detector surface integral with **PhiView**. The result with the object centered at $x = 0.0$ cm is $C_{12} = 0.3162745$ pF. The object raises the mutual capacitance by about 4.9%.

To complete the calculation, we want to find C_{12} with the object at positions x = -0.50, -0.25, 0.00, 0.25 and 0.50 cm. To analyze each solution, we must load `CapDetector.HOU` into **PhiView** and call up the surface integral. Rather than perform the same operations every time, we can define an automatic analysis sequence. In the *PhiView* main menu, click *File operations/Create script*. Supply the file prefix *CapDetector*. The program opens the internal editor with a template of available analysis script operations. Type in information so that the script looks like this:

```
INPUT CapDetector.HOU
OUTPUT CapDetector.DAT Append
SURFACEINT 4 -1 -2 -3 -5
ENDFILE
```

Click *Save* to create the file `Surface.SCR` and exit the editor. The script specifies the following operations

Load the **HiPhi** solution `CapDetector.HOU`.

Write information to a data file `CapDetector.DAT`. The *Append* directive signals that **PhiView** should add data to the file if it exists.

Perform a surface analysis. The numbers following the command indicate that that *Region 4* (the detector) should be inside and all others outside.

To add a value of mutual capacitance to the output file, simply click *File operations/Run script* in the main menu of **PhiView** and choose `CapDetector.SCR`.

A manual run to find C_{12} for an object position requires the following activities:

1. Edit `CapDetector.MIN`. To move the object to $x = -0.5$ cm, change the *Shift* command in the object section to `Shift:  -0.500 0.000 0.0625`.
2. Run **MetaMesh**, click *File/Load MIN file* and choose `CapDetector.MIN`. Then click *Process* followed by *File/Save mesh*.
3. Run **HiPhi**, click *Run/Start run* and choose `CapDetector.HIN`.
4. Run **PhiView** and start the analysis script `CapDetector.SCR`.

There are many redundant operations. To start, let's automate the second, third and fourth activities with a Windows batch file. I'll assume that you are running the programs from **FPController** and that the *Data folder* is set to the working directory. In **AMaze**, click the *Create task* button to bring up the dialog of Fig. 68. Fill in the *Task prefix* as *CapDetector*. Under *Action*, click the arrow to activate the popup menu. In contains a list of the 3D programs and common batch operations. Pick **MetaMesh**. Then, click on the cell under *File in* and click the *Select file* button to show a list of **MetaMesh** input files available in the working directory. Choose `CapDetector.MIN`. The output file of the activity is always `CapDetector.MDF`, so we do not need to specify a value. Go to the next *Action* cell and pick **HiPhi**. For *File in*, choose `CapDetector.HIN`. Set **PhiView** for the next action with `CapDetector.SCR` as the input file. When you click *OK*, **AMaze** saves the setup as `CapDetector.BAT`, a standard batch file. Here is the content:

```
START /B /WAIT C:\fieldp_pro\amaze\metamesh.exe C:\...\CapDetector
START /B /WAIT C:\fieldp_pro\amaze\hiphi.exe C:\...\CapDetector
START /B /WAIT C:\fieldp_pro\amaze\phiview.exe C:\...\CapDetector
START /B /WAIT C:\fieldp_pro\amaze\NOTIFY.EXE
IF EXIST CapDetector.ACTIVE ERASE CapDetector.ACTIVE
```

The batch file runs each program in the background with the specified input file, waiting until an operation is complete before moving to the next one. The fourth line calls for an audio signal to mark the end of the sequence. The fifth line deletes a temporary file used to keep track of which tasks are active.

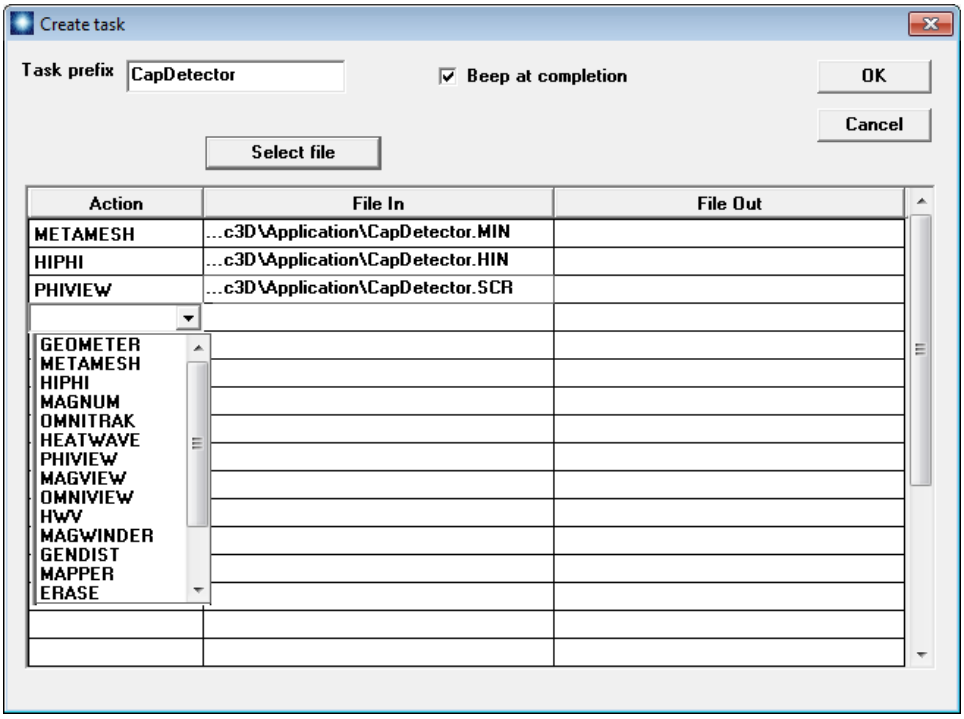


Figure 68: **FPController** dialog to create a task.

At this stage, we have reduced the work to two user operations for each position of the object:

Edit the *Shift* operation in `CapDetector.MDF` to change the object position along x.

In **FPController**, click the *Run task* button and choose `CapDetector.BAT` from the list of available tasks. In response, the mesh is regenerated, the field solution is updated and the surface analysis result is added to the data file `CapDetector.DAT`.

The end result is a list of C_{12} values at the different positions.

To conclude, we can eliminate the first task to automate the entire process. Here. we will specify the changing object position within the calling batch file. We need to make two changes. First, the *Part* section for the object in the **MetaMesh** input file should be modified to:

```
PART
  Region: Object
  Name: Object
  Type: Sphere
  Fab:   3.75000E-01
  Shift:   %1   0.00000E+00  6.25000E-01
  Surface Region Air
END
```

Note that the x position in the *Shift* operation is represented by a variable. The value to be used is given as the first command line parameter when calling **MetaMesh**. The file `CapDetector.BAT` should be edited to look like this:

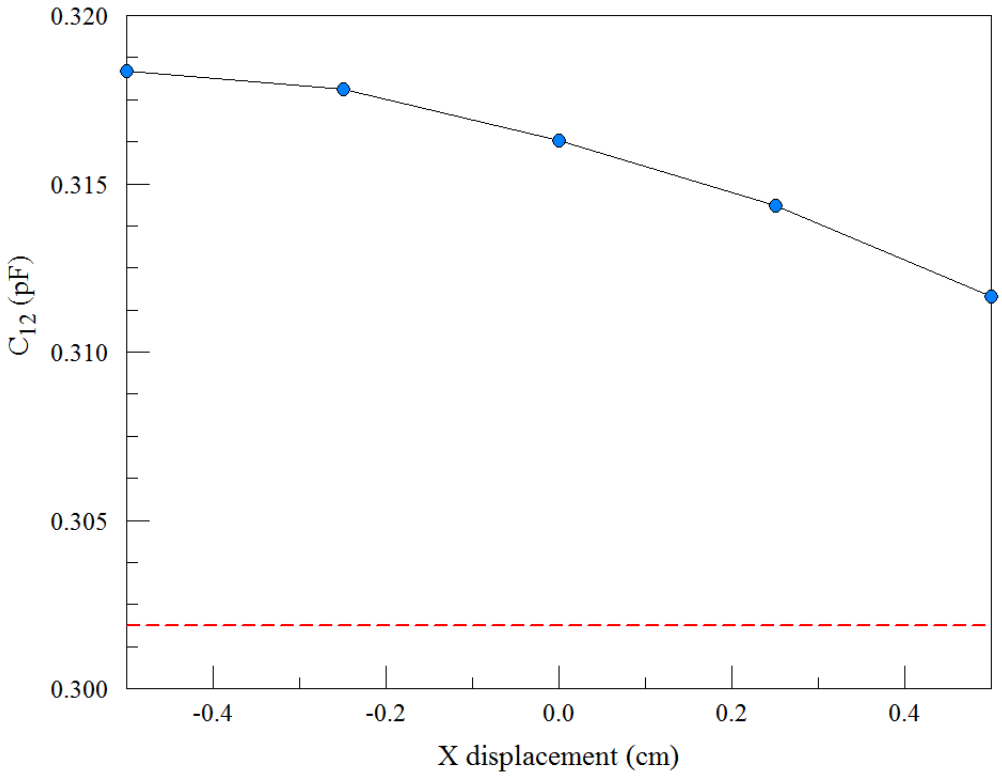


Figure 69: Variation of C_{12} with the displacement of the dielectric object. The dashed line shows the value with no object.

```
START /B /WAIT C:\fieldp_pro\amaze\metamesh.exe C:\...\CapDetector -0.50
START /B /WAIT C:\fieldp_pro\amaze\hiphi.exe C:\...\CapDetector
START /B /WAIT C:\fieldp_pro\amaze\phiview.exe C:\...\CapDetector
START /B /WAIT C:\fieldp_pro\amaze\metamesh.exe C:\...\CapDetector -0.25
START /B /WAIT C:\fieldp_pro\amaze\hiphi.exe C:\...\CapDetector
START /B /WAIT C:\fieldp_pro\amaze\phiview.exe C:\...\CapDetector
  ...
START /B /WAIT C:\fieldp_pro\amaze\metamesh.exe C:\...\CapDetector 0.50
START /B /WAIT C:\fieldp_pro\amaze\hiphi.exe C:\...\CapDetector
START /B /WAIT C:\fieldp_pro\amaze\phiview.exe C:\...\CapDetector

START /B /WAIT C:\fieldp_pro\amaze\NOTIFY.EXE
IF EXIST CapDetector.ACTIVE ERASE CapDetector.ACTIVE
```

The task generates the entire data set. Figure 69 shows a plot of the results. The dashed line is the mutual capacitance with no object. The relative variation of C_{12} with position of the object is about $\pm 1.1\%$. The conclusions are that the application would require a sensitive detector and that small variations in the size or properties of the dielectric object could overwhelm the measurement.

An alternate way to find mutual capacitance is the energy method described in the **HiPhi** manual. In this case, it is necessary to make three solutions for each configuration with different combinations of electrode voltages. The capacitance values C_{12}, C_{10} and C_{20} are determined by comparing values of the total field energy. The procedure requires 15 solutions – here, the automation features of **HiPhi** provide a significant benefit. In the calculations, the electrode voltages as well as the object position are set by command line parameters.

16 3D magnetic fields: defining coil currents

In this chapter, we'll start the final topic of the book: 3D magnetic field solutions. The previous chapters on electrostatics covered many common techniques for 3D solutions, particularly mesh generation. Here, we'll focus on the differences between **Magnum** and **HiPhi** calculations that arise from the unique characteristics of magnetic fields:

The vector driving terms for magnetic solutions require more intensive data input.

Two completely different solution techniques apply, depending on whether ferromagnetic materials are present.

Magnetic materials have more complex properties.

The phrase *driving terms* refers to the information we supply to a finite-element program to create fields. For example, the drivers of electrostatic solutions are differences in the electrostatic potential between electrodes and space-charge. In other words, we apply a fixed potential to the nodes of a region of the solution volume or define space-charge density ρ over elements of a region. In both cases, the specified quantities are scalar. For either drive type, we must employ the finite-element technique to determine the self-consistent charge density on electrode surfaces.

There are two possible drives in magnetic-field solutions:

Currents in coils.

Material currents in permanent magnets.

Coil currents constitute a unique challenge in 3D solutions. To review, we saw in Chap. 6 that it was easy to define currents in 2D solutions. In cylindrical geometries, current flowed only in θ and in planar solutions the current flow was along z. In other words, current was a scalar quantity assigned to regions that represented the cross sections of coils. In 3D solutions, the drive current is a vector quantity that can flow anywhere. If you think about devices like electric motors, it's clear that drive coils may be highly complex assemblies. The implication is that the definition of drive coils is a significant task in 3D solutions, on a par with mesh generation. In consequence, the **Magnum** package contains a utility **MagWinder** specifically for coil windings.

Next, consider options for magnetic solution procedures. Suppose we have a set of drive coils in a system with no ferromagnetic materials. (In other words, all objects in the solution space have relative magnetic permeability $\mu_0 = 1.0$.) In this case, there are no unknown currents. The field at any point can be determined by a Biot-Savart integral[14] over the drive current elements. It is not necessary to build a conformal mesh of regions or even to apply the finite-element method. On the other hand, a mesh and a finite-element solution are essential when the solution space contains permanent magnets or magnetic materials ($\mu_r \neq 1.0$). Here, material currents that are not known *a priori* have a significant effect on the fields. The implication is that a 3D magnetic field program should be capable of two different types of calculations.

[14]Section 9.1 of the book **Finite-element Methods for Electromagnetics** reviews the Biot-Savart law.

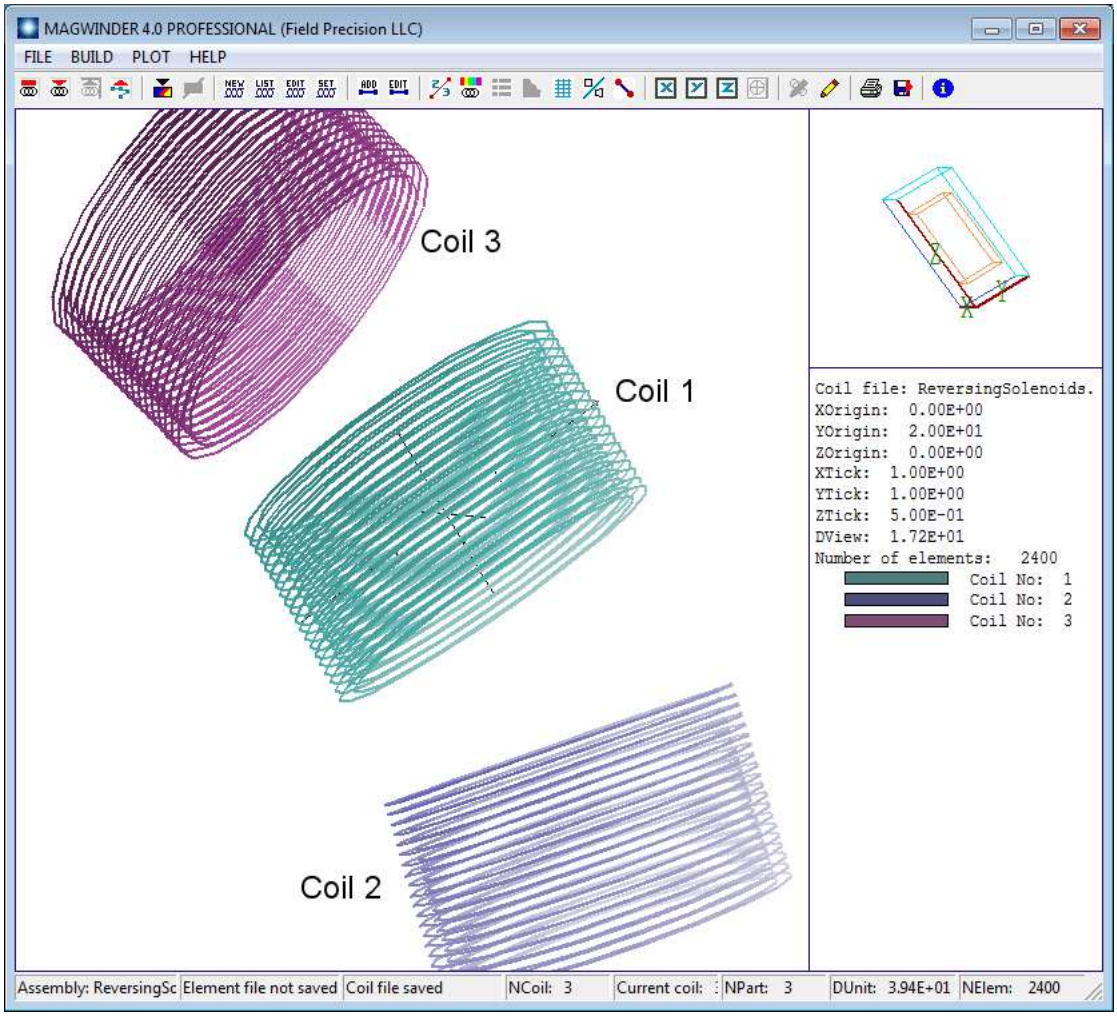


Figure 70: Demonstration coil assembly displayed in the **MagWinder** working environment.

Finally, let's address the nature of materials. In electrostatic solutions, dielectrics have a moderate influence on the total fields and preserve their properties over the practical range of field intensity. An isotropic dielectric can be characterized by a single value of relative dielectric constant ϵ_r (usually in the range < 100) that changes little all the way to breakdown. In contrast, values of the relative magnetic permeability for iron and other ferromagnetic materials may exceed 10,000 at low field levels. In other words, these materials sustain large surface currents and have a strong effect on the total field. The challenge follows from the fact that the materials may become saturated at field levels encountered in practical systems. Here, the value of μ_r depends on the local level of magnetic flux density. A magnetic field program must be capable of handling nonlinear solutions.

In the coming articles, we'll work through two application examples. The first demonstrates the free-space solution type (*i.e,* no magnetic materials). We'll investigate the fields generated by a heater coil at the emission surface of a thermionic cathode. The second application, the holding force of a latching solenoid, demonstrates a full finite-element solution. It involves drive coils, permanent magnets and high-permeability steel. Both applications require work with **MagWinder**. In the remainder of this chapter, we'll get familiar with the program.

Figure 70 shows the coil assembly we will construct, part of a transport system for a high-

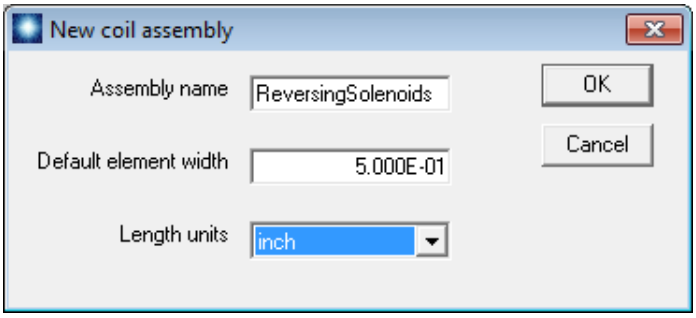


Figure 71: Dialog to start a new drive coil assembly.

current electron beam. The full assembly is an array of solenoid lenses with alternating polarity to focus the beam through a 90° bend. The bend is centered at the origin of the y-z plane at $x = 0.0$". The bend radius from the origin to the solenoid centers is 20.0". We will build three of the coils with total amp-turns $I = 20$ kA, -20 kA and -20 kA. We'll start with the middle coil. Click the **MagWinder** button in **FPController** [15]. Click *File/New coil assembly*. In the dialog of Fig. 71, supply the name *ReversingSolenoids* and set the length units to inches.

Some explanation is required to understand the quantity *Default element width*. For the Biot-Savart integral, wires are divided into a large number of short elements. In the present setup, we will divide each current loop in the solenoid into 20 azimuthal pieces. Although the elements are plotted as thin wire segments, they are not treated that way in **Magnum**. Infinitely-thin wires would give rise to discontinuous, diverging values for the applied field. Instead, each element is treated as a cylinder of uniform current density with a diameter equal to its length. Therefore, the wires of Fig. 70 act more like current density layers than a set of discrete filaments.

Next we need to add coils to the assembly. We'll start with the middle coil. The central field points along z. The center is displaced 20.0" along y. Click *Build/New coil* to display the dialog of Fig. 72. Fill out the values as shown. Then click *OK*. A coil is data structure that may contain one or more physical parts. In order to see a plot, we need to add a *part*. Click *Build/Add part* to open the dialog of Fig. 73. **MagWinder** includes several predefined models – the solenoid is one of the most useful. Note that the active fields and labels change when you choose the model. The values shown in Fig. 73 designate that the solenoid has inner radius 2.5", outer radius 3.0" and a length 3.0". The radial thickness is represented by two layers. There are 20 sets of circular coils along the axial direction for a total of 40. Each circle is divided into 20 parts. Therefore, **MagWinder** will create 800 current elements to represent the solenoid. Note that the positions and rotations of parts are taken relative to the encompassing coil. We'll leave the quantities at the bottom of the dialog at their default values. For the other solenoids, we will apply shifts and rotations to the encompassing coils. Click *OK* when you are finished entering values. Figure 74 shows the **MagWinder** display of the first coil[16].

[15] The **MagWinder** button becomes active when **Magnum** is installed.

[16] The three coils of the assembly each contain one part (a solenoid). An example of a coil with multiple parts is a solenoid with wire leads.

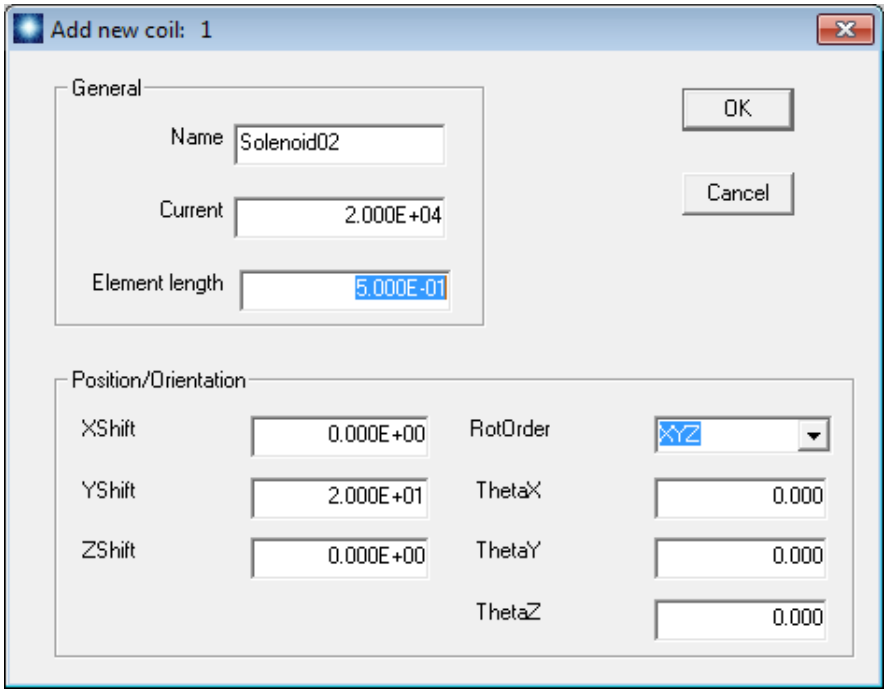


Figure 72: Dialog to add a coil to the assembly.

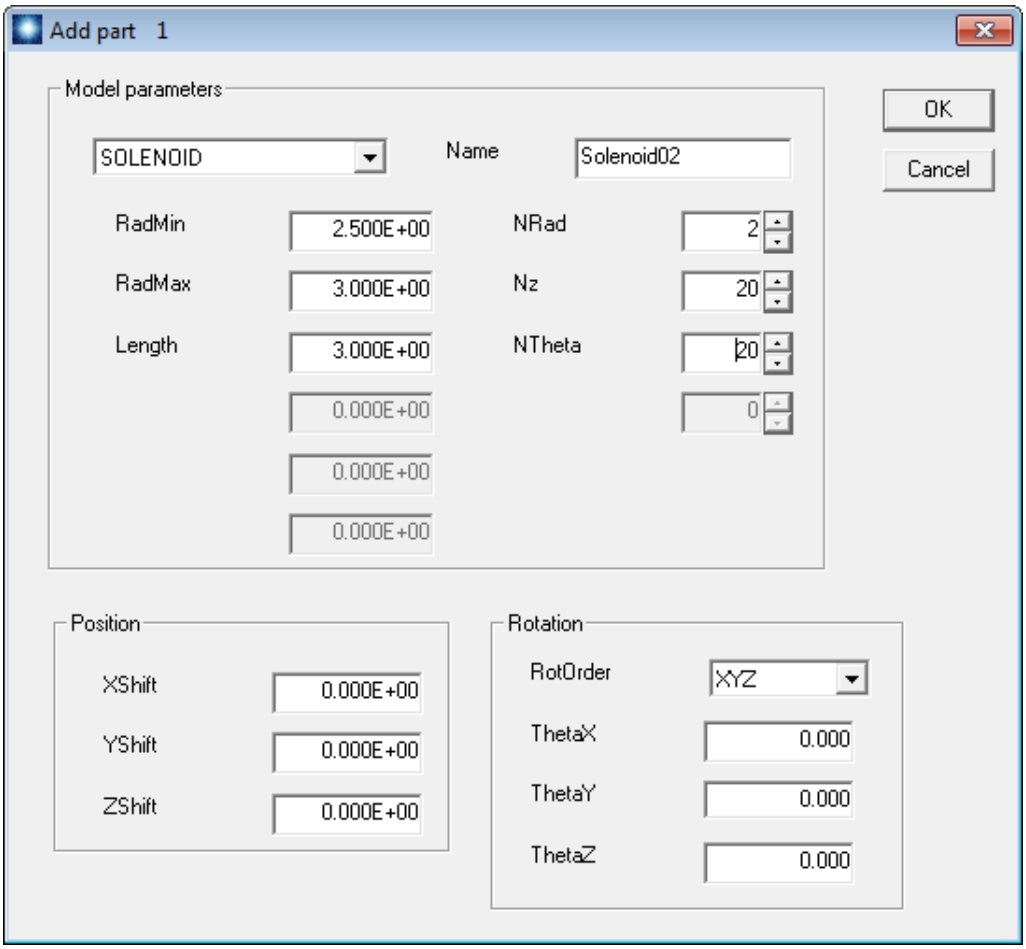


Figure 73: Dialog to add a solenoid part to the present coil.

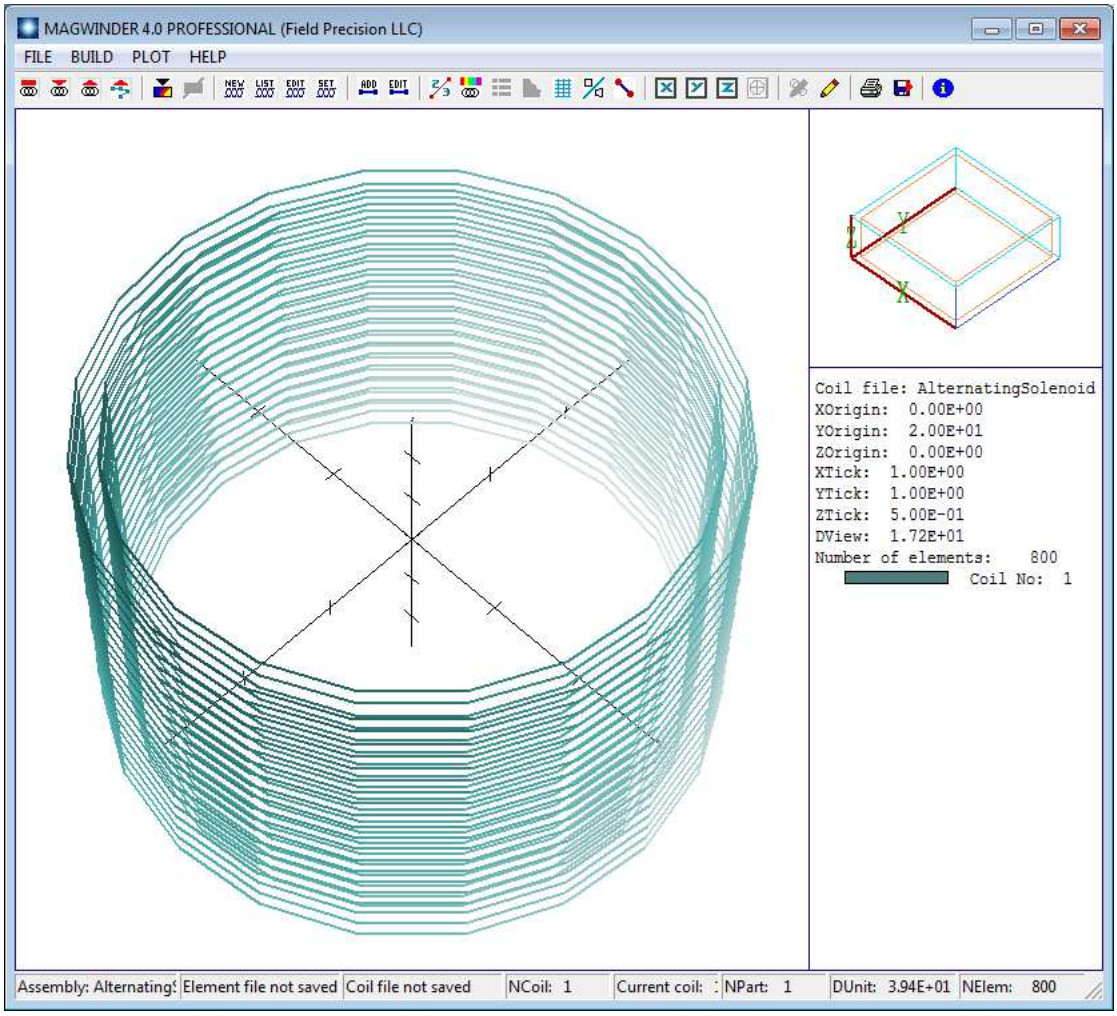


Figure 74: **MagWinder** display with one coil added.

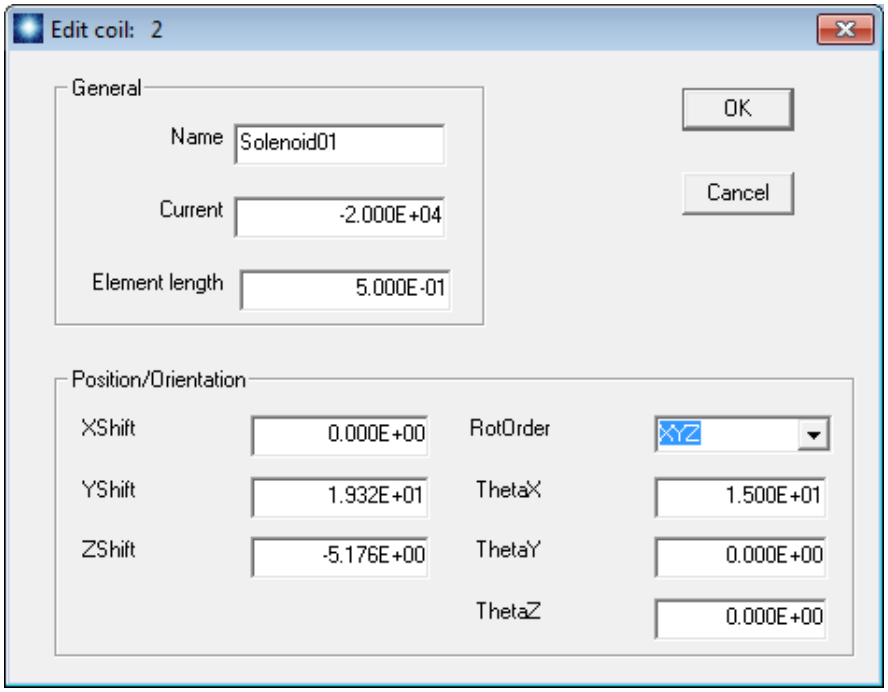


Figure 75: Dialog to add a second coil to the assembly.

The second coil is identical to first except that it is rotated -15° about the x axis at the origin of the y-z plane and has a current of -20 kA. Click *File/New coil assembly*. Fill in the dialog with the values shown in Fig. 75. For reference, the displacement in z is $20.0\sin(-15^o)$ and in y is $20.0\cos(-15^o)$. A rotation of $+15^o$ about the x axis aligns the solenoid with the beam axis. Rather than redefine the solenoid part in an *Add part* dialog, we will copy and paste the part from the first coil. Click *Build/Set current coil* and pick the first coil. Then click *Build/Copy part* and pick the solenoid. Use *Build/Set current coil* to return to the second coil and click *Build/Paste part*. When you click *OK*, the plot shows both coils. The total number of elements is 1600. Use a similar procedure to create a third coil rotated $+15^o$ about the y-$z$origin.

Assuming the assembly looks like Fig. 1, we need to save the work. There are two types of output files:

> The coil-definition-file (`CDF`) is a symbolic representation of the coil geometry. This file may be reloaded and modified.
>
> The winding file (`WND`) is a list of current elements, input for **Magnum**. A `WND` file can always be regenerated from the `CDF` file, but the `CDF` file cannot be inferred from information in the `WND` file.

Both files are in text format. Click *File/Save coil file* to make the file `ReversingSolenoids.CDF`. You can view it with the internal program editor. By now, the conventions of the script formats should be familiar. It it easy to recognize the values you entered via the interactive dialogs. Finally, click `File/Save element file` to generate *ReversingSolenoids.WND*. Again, this well-formatted file can be inspected with an editor. Here is an excerpt:

```
Magnum Current Element File (Field Precision, Albuquerque NM)
NCoil:   3
NElem:   2400

Coil     I
 No     (A)
 ===============
 1  2.0000E+04
 2 -2.0000E+04
 3 -2.0000E+04

Coil   XStart      YStart      ZStart       XEnd        YEnd        ZEnd         I
No     (m)         (m)         (m)          (m)         (m)         (m)        (A)
================================================================================
 1  6.6675E-02  5.0800E-01 -3.6195E-02  6.3412E-02  5.2860E-01 -3.6195E-02  5.0000E+02
 1  6.3412E-02  5.2860E-01 -3.6195E-02  5.3941E-02  5.4719E-01 -3.6195E-02  5.0000E+02
 1  5.3941E-02  5.4719E-01 -3.6195E-02  3.9191E-02  5.6194E-01 -3.6195E-02  5.0000E+02
 ...
```

There are 2400 element data lines. Dimensions have been converted to meters for input to **Magnum**. There are 40 current loops per solenoid, so each loop carries $20000/40 = 500$ A.

In the next chapter, we'll concentrate on the free-space mode of **Magnum**. We'll calculate the field for this assembly, and then follow the application example for the cathode heater.

17 3D magnetic fields: free-space calculations

In this chapter we'll continue the topic of 3D magnetic fields by discussing **Magnum** operation in the *FreeSpace* mode. The mode applies to calculations in an infinite space without magnetic materials. In other words, all objects in the solution space have $\mu_r = 1.0$. To start, let's determine the field generated by the set of three solenoid coils that we built in the previous chapter. The end result of that work was the file `ReversingSolenoid.WND` that contained a set of current elements to approximate the coil set.

The first step in the field solution is to create a mesh. Wait a minute – why do we need a mesh? The previous chapter emphasized that free space calculations used Biot-Savart integrals and had nothing to do with finite-element methods. Here's the reason. An integral over the full set of current elements involves considerable computational work and gives the magnetic flux density $[B_x, B_y, B_z]$ at a single point in space. With this approach, it could take several minutes to create a high-resolution plot of field quantities in a plane. Particle tracking, which involves field calculations at thousands of points along the trajectory, would be impossible. It is much more efficient to apply integrals to calculate field values at a given set of points in space (*i.e.* a mesh), and then to use interpolations to find field values at intervening points. In other words, we do the Biot-Savart integrals once to calculate **B** values at mesh points, and then we can use the values to make any plots and trajectory calculations we want. Another advantage of recording values on a mesh is that all the **Magnum** plotting and interpolation routines developed for finite-element calculations can be applied directly.

Fortunately, it takes only a minute or two to make a mesh for a **Magnum** free-space calculation. All we need is an appropriate set of interpolation points. It is not necessary to make a detailed conformal mesh to represent physical objects[17]. Run **Geometer** and choose *File/New script*. We'll create an interpolation mesh that encompasses the three coils[18]. Supply the name *ReversingSolenoid* and the following dimensions for the solution volume: $-5.0 \leq x \leq 5.0$, $14.0 \leq y \leq 24.0$, $-10.0 \leq z \leq 10.0$. When you click *OK*, **Geometer** sets up the mesh with one default region and part that covers the full volume. Go to the *Foundation* menu and change the element sizes along each axis to 0.10. Then, save the mesh. Run **MetaMesh** and process `ReversingSolenoid.MIN` to create `ReversingSolenoid.MDF`.

Run **Magnum**, click *Setup* and choose `ReversingSolenoid.MDF`. Figure 76 shows the setup dialog with appropriate settings. Note that the dialog fields associated with finite-element solutions (including the material properties at the bottom) become inactive when you click the *Free space* radio button. Save the information as `ReversingSolenoid.GIN`. We now have the three files necessary for the calculation:

[17] There is considerable flexibility in creating meshes for free-space calculations. You could employ variable resolution for high accuracy in a region. You could even use a conformal mesh if you wanted to represent object boundaries in plots.

[18] It is important to realize that the size and location of the interpolation mesh has no effect on the field values. You could create a small mesh if you wanted detailed field values within one of the coils.

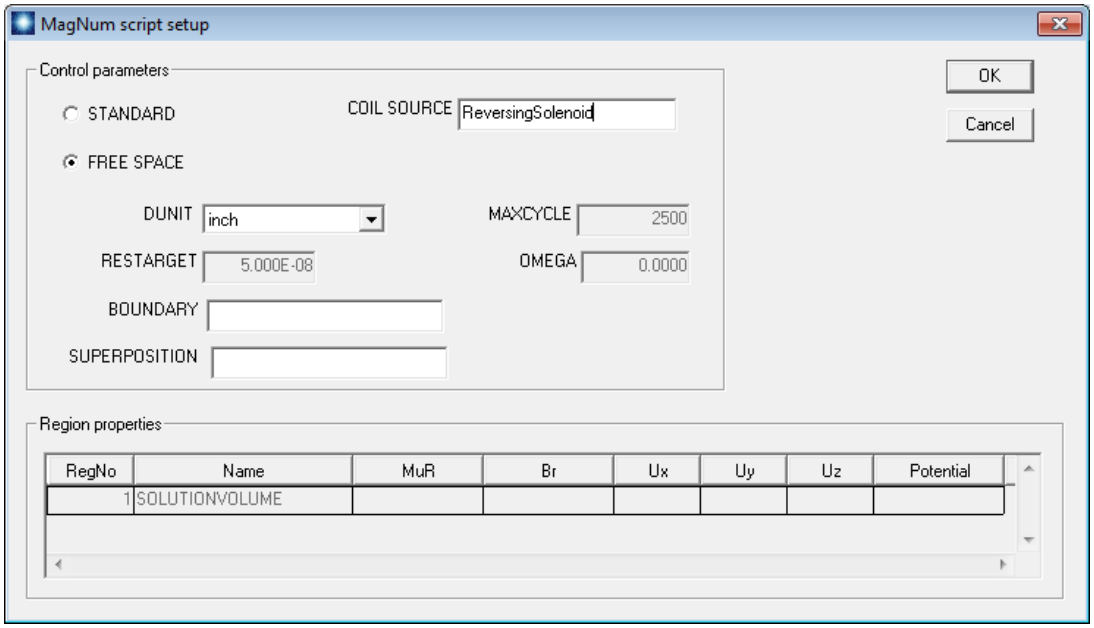


Figure 76: Magnum input dialog for a free-space calculation.

`ReversingSolenoid.CDF`: current elements for the Biot-Savart integral.

`ReversingSolenoid.MDF`: interpolation points to calculate $[B_x, B_y, B_z]$.

`ReversingSolenoid.GIN`: a short control script to let **Magnum** know what it's supposed to do.

In the main **Magnum** menu, click *Run* to create the solution file `ReversingSolenoid.GOU`.

To analyze the results, run **MagView**, click *File/Load solution file* and choose the solution. Go to *Slice plots* and click *Slice normal to X*. Zoom in to the create the plot of Fig. 77, showing the distribution of $|\mathbf{B}|$ in the plane $x = 0.0$". **MagView** has plot and analysis capabilities similar to those of **PhiView**. For example, we can make a quick check of the magnitude and direction of $\mathbf{B}$ with the vector probe. To activate it, click *Vector tools/Vector probe* and then move the mouse cursor into the plot region. The probe, shown in Fig. 77, points along $\mathbf{B}$. Values of the cursor position and $|\mathbf{B}|$ are listed in the status bar at the bottom.

To conclude this chapter, we'll run through an application example with prepared input files (`CathodeHeater.CDF`, `CathodeHeater.MIN` and `CathodeHeater.GIN`). The goal is to find the magnetic flux density on the surface of a thermionic cathode generated by its heater coil. Figure 78 shows the heater geometry, counter-wound helices with connections and leads. Arrows have been added to show the direction of current. The drive coils are defined in the file `CathodeHeater.CDF`, listed in Table 4.

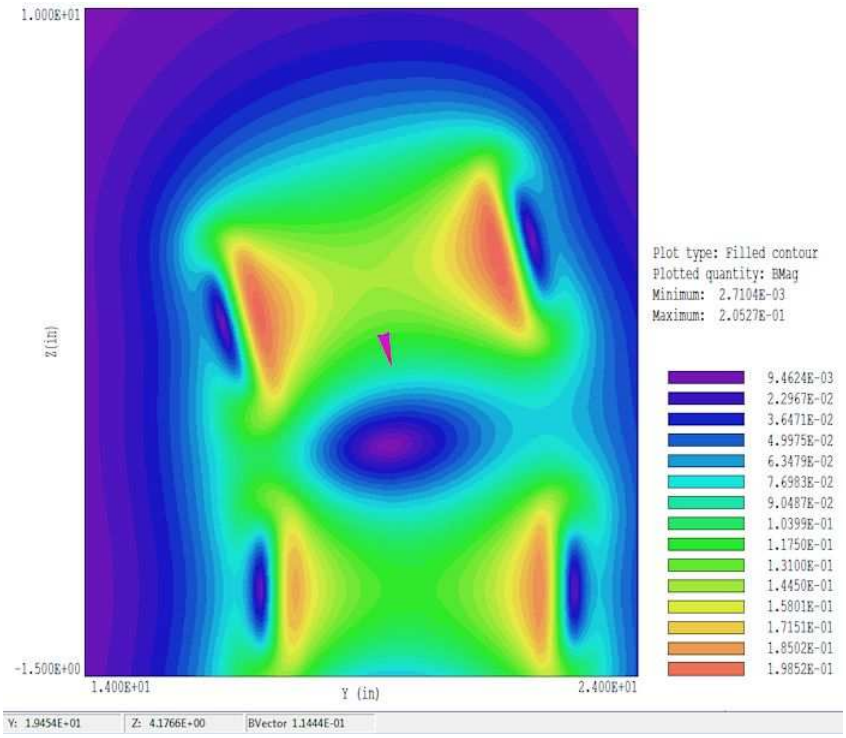


Figure 77: Reversing solenoid example, plot of $|\mathbf{B}|$ in the plane $x = 0.0$", showing the vector probe.

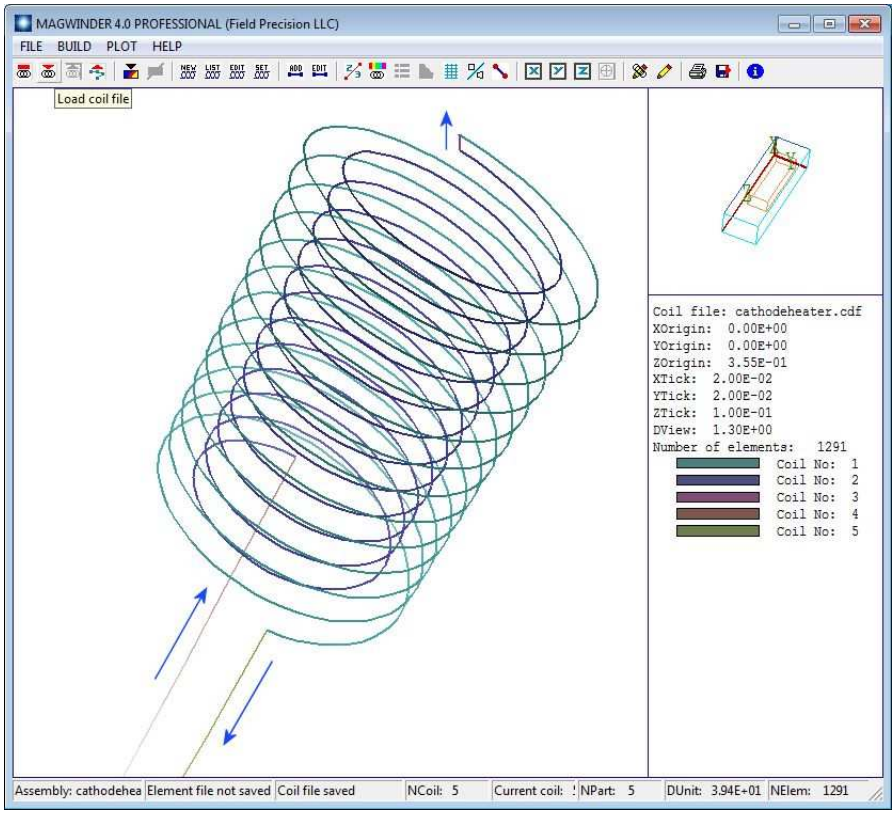


Figure 78: Geometry of the heater coil, arrows added to show the direction of the current.

Table 4: File `CathodeHeater.CDF`

```
GLOBAL
  DUnit = 39.37
  Ds = 0.010
END
COIL 1
  Current = 1.0
* Outer helical coil, half turn extra
  Part
    Type = Helix
    Fab = 0.095 0.060 0.310 0.020
  End
* Connection near surface, inner to outer
  Part
    Type = Line
    Fab = 0.070 0.000 0.060  0.095  0.000  0.060
  End
* Input lead
  Part
    Type = Line
    Fab = 0.070 0.000 0.650  0.070 0.000  0.300
  End
* Output lead
  Part
    Type = Line
    Fab = -0.095 0.000 0.310  -0.095 0.000  0.650
  End
END
* Inner helical coil (input, current in -z direction)
COIL  2
  Current = -1.0
  Part
    Type = Helix
    Fab = 0.070 0.060 0.300 0.020
  End
END
ENDFILE
```

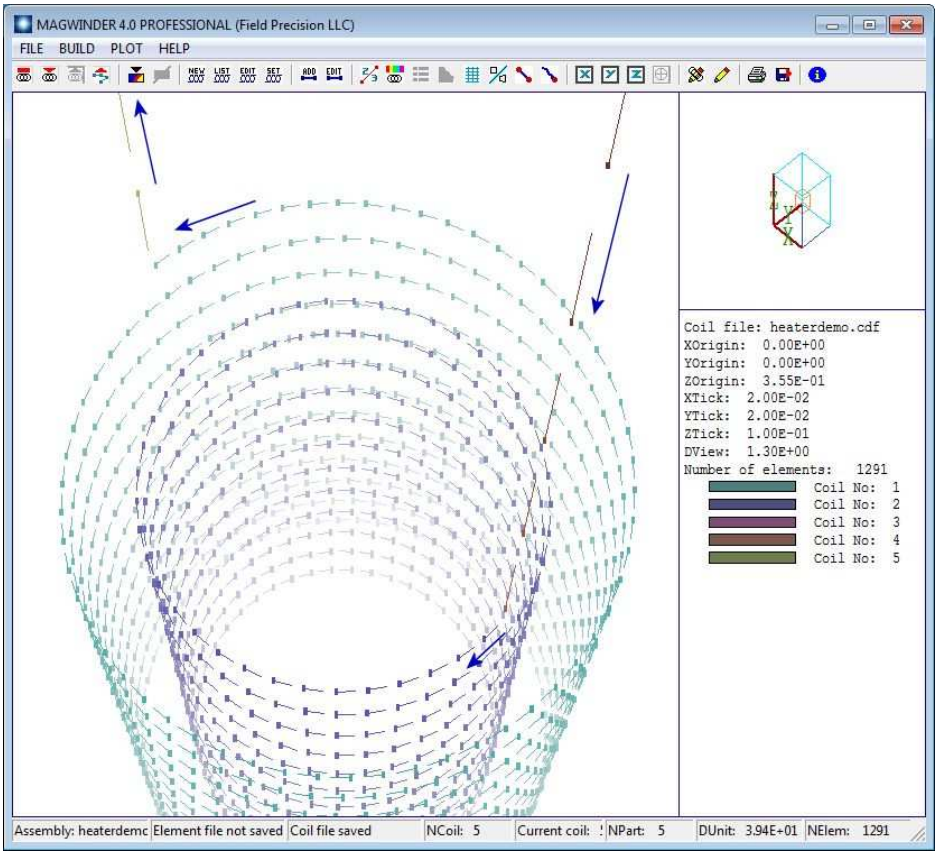


Figure 79: Heater coil, sperm plot option to show the direction of current flow in the elements.

The assembly contains two coils. The first coil, with current 1.0 A, consists of four parts. Three of the parts are straight wires, defined by their start and end points in the assembly space: the input and output leads and the connection between the helical coils. Current always flows from start to end. The *Helix* model requires four fabrication parameters:

```
Radius (0.095")
ZStart (0.060")
ZEnd (0.310")
Pitch, or distance between the turns (0.020")
```

The number of turns equals $|Z_{end} - Z_{start}|/Pitch$. The cathode surface is at z = 0.00", so the distance to the closest part of the coil is 0.060". Helices always have positive rotation proceeding from Z_{start} to Z_{end}. To make counter-wound coils, the return helix is contained in its own coil structure with current -1.0 A. For physically-correct results, it's important that different parts connect to make a continuous coil, and that currents flow in the correct direction. To check validity, you can use the *Plot/Sperm plot* option in either the 2D or 3D views. Figure 79 shows the result. The elements swim in the direction of the current. In particular, the helix currents are correct with respect to the leads and to each other. Move the example files to a working directory, set the *Data folder* in **Amaze** and run **MagWinder**. Load the coil-definition file and click *File/Save element file* to create `CathodeHeater.WND`.

The file `CathodeHeater.MIN` defines a network of points in space where values of **B** are calculated from the current elements and recorded. It has the content:

```
GLOBAL
  RegName 1 Air
  RegName 2 Assembly
  XMesh
    -0.150 0.150 0.005
  End
  YMesh
    -0.150 0.150 0.005
  End
  ZMesh
    -0.100 0.400 0.005
  End
END
PART
  Type Box
  Name SolutionVolume
  Region Air
  Fab 0.3 0.3 0.800
END
PART
  Type Cylinder
  Name CathodeOutline
  Region Assembly
  Fab 0.125 0.400
  Shift 0.000 0.000 0.200
  Surface Region Air Edge 0.95
END
ENDFILE
```

The commands in the *Global* section create a solution volume that encloses the heater coil and cathode surface. The solution volume part fills the entire solution space. Note that there is an additional region that represents the cylindrical cathode volume. The conformal region has no effect on the calculation of applied fields – it is added to include a reference outline of the cathode in plots. Run **MetaMesh**, process the mesh and save the file `CathodeHeater.MDF`.

The file to control the **Magnum** calculation, `CathodeHeater.GIN`, is quite simple:

```
SOLTYPE Free
MESH CathodeHeater
SOURCE CathodeHeater
DUNIT 39.37
ENDFILE
```

The actions are to load `CathodeHeater.WND` and `CathodeHeater.MDF` and to interpret the dimensions in inches. Run **Magnum** and generate the solution, then run **MagView** to analyze the results. Figure 80 shows a filled-contour plot of $|\mathbf{B}|$ in the plane $y = 0.0$". As expected, the magnetic flux is concentrated between the helices. The plot illustrates two special features of **MagView** slice plots:

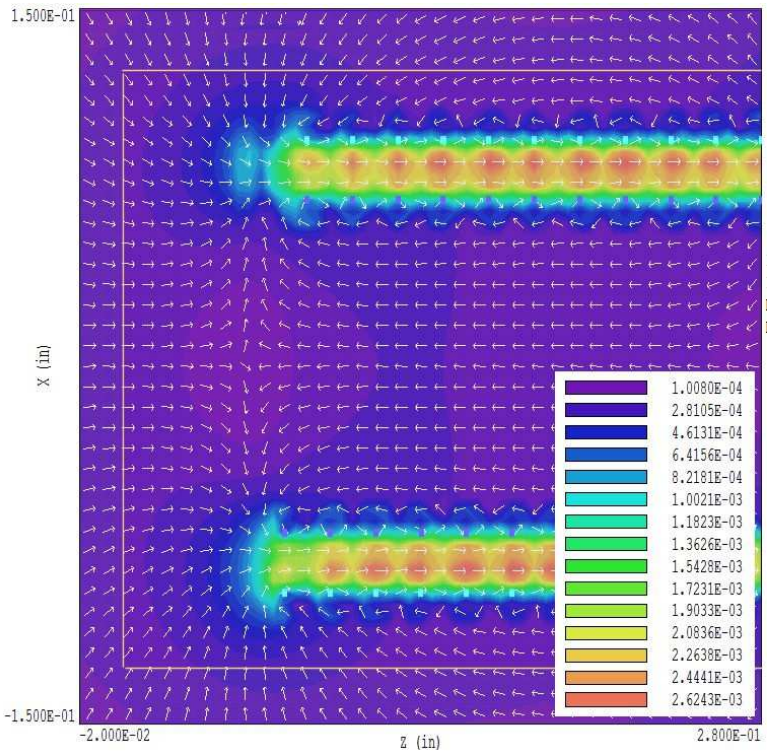


Figure 80: Heater coil, variation of $|\mathbf{B}|$ in the plane $y = 0.0$". The slice plot shows the intersection points of the helical coils and arrows to designate the direction of $\mathbf{B}$.

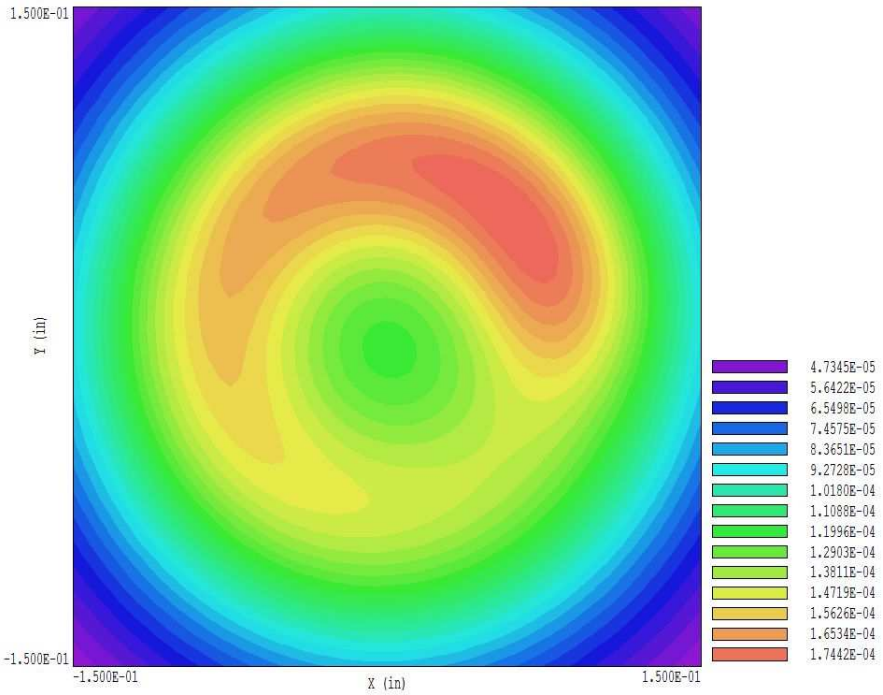


Figure 81: Heater coil, variation of $|\mathbf{B}|$ at the cathode surface.

The drive coils may be superimposed on field plots. The command *File/Load coils* was used to load `CathodeHeater.WND`. The intersections of the coil with the plane $y = 0.0$" are shown as cyan and violet rectangles in the plot.

Arrows showing the direction of $\mathbf{B}$ were added with the command *Vector tools/Vector arrow plot*.

The cathode boundary is marked by yellow lines. The field at the surface approximates a dipole variation. Finally, Fig. 81 shows a plot of $|\mathbf{B}|$ at the cathode surface. The field from the heater configuration approaches 2 Gauss, about 8 times higher than the earth's field. It would be worthwhile to check alternate heater geometries.

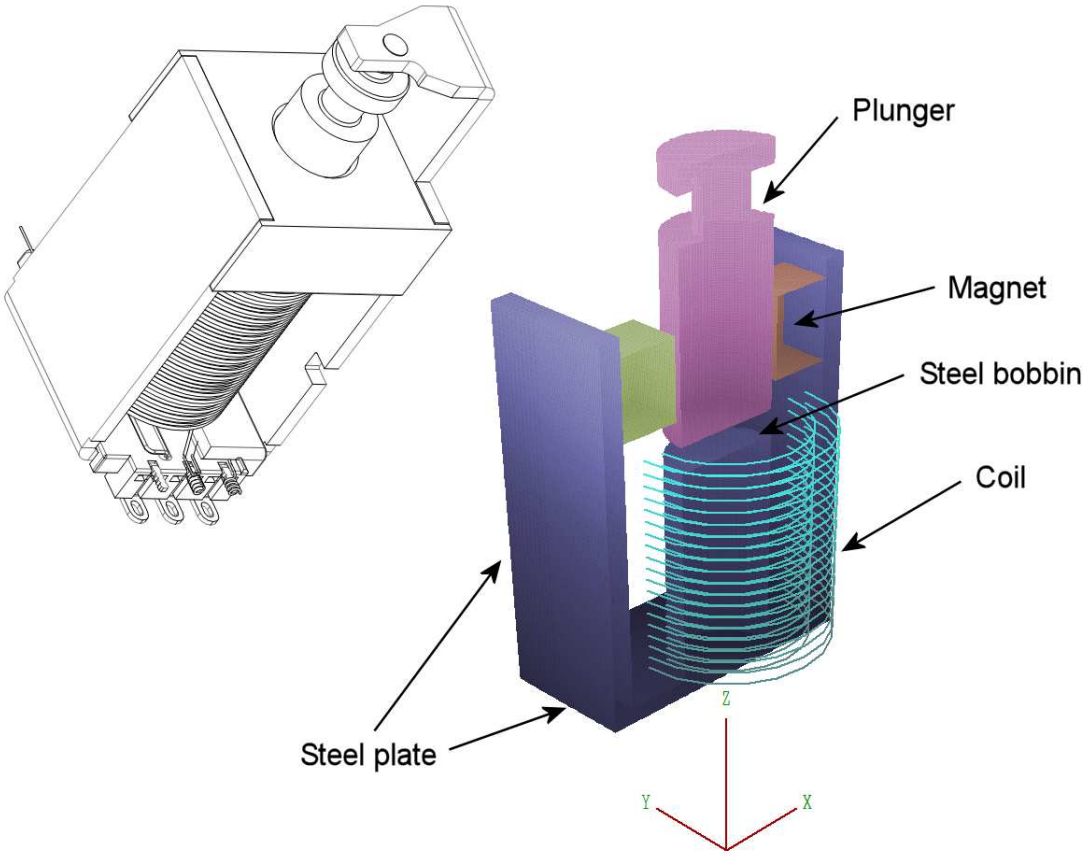


Figure 82: Latching solenoid assembly – drawing and three-dimensional mesh. The parts are displayed in the space $y \geq 0.0$ mm and the coil in $y \leq 0.0$ mm.

18 3D magnetic fields: iron and permanent magnets

For the final calculation of the course, we'll characterize the forces in a latching solenoid. This solution exercises the full finite-element capabilities of **Magnum** and provides an opportunity to use the force-calculation capabilities of **MagView**. In preparation, copy the files `Latching.CDF`, `Latching.MIN` and `Latching.GIN` to a working directory and set the *Data folder* of **FPController**. The three input files have the same functions as the ones we encountered in the previous chapter.

Figure 82 shows a drawing of the assembly along with the mesh created by **MetaMesh**. The neodymium-iron permanent magnets have magnetization directions pointing toward the plunger. They provide a resting holding force to keep the plunger in contact with the steel bobbin. Depending on the polarity of solenoid current, the coil may work in opposition to the permanent magnet to unlatch the plunger or it may assist the permanent magnet to pull in the plunger.

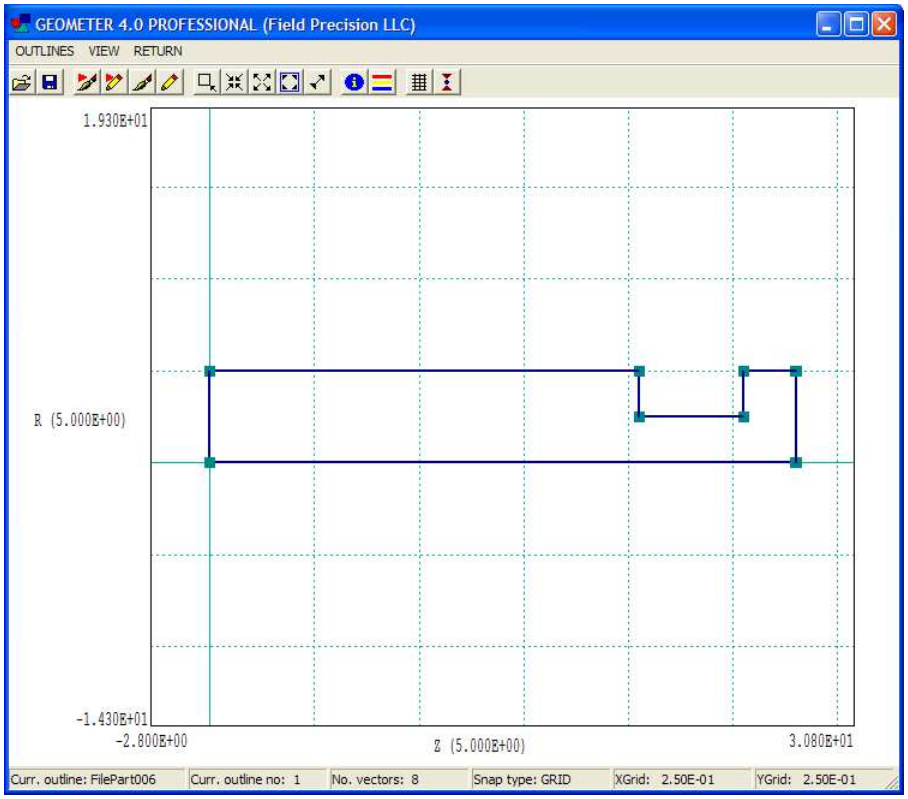


Figure 83: Outline editor showing the outline for the plunger turning.

Run **Geometer** and load the file `Latching.MIN`. Check out the script content with the internal editor. A variable-resolution foundation mesh is employed for accurate field calculations at the gap between the bobbin and plunger. The solution includes five physical regions: air, the steel of the case, the steel plunger and the upper and lower magnets. Notice the use of labels and comments to document features of the calculation. Construction of the mesh is straightforward. The *Box* model is used to represent the steel plates and the magnets, while the bobbin is a *Cylinder*. The plunger is a *Turning*, a model we have not yet discussed. A turning is an outline rotated about the z axis of the workbench space. Note the outline vectors in the script following the *Type* command of the *Plunger* section.

To see the outline, exit the editor and click *Outline*. **Geometer** opens the window of Fig. 83. In contrast to the convention for extrusions, the outline of a turning is defined in cylindrical coordinates, $(z,\ r)$[19]. In the editor, you can modify vectors of the outline using the CAD operations. The changes appear in the **Geometer** display when you return to the main menu. Modifications are recorded if you save the `MIN` file under the same or a different name. To check the variable mesh definitions, exit the outline editor and click *Foundation*. The foundation mesh window shows 2D plots of the assembly along with the initial mesh divisions (before fitting). Figure 84 is a zoomed view in a plane normal to the y axis. Note the region of very fine elements (0.025 mm) along z near the gap between the bobbin and plunger. To investigate the holding force of the solenoid, we need to perform surface integrals with very small gaps.

[19]Vector coordinates of outlines for turnings must satisfy the condition $r \geq 0.0$.

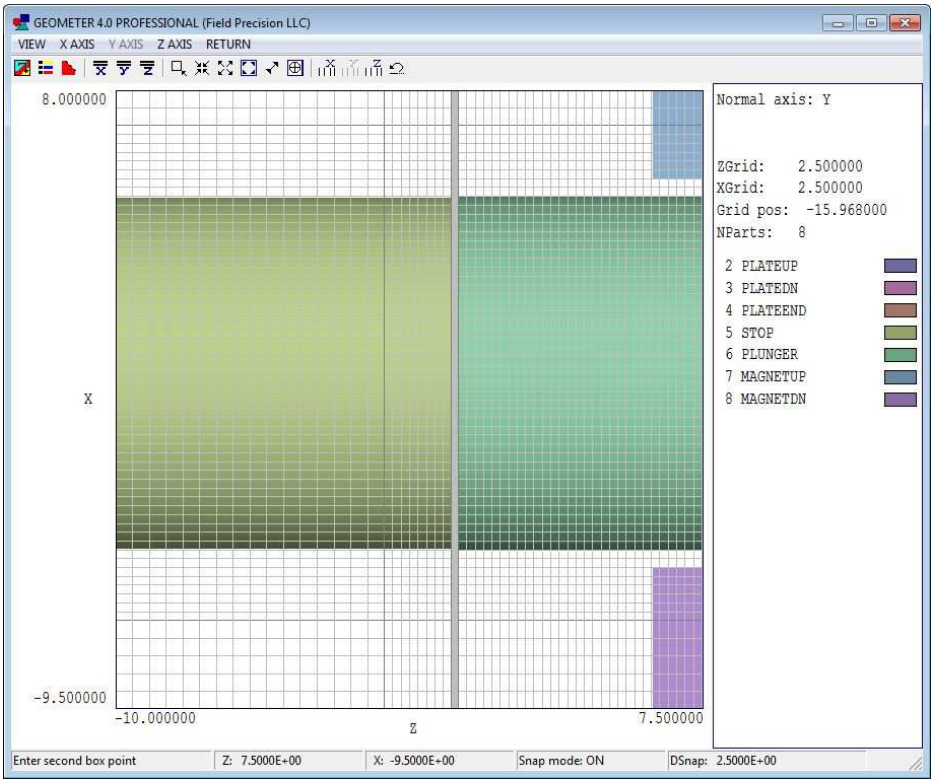


Figure 84: Detailed view of the foundation mesh showing the fine division in z at the bobbin-plunger gap.

Let's proceed to the solution. Run **Magwinder** and load `Latching.CDF` with the content:

```
GLOBAL
  DUnit:   1.0000E+03
  Ds:   2.0000E+00
END
COIL
  Name: Solenoid
  Current: -1.0000E+03
  Part
    Name: Solenoid
    Type: Solenoid
    Fab:   6.0  10.0  27.0  2    20    20
    Shift:   0.00  0.00  -9.50
  End
END
ENDFILE
```

The coil definition file uses the *Solenoid* model to create 800 applied current elements with a coil current of -1000 A-turn. The negative value gives a coil field inside the bobbin in the same direction as the permanent magnet field. Click *File/Save element file* to create `Latching.WND`.

Run **MetaMesh** and process the `MIN` file to create `Latching.MDF`. To check the controls for the finite-element solution, run **Magnum**, click *File/Edit input files* and choose `Latching.GIN`. The file has the following content:

```
SolType = STANDARD
Mesh = Latching
Source = Latching
DUnit = 1000.0
ResTarget = 5.00E-08
MaxCycle = 2000
* Region 1: AIR
Mu(1) =   1.0
 * Region 2: STEEL
Mu(2) =   1000.0
 * Region 3: PLUNGER
Mu(3) =   1000.0
 * Region 4: MAGNETUP
PerMag(4) =   1.25 ( -1.0 0.0 0.0)
 * Region 5: MAGNETDN
PerMag(4) =   1.25 (  1.0 0.0 0.0)
EndFile
```

In contrast to the free-space solutions we discussed previously, the solution type is set to *Standard* and physical characteristics are assigned to the regions. The quantities *ResTarget* and *MaxCycle* control the iterative solution of the finite-element equations – default values are usually appropriate. For magnetic-field solutions, material quantities are the relative magnetic permeability and the parameters of permanent magnets. Because we do not expect saturation effects at the device field levels, we assign the fixed value $\mu_r = 1000.0$ to the case, bobbin and plunger. The specification of a permanent magnet material includes the remanence field $B_r = 1.25$ tesla and a vector pointing along the direction of magnetization. The magnetization of the top magnet points in the -x direction and the bottom magnet in the $+x$ direction.

Run **Magnum** to create the output file `Latching.GOU`. Figure 85 shows the distribution of $|\mathbf{B}|$ in the plane $y = 0.0$ mm with the plunger in contact with the bobbin. The combination of flux from the two magnets produces an approximately uniform field at the contact point of $B_0 = 1.61$ tesla. Note that it is not necessary to surround the assembly with a large external volume because the flux is well-contained in the magnetic circuit.

The goal of the calculation is to find the force on the plunger as a function of the gap width. The force calculation is easy when the plunger is well-separated from the bobbin. Because the plunger is surrounded by air ($\mu_r = 1.0$) elements, we can apply a surface integral of the Maxwell stress tensor over the plunger facets. The definition of the Maxwell tensor is contained in the configuration file `Magview_Standard.CFG`[20]. It is useful to take a moment to look at the configuration file (usually contained in the *Program folder* defined in **FPController**). Open the file with an editor. It contains definitions for plot quantities and numerical calculations. This section applies to automatic surface integrals:

[20]The **MagView** default configuration file is sufficient for most code users. On the other hand, the program has the flexibility to meet the needs of power users. You can set up custom configuration files with user-defined quantities.

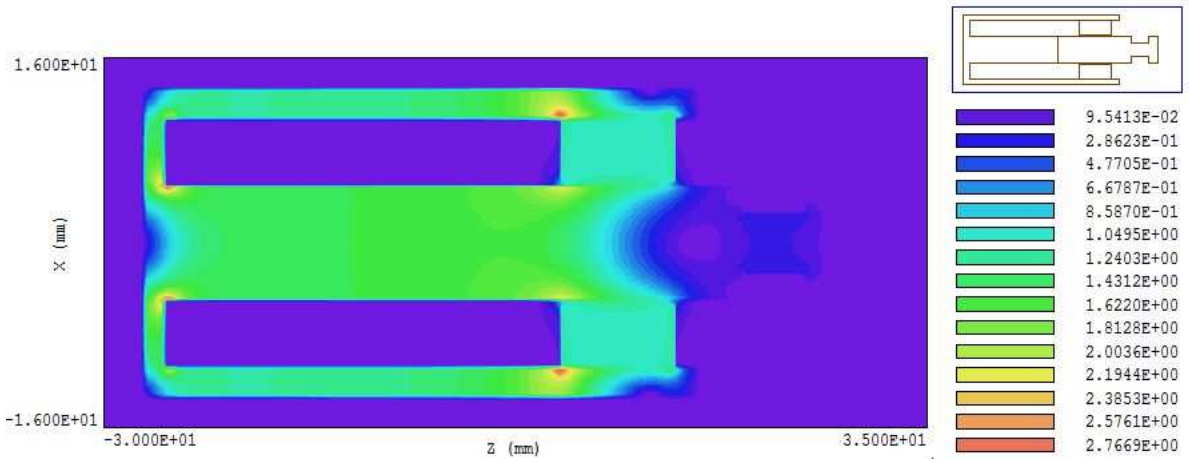


Figure 85: Field distribution in the latched state in the plane $y = 0.0$ mm. Color-coding shows $|\mathbf{B}|$ in tesla.

```
SURFACE
  ...
* Force components
  FxSurf = &Bx 2 ^ &BMag 2 ^ 2.0 / - $IMu0 *;&Bx &By * $IMu0 *;&Bx &Bz * $IMu0 *
  FySurf = &By &Bx * $IMu0 *;&By 2 ^ &BMag 2 ^ 2.0 / - $IMu0 *;&By &Bz * $IMu0 *
  FzSurf = &Bz &Bx * $IMu0 *;&Bz &By * $IMu0 *;&Bz 2 ^ &BMag 2 ^ 2.0 / - $IMu0 *
  ...
END
```

The expressions give the force components determined from the Maxwell integral at a point. Quantities like `&Bx` are calculated field quantities at the point, while quantities like `IMu0` are defined constants.

Run **MagView** and load the file `Latching.GOU`. To find the total force on the plunger, click *Analysis/Surface integrals* in the main menu to bring up the dialog of Fig. 86. The internal region is the *Plunger* and the single external region is *Air*. Click *OK* and save the results to the file `Latching.DAT`. Here is the result for a gap width of 0.20 mm with zero coil current:

```
---------- Surface Integrals ----------
 Region status
 RegNo   Status    Name
 ==================================
     1  External   AIR
     3  Internal   PLUNGER
 Surface area of region set (m2):   1.076727E-03
   FxSurf:   1.103232E-04
   FySurf:  -1.477638E-03
   FzSurf:  -2.413384E+02
```

As an indication of accuracy, the force components F_x and F_y (theoretically zero) are smaller than F_z by a factor exceeding 1/100,000. With a gap of 3.0 mm, the axial force with no coil current is $F_z = -1.111$ N. The force increases to -7.427 N with a coil current of -1000 A-turns.

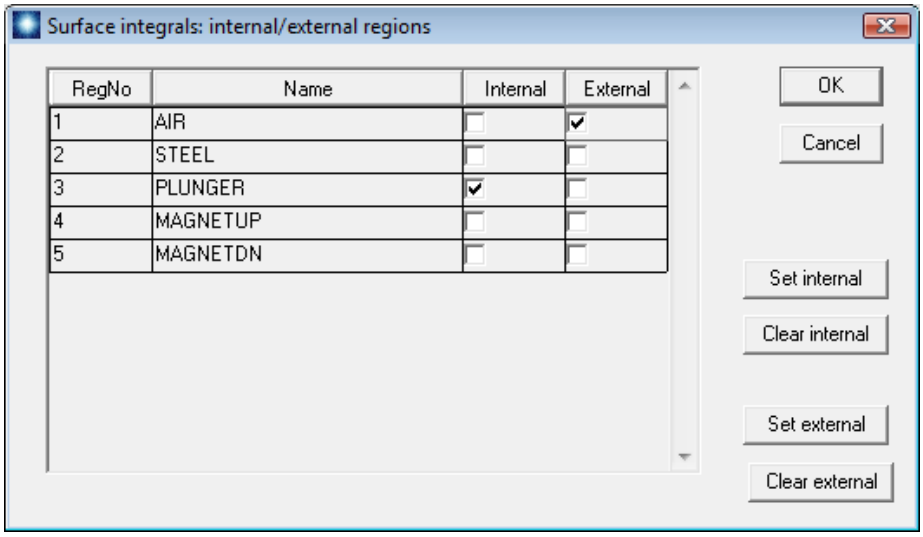


Figure 86: Surface integral dialog. In this case, the integral is taken over all external facets of the plunger in contact with air elements.

A quantity of particular interest is the holding force in the latched state (*i.e.*, plunger touching the bobbin with no coil current). In this case, a Maxwell stress tensor integral around the plunger does not apply because the plunger and bobbin are effectively the same piece of material. One option is to perform a series of calculations with an air gap of decreasing width d_g. The goal would be to fit the force variation with an interpolation function extrapolated to $d_g = 0.0$. Figure 87 shows results of such a calculation. A simple plot of F_z versus d_g would not be informative because the force varies by orders of magnitude. The strong variation reflects the familiar experience of two magnets snapping together when they are close. A helpful observation is that force scales as $1/d_g^2$ for gaps greater than 0.5 mm. Therefore, it is useful to construct a log-log plot of $1/\sqrt{F_z}$ versus d_g. The data of Fig. 87 indicate that the force approaches a constant value at zero spacing. This calculation strategy requires considerable accuracy and effort. It is necessary to include results for very small gap widths ($d_g = 0.05$ mm) to observe the inflection toward a constant value.

Fortunately, there is a simple way to determine the exact holding force from a knowledge of the flux distribution at $d_g = 0.0$ mm. Suppose we displace the plunger an infinitesimal distance δx from the bobbin. The field in the air gap would remain confined to the cross section area A of the steel parts with a value approximately equal to the zero gap field, B_0. The change in field energy in the magnet circuit is

$$\delta U = \frac{B_0^2}{2\mu_0}\, A\, \delta x. \tag{11}$$

Using the principle of virtual work, the holding force is

$$F_z = -\frac{\delta U}{\delta x} = -\frac{B_0^2}{2\mu_0}\, A. \tag{12}$$

With a plunger diameter of 10.0 mm, the area is $A = 7.854 \times 10^{-5}$ m^2. With $B_0 = 1.61$ tesla, the total predicted force is $F_z = -80.935$ N (plotted as a dashed red line in Fig. 87). The mass equivalent is 8.25 kg.

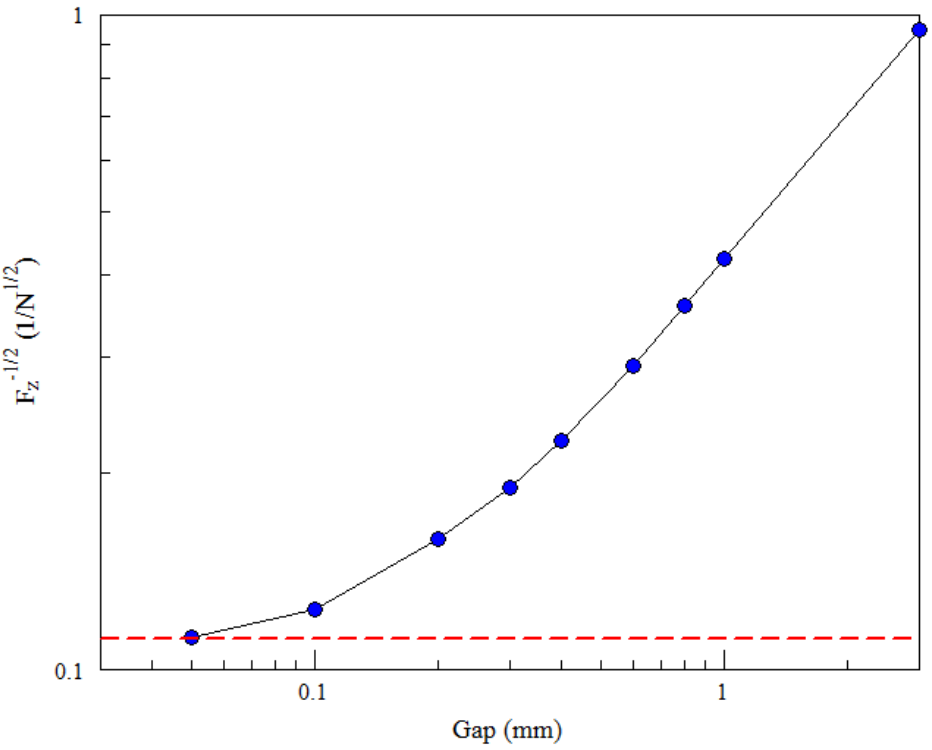


Figure 87: Plot of $1/\sqrt{F_z}$ as function of the gap between the plunger and the bobbin, where F_z is the force on the plunger in newtons. Blue circles indicate results determined by a **MagView** surface integral. The dashed red line indicates the theoretical value for zero gap.

We've covered a lot of territory in this book. We've had a chance to get familiar with the operation sequences and data organization of 2D and 3D programs and discovered many useful analysis techniques. Hopefully, the material will help you get started on your own electric and magnetic field applications. There's still a lot to discover – **EStat**, **PerMag**, **HiPhi** and **Magnum** have a wealth of capabilities that couldn't be covered in a short introduction.

To conclude, I'll list additional resources. First, let's review materials included with the software packages. There are individual PDF manuals for **Mesh**, **EStat**, **PerMag**, **MetaMesh**, **HiPhi** and **Magnum**. They serve as comprehensive references through the extensive use of hyperlinks. The active table of contents is displayed if you activate the bookmark view in your PDF reader. Each manual also has a index with active page links. All software packages include an example library containing ready-to-run, annotated input files for a variety of applications. Be sure to look at text files in the example directories with names like HIPHI_EXAMPLE_INDEX.TXT. They contain a list of the examples along with a brief description of interesting features.

The following free resources are available on our Internet site:

Use this link to download a zip archive of input files for the examples discussed in this book: https://www.fieldp.com/document/femfield_examples.zip.

Finite-element Methods for Electromagnetics. A full-length text published in 1997 by CRC Press. It reviews the physics of electrostatics and magnetostatics and gives a detailed description of the mechanics of **EStat** and **PerMag**. The book is an essential reference if you want to check under the hood to see how finite-element programs work. https://www.fieldp.com/femethods.html

Field Precision Technical library. This Internet page has downloadable copies of the latest manuals for all Field Precision programs. In addition, there are many tutorials in PDF format that review solution techniques for electric and magnetic field applications. http://www.fieldp.com/library.html

Field Precision software tips. This blog includes almost 300 articles on finite-element modeling of electromagnetic fields as well as tips on using Windows computers. The best place to start is the index where articles are organized by related software packages. http://fieldp.com/myblog/index-computational-techniques-by-program/

About the author

Stan Humphries spent the first part of his career as an experimentalist in the fields of plasma physics, controlled fusion and charged particle acceleration. A notable contribution was the creation and demonstration of methods to generate and to transport intense pulsed ion beams. Currently, his work centers on simulations of electromagnetic fields, X-ray technology and material response at high pressure and temperature. Dr. Humphries is the author of over 150 journal publications and the textbooks **Principles of Charged-particle Acceleration** (John Wiley, New York, 1986), **Charged Particle Beams** (John Wiley, New York, 1990) and **Field Solutions on Computers** (CRC Press, Boca Raton, 1997). He received a B.S. in Physics from the Massachusetts Institute of Technology and a Ph.D. in Nuclear Engineering from the University of California at Berkeley. He was elected a Fellow of the American Physical Society and of the Institute of Electrical and Electronic Engineers. Dr. Humphries is a professor emeritus in the Department of Electrical and Computer Engineering at the University of New Mexico and President of Field Precision, an engineering software company.