# FEMM 4.2
# Magnetostatic Tutorial

based on the "Introduction to FEA with FEMM" tutorial by Ian Stokes-Rees, TSS (UK) Ltd.
revised Jan 2006 by David Meeker dmeeker@ieee.org
updated and extended Dec 2014 by Leslie Green CEng MIEE

## Contents

# 1. Introduction

Finite Element Method Magnetics (**FEMM**) is a finite element package for solving 2D planar and 3D axisymmetric[1] problems in low frequency magnetics, electrostatics, and heat flow. **FEMM** runs under Windows 95, 98, ME, NT, 2000 and XP. The program itself can be obtained via the **FEMM** home page at http://www.femm.info

The package is composed of an interactive shell encompassing graphical pre- and post-processing, a mesh generator, and various solvers. A powerful scripting language, Lua 4.0, is integrated with the program. Lua allows users to create batch runs, describe geometries parametrically,[2] perform optimizations, etc. Lua is also integrated into every edit box in the program so that formulae can be entered in lieu of numerical values, if desired. (Detailed information on Lua is available from http://www.lua.org/manual/4.0/ )

There is no hard limit on problem size – maximum problem size is limited by the amount of available memory. Users commonly perform simulations with as many as a million elements.

It is assumed that you are familiar with Windows, so standard operations like “left mouse button click on the OK button” are often abbreviated to “click OK”.

The purpose of this document is to present a step-by-step tutorial to help new users get "up and running" with **FEMM**. The **FEMM** User’s Manual can then be used to better effect. In the first part of this tutorial, the solution for the field of an air-cored coil is considered.

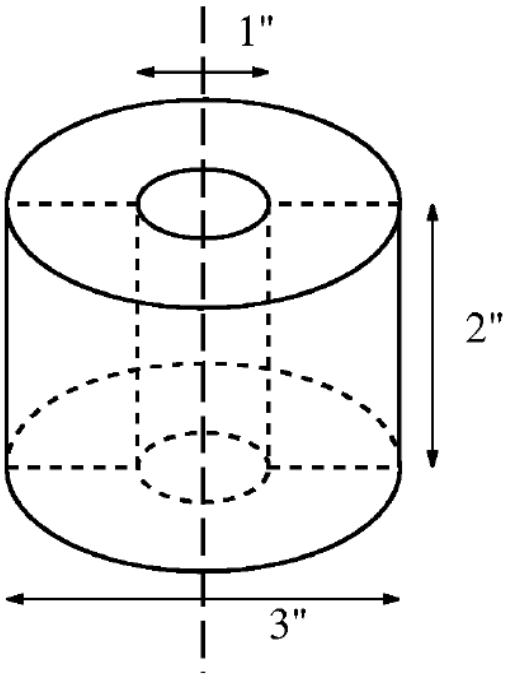



Figure 1: Air-cored coil to be analyzed in the first example.

[1] An 3D axisymmetric problem is a full 3D problem which, by virtue of its axial symmetry, can be drawn and analysed more simply. For FEMM the only type of axisymmetric problem supported is one where the radial cross-section (slice) is the same at every point.
[2] Parametric modelling is explained only in the advanced tutorial section.

# 2. Model Construction and Analysis

This example will take you through a step-by-step process to analyze the magnetic field of an air-cored solenoid sitting in open space. The coil to be analyzed is pictured in Figure 1 (above).

The coil has an inner diameter of 1 inch, an outer diameter of 3 inches, and an axial length of 2 inches. The coil is built out of 1000 turns of 18 AWG copper wire. For the purposes of this example we will consider the case in which a steady current of 1 A is flowing through the wire.

In **FEMM**, one models a slice of the axisymmetric problem. By convention, the $r = 0$ axis is understood to run vertically, and the problem domain is restricted to the region where $r \geq 0$. In this convention, positive-valued currents flow in the into-the-page direction.

## *2.1 Create a new model*

Run the **FEMM** application by selecting **FEMM 4.2** from the **femm 4.2** section of the Windows **Start Menu**.

The default preferences will bring up a blank window with a minimal menu bar.

Select **File** | **New** from the main menu.

The “Create a new problem” dialog will pop up with a drop list allowing you to select the type of problem to be simulated (magnetics, electrostatics, heat flow, current flow). Select **Magnetics Problem** and click OK (left mouse button).

A new blank magnetics problem will be created, and a number of new toolbar buttons will appear.

## *2.2 Set problem definition*

We need to tell the program what sort of problem is to be solved. To do this, select **Problem** from the main menu.

The “Problem Definition” dialog will appear. Set **Problem Type** to **Axisymmetric**.

Make sure that **Length Units** is set to Inches and that the **Frequency** is set to 0.

When the proper values have been entered, click OK.

## *2.3 Draw boundaries*

Fundamentally, finite element solvers find a solution over a finite region of space that contains the objects of interest and enough free space for any fields to adequately decay. In this case, we will choose our solution region to be a sphere with a radius of 4 inches. We need to draw the boundaries for the solution region.

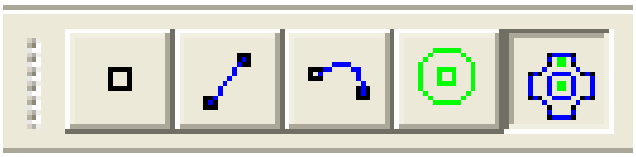
First, adjust the view so that it will contain the entire solution region. Select **Zoom** | **Keyboard** from the main menu to bring up a "Zoom" dialog that allows you to specify the extents of the visible screen. In this dialog, specify …

**Bottom** = -4.2
**Left** = -0.2
**Top** = 4.2
**Right** = 4.2

and click OK. ***Notice that a little bit (0.2) has been added at each edge so that the required 4 inch radius sphere is not hard up against the visible screen area***. (You can change the view area again at any point.)

The screen will be rescaled to the smallest rectangle that contains the specified region, aligned on the left edge.

Next, "node points" need to be defined that bound the sphere. To draw these node points, you can either use the menu **Operation** | **Node** or select the "operate on nodes" button from the horizontal tool bar (the first one in from the left shown below).



From left to right these buttons are:
operate on nodes
operate on segments
operate on arc segments
operate on block labels
operate on groups of objects

Notice that when the **FEMM** window has focus (is active because you have recently clicked somewhere with the **FEMM** application window) a text string will appear in the bottom left corner of the **FEMM** application window identifying the control over which the mouse cursor is hovering, or a position value if the mouse cursor is in the main drawing region.

Place nodes at the top and bottom of the sphere. That is, at points
(r= 0.0, z= 4.0) and
(r= 0.0, z= -4.0) and at the origin
(r= 0.0, z= 0.0).

One can place nodes either by moving the mouse pointer to the desired location and pressing the left mouse button, or by pressing the **<TAB>** key and manually entering the point coordinates via the "Enter Point" popup dialog.

If you happen to accidentally click in the main window and insert an unwanted node you need to know how to delete it. Make sure you are in “operate on nodes” mode and click the select button (the far left button shown below).

***Be warned that apparently nothing happens when you click the button***. You should then left mouse click, hold the button down, then drag the dotted rectangle over the object(s) that you wish to select. The selected items will turn RED. The selected item(s) will then be deleted when you either press the keyboard **Delete** button or mouse click the red cross toolbar button (shown above as the far right button. ***Be warned that you are only in this hidden selection mode for one mouse click. After that you are back to insert node mode again!***

Select the “operate on segments” toolbar button (or menu **Operation** | **Segment** ). In this mode, clicking on or near a desired endpoint with the left mouse button will make the nearest node an endpoint of the line.

Draw a line down the axis of symmetry by first selecting the point at (r= 0.0, z= -4.0). The node will turn RED to show it has been selected. Then left mouse click near the point (r= 0.0, z= 4.0). A blue line will appear linking the two nodes as soon as the second point is selected.

If you have accidentally selected the wrong node then pressing the <ESC> key will unselect it.

Select the “operate on arc segments” toolbar button (or menu **Operation** | **Arc Segment**). Draw an arc to the right of the axis of symmetry by selecting the point (r= 0.0, z= -4.0) and then the point (r= 0.0, z= 4.0). The dialog “Arc segment properties” will appear asking you for some attributes of the arc. In **FEMM**, arcs are approximated by a series of small, straight lines. The “Max. segment” specifies the coarseness with which the arc is divided into sections for meshing purposes. Enter 2.5 into this edit box to get a fairly fine representation of the arc. ***Note that the segment coarseness is not displayed in the drawing window. A perfect arc is drawn regardless of the value chosen.***

Put 180 in the “Arc Angle” edit box to denote that a half circle is being drawn.

The Boundary condition is left unchanged as <NONE>.

If the arc came out wrong and you want to delete it then make sure you are in “operate on arc segments” mode (toolbar button pressed in). Right mouse click near the arc to select it (it will turn RED.) Delete it using the keyboard **Delete** button or the red cross “delete selected objects” tool bar button.

***Note: the arc is drawn anti-clockwise from the first node selected. If you select the nodes in the wrong order, and if you had the drawing view set so the axis was right at the edge of the display area, you would miss this and apparently nothing would have been drawn! This “mystery” segment would also stop the simulation running at a later step, and you would not be able to see why.***

The screen should look like Figure 2 below. The [untitled] in the title bar simply means the design has not yet been saved.

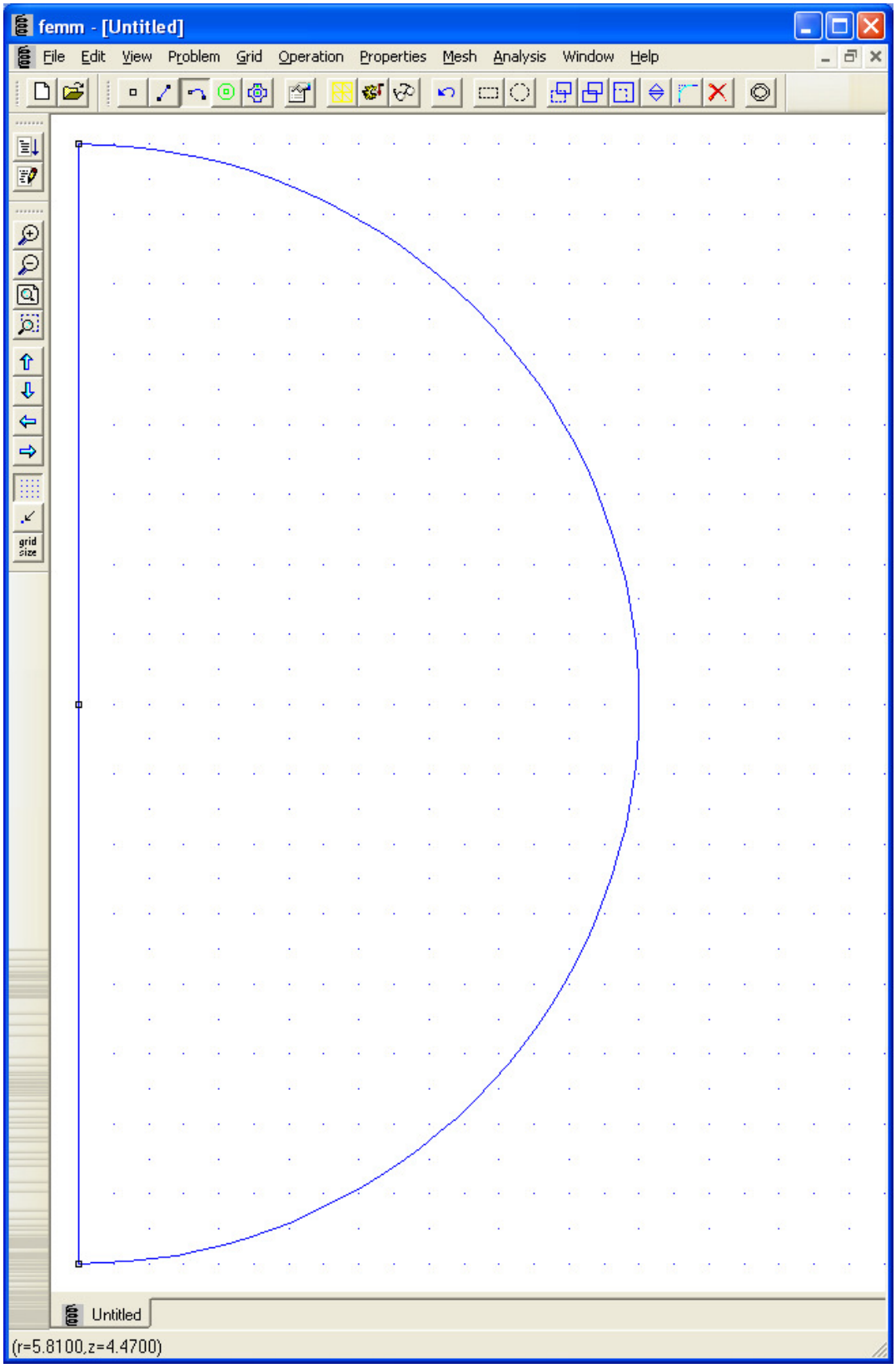



Figure 2: The outer boundaries of the model space have been defined geometrically.

## 2.4 Draw coil

Now, the coil itself can be drawn. Switch back to nodes insertion mode by selecting the "operate on nodes" toolbar button.

Place nodes at

(r= 0.5, z= -1.0)
(r= 1.5, z= -1.0)
(r= 1.5, z= 1.0)
(r= 0.5, z= 1.0)

defining the extents of the coil. This entry is most easily done using the keyboard only. So, for example, the first node would be typed in as

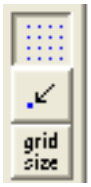
<TAB> 0.5 <TAB> -1.0 <ENTER>

Select the "operate on segments" toolbar button so that lines can be drawn connecting the points. By selecting the nodes defining the coil in pairs, sequentially, one obtains lines between each of the nodes, resulting in a large connected box.

## 2.5 Place block labels

Click the "operate on block labels" toolbar button (green octagon surrounding a green square) or use menu **Operation** | **Block**. Place a block label in the coil region, and place one in the air outside the coil region. Like node points, block labels can be placed either by a click of the left mouse button, or via the **<TAB>** point-entry dialog.

The program uses block labels to associate materials and other properties with various regions in the problem geometry. Next, we will define some material properties, and then we will go back and associate them with particular block labels.




NOTE: If snap-to-grid is enabled then it may be difficult to place the block label in the empty space. If this is the case, disable snap-to-grid by de-selecting the tool bar button with the point and arrow, shown on the left as the middle of the three grid manipulation buttons.

## 2.6 Add materials to the model

Select the main menu item **Properties** | **Materials Library**.

Then drag-and-drop "Air" from the left side **Library Materials** to the right side **Model Materials** using the left mouse button to add it to the current model. You will need to click on the + symbols in the Library tree to expand and see the contents of the relevant folder(s).

Expand the **Copper AWG Magnet Wire** folder and drag 18 AWG to the right window. There should now be two items, **Air** and **18AWG** showing in the **Model Materials** folder.

Click OK.

## 2.7 Add a "Circuit Property" for the coil

Select main menu sequence **Properties** | **Circuits**.

On the “Property Definition” dialog that appears, click the **Add Property** button to create a new circuit property. Name the circuit by replacing “New Circuit” with “Coil”.

Specify that the circuit property is to be applied to a wound region by selecting the **Series** radio button. Enter “1” as the Circuit Current.

Click OK for both the Circuit Property and Property Definition dialogs.

## 2.8 Associate properties with block labels.

Right click the block label node in the air region outside the coil. The block label will turn RED, signifying that it is selected. Press the keyboard spacebar to open the “Properties for selected block” dialog box. (Instead of pressing the spacebar, one could use the “Open up Properties Dialog for currently selected entities” toolbar button, shown below on the far right ).

Set the **Block type** to “Air”. (Notice that there are only two material types available in the selection. These were added to the project in a previous step.)

Uncheck the “Let Triangle choose Mesh Size” checkbox and enter 0.1 for the Mesh size. The Mesh size parameter defines a constraint on the largest possible element size allowed in the associated section. The mesh generator attempts to fill the region with nearly equilateral triangles in which the sides are approximately the same length as the specified Mesh size parameter. When the “Let Triangle choose Mesh Size” box is checked, the mesh generator is free to pick its own element size, usually resulting in a somewhat coarse mesh.

Click OK.

The block label will have changed from <None> to “Air”, and a circle will have appeared around the block label indicating the approximate mesh size in the associated region.

Repeat the same procedure for the block label node inside the coil region, again changing the mesh size to 0.1. Set the **Block type** to “18 AWG”. We want to assign currents to flow in this region, so we select “Coil” from the **In Circuit** drop list. The **Number of turns** edit box will become activated if a series-type circuit is selected for the region (in this case, the Coil property that was previously defined).

Enter “1000” as the number of turns for this region, signifying that the region is filled with 1000 turns wrapped in a counter-clockwise direction (that is, positive turns in the right-hand-screw rule sense).

Click OK.

NOTE: If we wanted to model the turns as being wrapped in a counter-clockwise direction instead, we could have specified the number of turns as “-1000”.

## 2.9 Create boundary conditions

Select **Properties** | **Boundary** from the menu bar, then click the **Add Property** button.

Replace the name “New Boundary” with “ABC” and change the “BC Type” to **Mixed**.

The ABC name is meant as a reminder that we are creating an "Asymptotic Boundary Condition" that approximates the impedance of an unbounded, open space. In this way, we can model the field produced by the coil in an unbounded space while still only modeling a finite region of that space. When the **Mixed** boundary condition type is selected the $c_0$ coefficient and $c_1$ coefficient edit boxes will become enabled.

These entries are meant to represent coefficients in a boundary condition of the form:

$$\frac{1}{\mu_r \mu_0} \cdot \frac{\partial A}{\partial n} + c_0 A + c_1 = 0$$

where

- $A$ is magnetic vector potential
- $\mu_r$ is the relative magnetic permeability of the region adjacent to the boundary
- $\mu_0$ is the permeability of free space
- $n$ represents the direction normal to the boundary.

For our asymptotic boundary condition, we need to specify:

$$c_0 = \frac{1}{\mu_r \mu_0 R} \qquad\qquad c_1 = 0$$

where $R$ is the outer radius of the spherical problem domain. To enter these values into the dialog box, enter 0 for the $c_1$ coefficient and 1/(**uo***4*inch) for the $c_0$ coefficient.[3] The Lua scripting language processes the contents of each edit box automatically when the dialog is closed, substituting the numerical value of the permeability of free space for **uo** and 0.0254 for inch [4] and evaluating the result.

Click OK on both dialog boxes.

To assign this boundary condition, switch to “operate on arc segments” mode. You need to open the arc properties dialog box for the outer boundary. Select the arc defining the outer boundary by right mouse clicking on the arc. It will turn RED to show that it is selected. You can now use either the spacebar or the “open properties dialog for currently selected items” toolbar button. Select “ABC” from the **Boundary cond.** drop list and click OK.

You have now defined enough boundary conditions to solve the problem, since the $r = 0$ line in axisymmetric problems is not an external boundary, just a symmetry axis.

You have now completed modeling the whole problem. Save it to some convenient file name such as *MyWork* using the standard windows **File** | **Save as …** menu.

---

[3] Make sure you enter **uo** as lower case letters U and O and not using a ZERO. To check what Lua gives you can try doing a modify on the boundary condition after it has been entered. Instead of being in symbolic form it will have been converted into a number (7832428.301…)

[4] It is essential that the evaluated value of $c_0$ is in units of meters, regardless of the units set for the model.

The finished pre-processor geometry should look as pictured in Figure 3 below.

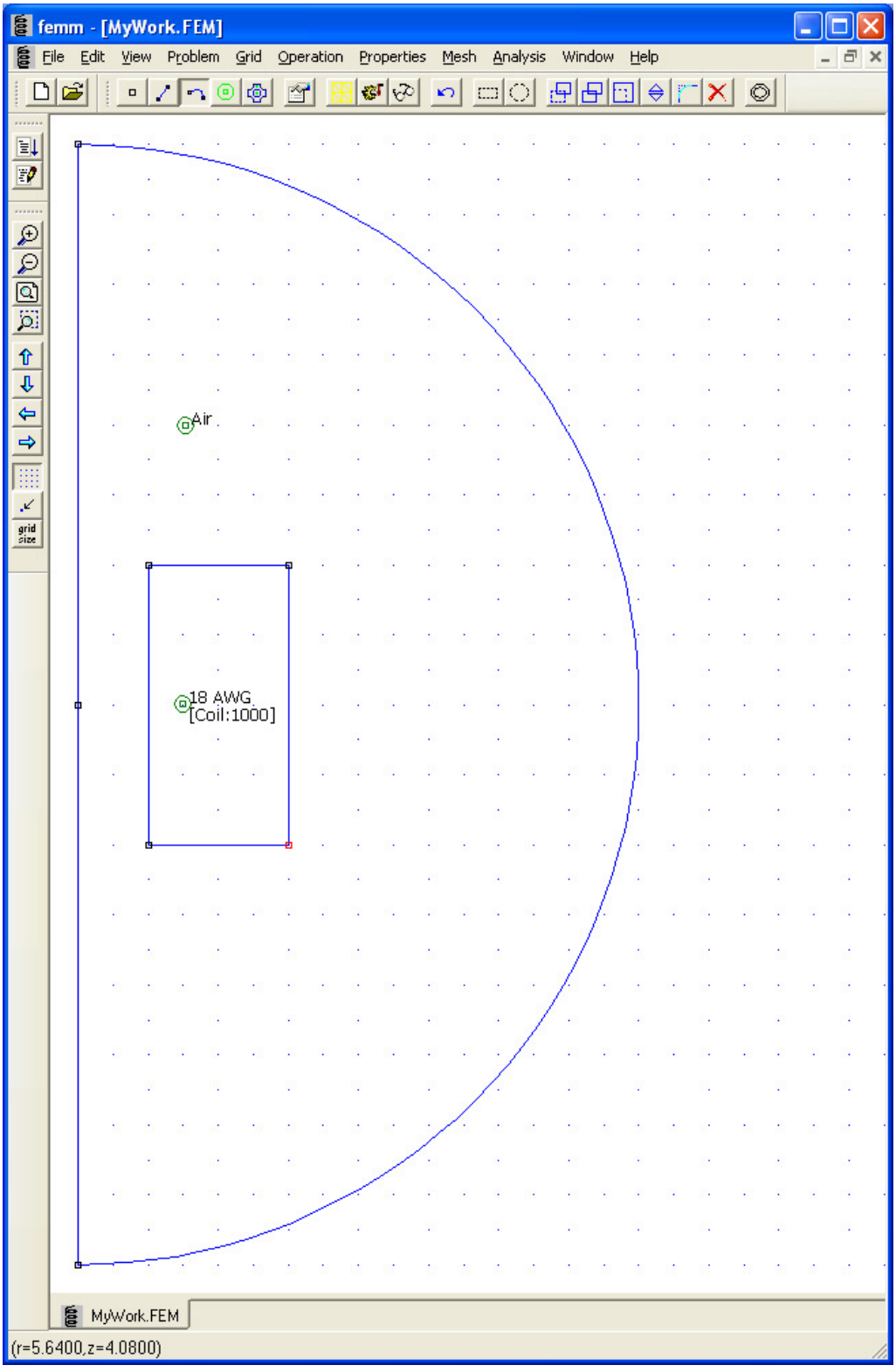



Figure 3: Completed coil model, ready to be analyzed.

## *2.10 Generate mesh and run FEA*

Click the toolbar button with the [***barely visible***] yellow mesh, shown on the left below.

The text in the bottom right of the main window will show “Run mesh generator” when the mouse pointer hovers over the correct button.

The mesh generator creates and then draws a triangular mesh for your problem. If the mesh spacing seems too fine or too coarse you can select block labels or line segments and adjust the mesh size defined in the properties of each object. The number of nodes generated is a key “size” for the model. Too small and the model will be inaccurate. Too large and the simulation time can become excessive.

Once the mesh has been generated, dismiss the node-count reporting dialog and click the “turn the crank” button, shown in the middle below, (“Run Analysis”) to analyze the model.

Processing status information will be displayed, but may disappear too quickly for you to see what happened. On the other hand, if the progress bars do not seem to be moving then you should probably cancel the calculation. This can occur if insufficient boundary conditions have been specified. For this particular problem, the calculations should be completed within a second. There is no confirmation when the calculations are complete, the status window just disappears when the processing is finished.

When the mesh size is set to 0.1 the number of nodes reported is 3077 (although this exact value may change with later versions of **FEMM**). The simulation time is too short to measure even on an old dual core AMD Athlon running at 2.2GHz. By reducing the mesh size to 0.03, and re-running the mesh generator, the number of nodes increases to 28813.

We might have expected an increase of up to $\left(\frac{10}{3}\right)^2 = 11.1\times$, and we actually got 9.4×.

It still simulates in less than 4 seconds.

Decreasing the mesh size again to 0.01 yields 255072 nodes, and the simulation time has increased to around 1 minute.

# 3. Analysis Results

Click the glasses icon (shown on the right below) to view the analysis results.

A post-processor window (actually a tab selectable window, with the tabs at the bottom) will appear. The post-processor window will allow you to extract many different sorts of information from the solution.

## *3.1 Point values*

Just like the pre-processor, the post-processor window has a set of different editing modes: Point, Contour, and Area. The choice of mode is specified by the mode toolbar buttons, where the first button corresponds to Point mode, the second to Contour mode, and the third to Area mode.

By default, when the program is first installed, the post-processor starts out in Point mode. By clicking on any point with the left mouse button, the various field properties associated with that point are displayed in the floating **FEMM** Output window. [NOTE: If the **FEMM** Output window is not visible use the menu sequence **View** | **Output window**.]

Similar to drawing points in the pre-processor, the location of a point can be precisely specified by pressing the **<TAB>** button and entering the coordinates of the desired point in the dialog that pops up. For example, if the point (r= 0.0, z= 0.0) is specified in the pop-up dialog, the resulting properties displayed in the output window are as pictured in Figure 4 below. Since flux density is a vector quantity it is displayed as a magnitude , |***B***|, as well as in ***r*** and ***z*** component form. Likewise for the ***H*** field.

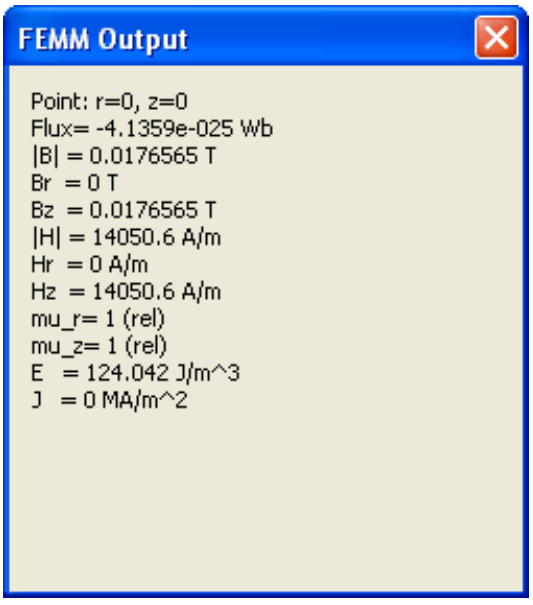



Figure 4: Display of field values at the point (0,0).

## 3.2 Coil terminal properties

With **FEMM**, it is straightforward to determine the inductance and resistance of the coil as seen from the coil's terminals. Press the toolbar button with the coil symbol (shown below on the right)

to display the resulting attributes of each Circuit Property that has been defined. For the Coil property defined in this example, the resulting dialog is pictured in Figure 5 below.

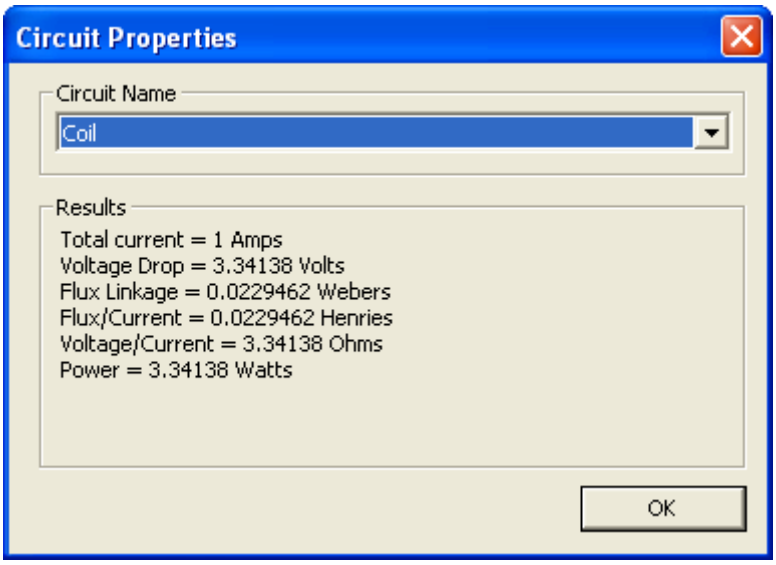



Figure 5: Circuit Property results dialog.

Since the problem is linear, and there is only one current, the Flux/Current result can be unambiguously interpreted as the coil's inductance (22.9 mH). The resistance of the coil is the Voltage/Current result (3.34 Ω).

Any finite element analysis model is an approximation to the real situation and it is useful to quickly explore how the circuit results change according to the model size.

| | Auto mesh | 0.10 mesh | 0.03 mesh | 0.01 mesh |
|---|---|---|---|---|
| Nodes | 3039 | 3077 | 28813 | 255,072 |
| Resistance | 3.341Ω | 3.341Ω | 3.341Ω | 3.341Ω |
| Inductance | 22.90mH | 22.90mH | 22.95mH | 22.95mH |

Clearly FEMM has produced a stable result, independent of the mesh size. This is always a good test to ensure that the meshing has succeeded.

## *3.3 Plotting field values along a contour*

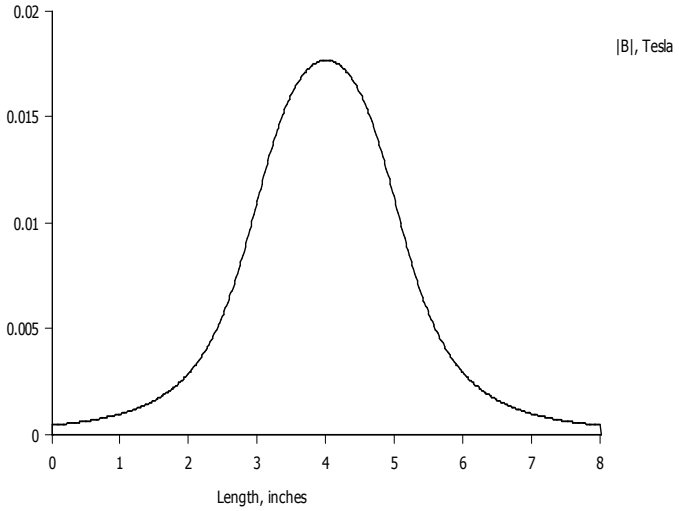
FEMM can also plot values of the field along a user-defined contour. Here, we will plot the flux density along the centerline of the coil. Switch to **Contour mode** by pressing the "Contour Mode" toolbar button. You can now define a contour along which flux will be plotted.

There are three ways to add points to a contour:

1. ***Left Mouse Button Click*** adds the nearest input node to the contour
2. ***Right Mouse Button Click*** adds the current mouse pointer position to the contour
3. ***<TAB> Key*** displays a point-entry dialog that allows you to enter the coordinates of a point to be added to the contour.

Here, method 1 will be used. Click near the node points at (r= 0.0, z= 4.0), (r= 0.0, z= 0.0), and (r= 0.0, z= -4.0) with the left mouse button, adding the points in the order given above. Then, press the **Plot** toolbar button (with the graph symbol, shown on the left below)

Click OK in the "X-Y Plot of Field Values" pop-up dialog – the default selection is magnitude of flux density. If desired, different types of plot can be selected from the drop list on this dialog.




**NOTE:** It is often the case in the solution to magnetic problems that the field values are discontinuous across a boundary. In this case, **FEMM** determines which side of the boundary will be plotted, based on the order in which points are added. For example, if points are added around a closed contour in a counterclockwise order, the plotted points will lie just to the inside of the contour. If the points are added in a clockwise order, the plotted points will lie just to the outside of the contour. The implication for our example problem is that the contour along the $r = 0$ line must be defined in order of decreasing ***z*** (and therefore counterclockwise, meaning that the plotted points will lie inside the solution domain instead of outside it, where the field values are not defined).

## *3.4 Plotting flux density*

By default, when the program is first installed, only a black-and-white graph of flux lines is displayed. Flux density can be plotted as a color density plot, if you so desire. To make a color density plot of flux, click the rainbow shaded toolbar button (shown below, second from the right) to generate a color flux density plot.

When the dialog box comes up, select the Flux density plot radio button and accept the other default values. Click OK. The resulting solution view will look similar to that pictured below in Figure 5. [Note how the plot has been moved in slightly from the edges using the **Zoom** | **keyboard** method mentioned at the very beginning of this tutorial.]

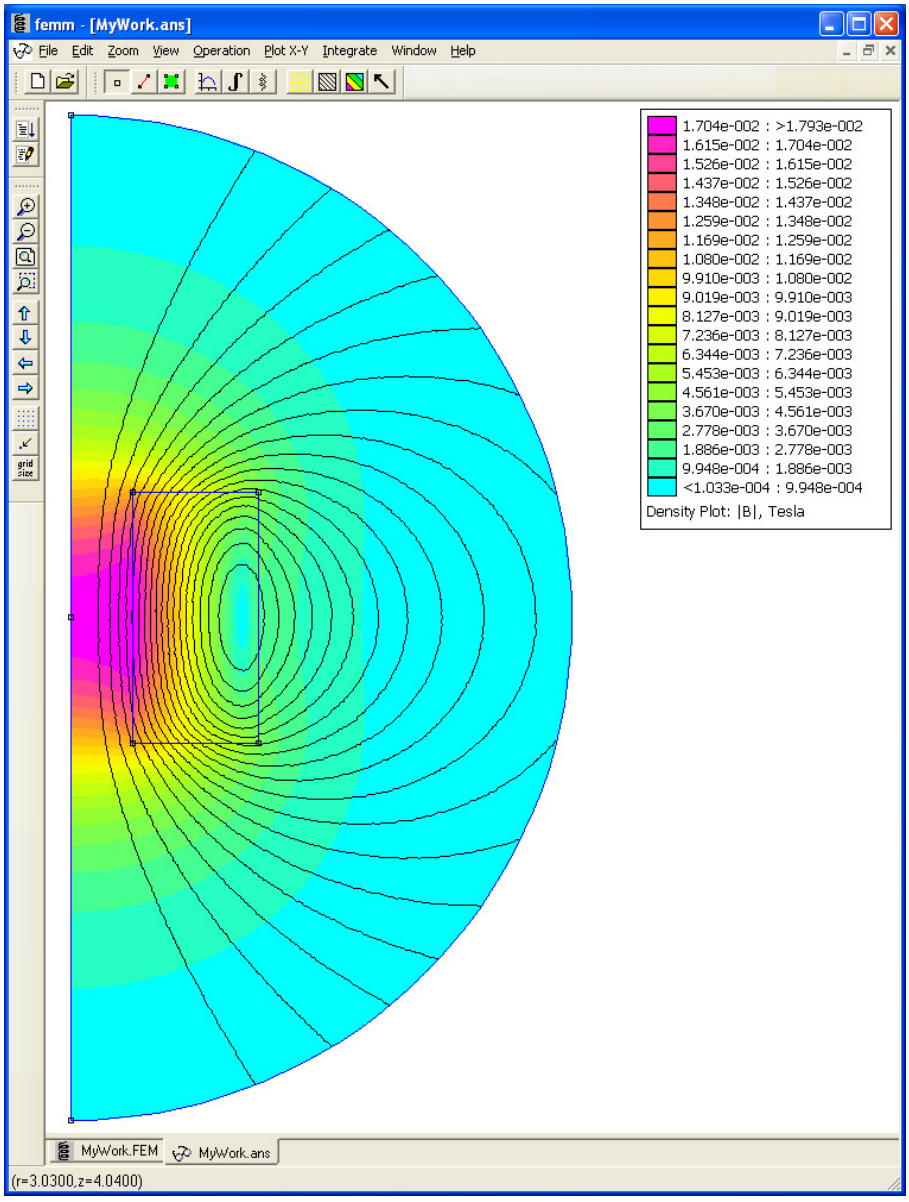
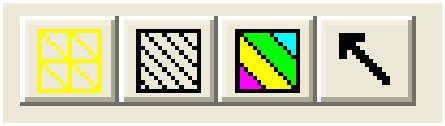
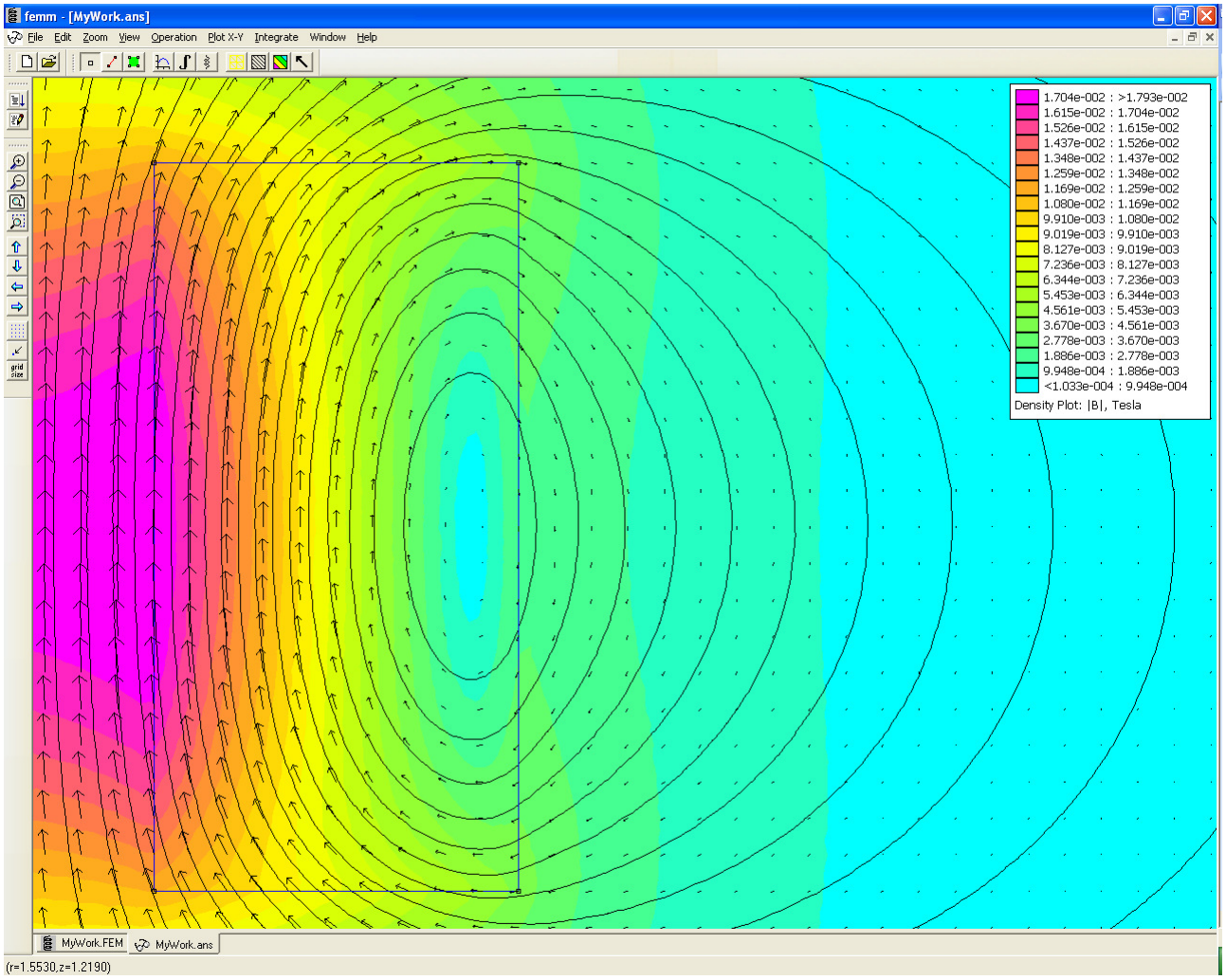


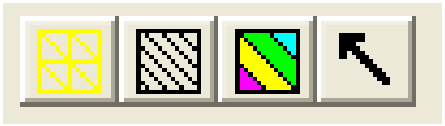

Figure 5: Color flux density plot of the solution.



The arrow button (on the right above) adds a vector plot on top of the existing plot, where little arrows show the direction and magnitude of the field.




The above plot was zoomed-in using the **Zoom** | **Zoom In** menu sequence. It was also necessary to scroll the plot using the keyboard cursor keys, but again a menu sequence **Zoom** | **Scroll Left** could have been used. To turn off the vector plot, re-click the vector plot button and set the vector plot type to <None>.



The black and white hatched button calls up a dialog box (unimaginatively) called “Dialog” which sets the number of flux contours drawn. Unchecking the “Show flux lines” checkbox removes the flux lines completely.

The yellow toolbar button (shown on the left above) draws the mesh on the plot when the button is pressed in. To remove the displayed mesh just unpress the button.

# 4. Gaining Confidence and Expertise with FEMM

## *4.1 Getting sensible results when using FEMM*

FEMM does not have built-in error checking and as such it is neither idiot proof nor even idiot resistant. In the initial setup we defined a coil size, number of turns, and wire gauge. Unfortunately, these three factors are mutually dependant. In real life you can really only choose two of the three.

Let's check what happens when we define 10× the number of turns in the coil. The resistance increases by a factor of 10× as expected. The inductance increase by roughly a factor of 100×, as expected. And the power increases by a factor of 10× as expected. But what is not mentioned is that the volume of copper has increased by a factor of 10× as well. Therefore the very first thing you need to do when specifying a coil is to look at how many turns will actually fit in the available winding space. This depends not only on the wire gauge (the copper part), but also on the thickness of the insulation and any bondcoat [5] used. These details will change from one manufacturer to another, and from one wire type to another, being dependant on voltage and temperature ratings in particular.

You may have wondered why I limited the comparison resolution on the resistance and inductance in an earlier section to 4 digits. The answer is that all the rest of the digits presented by **FEMM** are essentially meaningless. Consider a real copper coil. It is usual that the self-heating of a coil could be 50°C or more from an OFF to an ON state. It is well known that copper has a temperature coefficient of resistance around +0.4%/°C. A 50°C temperature rise gives something like a 20% increase in resistance. Quoting the coil resistance to 0.1% accuracy, or better, is therefore meaningless for any real world application.

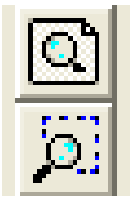
## *4.2 Verifying FEMM's results for inductance*

It is important that you trust the results from simulations otherwise there is no point in using the program. In order to gain trust in the program it is important to take known situations, simulate them, and see how the **FEMM** results compare to your expectations. This rule of course applies to any type of software, from any source.

**Example 1:**

A single-turn coil with a radius of 100mm, wound using 2mm diameter round wire, has a calculated inductance of 589nH. (This value is based on a formula that is ultimately traceable to work published by Bashenoff in 1927.)

Try simulating this and see what inductance value you get.



HINT: You should zoom in on the coil. This is done using the menu sequence **View** | **Window** and left mouse click/dragging a zoom box around the coil. **View** | **Natural** will take you back to an overview of the whole problem. There are also toolbar buttons on the left hand side of the screen which have the same function. The lower button shown on the left is the zoom to window.

[5] Bondcoat is used to stick the wire to itself and is essential in self-supporting windings, but not essential for coils wound on bobbins.

My simulation gave an inductance of 612nH, an agreement to within 4%.

Be careful to set the boundary condition for $c_0$ correctly. For this problem I set the spherical outer boundary at 200mm with the problem units set to mm. But $c_0$ has to be set in meters, so the correct entry for $c_0$ is 1/(**uo***0.2)

If you think the two inductance values should agree to within 0.1% then you should rethink the situation. It is not at all certain which of the two inductances would be closer to reality if we made such a coil. Therefore adjusting things to make the two results closer to each other can give misleading results.

It is highly probable that your simulation will give a (slightly) different value to mine. In general you have to pick a boundary and there is no "right answer" except when you build and measure the final product.

**Example 2:**

A single layer coil of 100 turns of 1mm round wire has a diameter of 100mm and a height of 100mm. The theoretical inductance is 0.6794mH. (This value is ultimately derived from a 1909 paper by Nagaoka.)

What value does **FEMM** give for your simulation?

My simulation gave 0.6733mH, an agreement to within 1%.

## *4.3 Verifying FEMM's results for pull force*

Whilst inductance simulations are all very well for testing purposes, they were easy to calculate anyway. The more interesting situations arise where calculation is not possible and therefore verification can only occur by measurement. This is an expensive business unless somebody else has published their measurement data.

The **FEMM** wiki page does have a comparison of pull force published in a text book from 1941, http://www.femm.info/wiki/RotersExample

This is not the sort of data you seem to be able to find in modern text books. More measured data can be found in "Solenoids, Electromagnets and Electromagnetic Windings" by C.R. Underhill (2nd edition). The reprinted edition of 1992 shows the original publication date as 1921 (with copyright dates of 1910 and 1914).

# 5. Conclusions

You have now completed your first few models of magnetic problems with **FEMM**. From this basic introduction, you have been exposed to the following concepts:

- How to draw a model using nodes, segments, arc, and block labels.
- How to add materials to your model and how to assign them to regions.
- How to specify the finite element mesh size.
- How to define boundaries for your model.
- How to define and apply boundary conditions.
- How to analyze a problem.
- How to inspect local field values.
- How to plot field values along a line.
- How to compute inductance and resistance.
- How to display color flux density plots.
- How to interpret and understand the accuracy of simulation results.

Hopefully, this tutorial has presented you with enough of the basics of **FEMM** so that you can explore more complicated problems without getting sidetracked by the mechanics of how a problem is drawn and analyzed.

More information can be found in the **FEMM** User's Manual.

Now might be a good time to experiment with **FEMM** a bit more before going on to the Advanced Tutorial which follows.

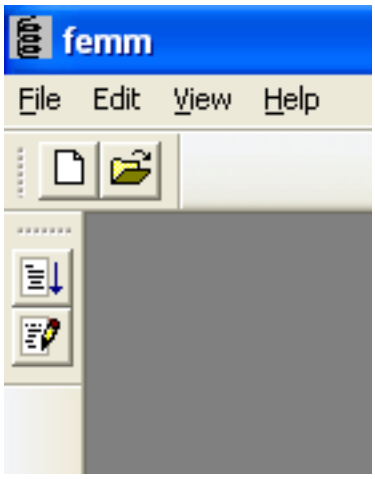
# 6. Advanced FEMM Tutorial on scripting

## *6.1 Introduction to Lua*

**FEMM** plots fields on a static model. It would be nice to be able to move or change the model and re-simulate, then plot the results of the change. This can (mostly) be done using the built-in Lua scripting language. The plotting needs an external program like Excel.

A Lua script is a plain text file with a .lua file extension contain commands which can be understood by the Lua interpreter built into **FEMM**.

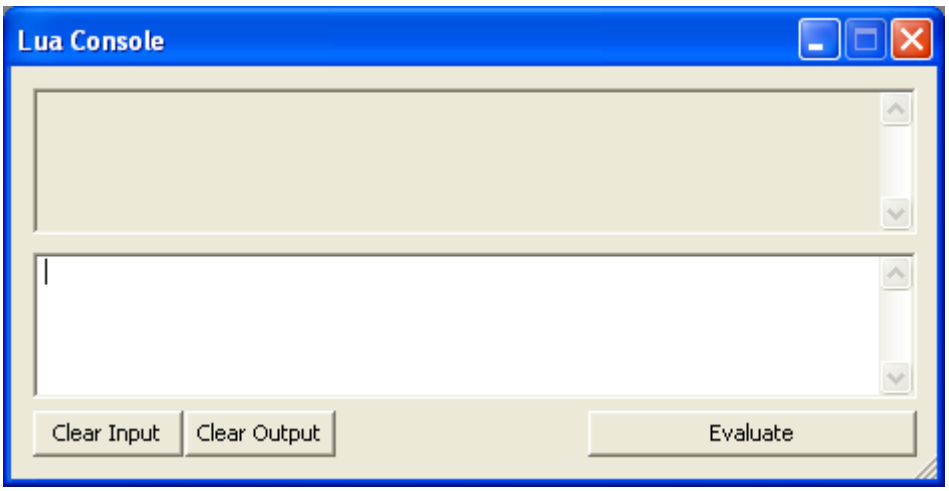
When you first start **FEMM** you get a very basic menu structure.




The icon with the page and a down arrow opens an existing lua script file.

The icon with the page and the pencil opens the Lua console. Alternatively the Lua console can be opened using the menu sequence **View** | **Lua Console**.

Go ahead and open the Lua console.




The upper area is an output screen. The lower area is a command entry field. You can type in a command and click the **Evaluate** button to try out Lua commands. Start by typing

**newdocument(0)** then click **Evaluate**

This is equivalent to clicking on the **FEMM** file menu and opening a new magnetics problem. Notice that the command is echoed (copied) into the Lua Console output window,

showing the command history. Any command action that can be carried out on the **FEMM** interface will have an equivalent command in the **FEMM** specific Lua scripting language. These are written out in the **FEMM** user manual, but that is not a good way to learn the language – hence this tutorial.

## *6.2 Overview of the tutorial problem*

For the purposes of this tutorial we are going to plot the pull force between two identical cylindrical permanent magnets. This will be an axisymmetric problem so the two magnets have to be on a common axis.

Whilst you can enter textual values into **FEMM** dialog boxes, which Lua then interprets, this does *not* make **FEMM** a parametric design tool. In some CAD tools you can define dimensions in terms of variables. So, for example, you could define a diameter as **dia** and then change the value of **dia** at some later stage in the design. This will not work in **FEMM** because Lua interprets the string as soon as it is entered. It does not remember that **dia** was used to define a dimension so when you change **dia** later, only subsequently entered values will change.

However, if you write the whole model as a Lua script you can change the variable values and re-run the whole script. Now you do have a parametric design tool! Whilst it takes longer to design the model parametrically, it is very convenient to be able to answer questions such as “Can I make the magnets smaller, but from a stronger material and get as strong a pull force?”

Lua is a dynamically typed language so you can just go ahead and define variables. Let’s define our magnet problem parametrically.

```
magnetDia = 10.0
magnetHeight= 11.0
```

I am using colour for contrast, but that is not relevant. Copy and paste these commands into the Lua console (highlight with mouse, CTRL+C to copy, CTRL+V to paste). I won’t bother mentioning to click the **Evaluate** button to process these commands from now on.

Notice that the Lua Console again echoes the commands. This time you pasted two lines into the input window at once. You can paste as many lines as you like at one time. To show that Lua has remembered these values you could enter the following command

```
print( magnetdia)
```

but the result is **nil** because Lua is case sensitive. You need to be precise in the entry.

```
print( magnetDia)
```

works as expected.

When testing commands by typing them into the Lua Console, the screen is not always updated. If you suspect that this has happened, use the refresh view command.

```
mi_refreshview()
```

## 6.3 Plan of action

The plan of action for the scripting task is as follows.

- Adjust the problem variables to suit what you would like to simulate.
- Calculate how big the axisymmetric space should be and set it.
- Calculate the window zoom setting and set it.
- Draw the magnets with appropriate properties.
- Calculate the boundary conditions and set them.
- Calculate an appropriate mesh size and run the mesh generator.
- Simulate the problem.
- Calculate and save the pull force.
- Repeat for all the gaps we are interested in.

Once you have written the script you will be able to automatically calculate the pull force between any two identical cylindrical magnets. You will also have used and understood many of the Lua scripting facilities.

The Lua script is going to run to quite a few lines, but don't be intimidated by that. Each step is very simple once you see what it is trying to do. Hence comments, preceded by a double dash, occur frequently.

It would be boring to go through each line of the script, one by one. And in any case it is unnecessary. If you read through the script, along with the comments, you will be able to see what it is doing. You can also copy and paste the commands in sequence into the Lua console to see what they do.

It is convenient to use Notepad++ to write the script, but you could just use Notepad or any other text editor. The advantage of using Notepad++ is that if you tell it the language is VB (Visual Basic) you get a certain amount of relevant syntax colouring to improve readability.

When you follow along with the script, some of the commands have multiple parameters. To see what is being done you may need to look up the command in the **FEMM** User Manual to see what all the parameters of the functions do. This is good practice as well!

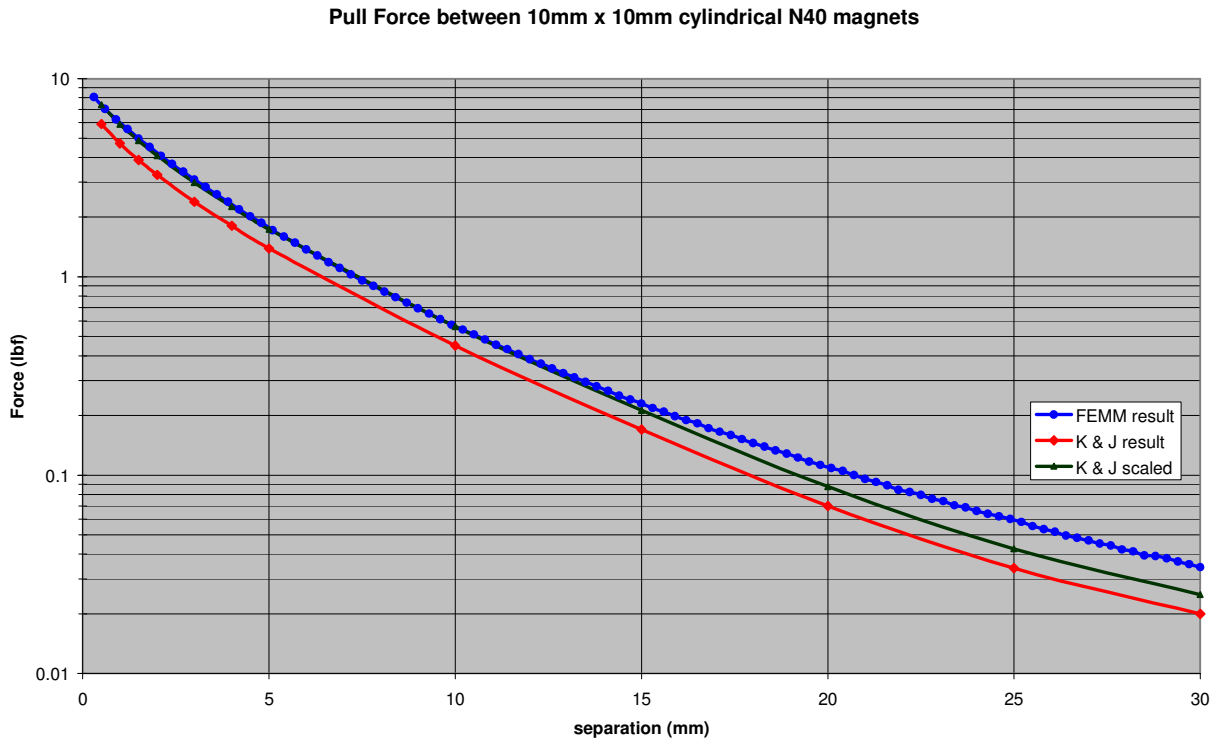
Lua has similarities to Basic, but function return values can seem strange. Most programming languages return at most one value, whereas Lua can return an arbitrary number. This feature is sufficiently unusual to warrant a little example on its own.

```
outFile, why = openfile("c:\\FEMMresult.csv","w")
if outFile==nil then print("Failed to create result file: ", why) end
```

The first line tries to open a .csv file for writing the output data. Notice that there are two variables to the left of the equals sign! Both of them are return values from the **openfile** function. If the function is successful then **outFile** will contain a ***handle*** to the open file. If unsuccessful then the **why** variable will be a string saying what went wrong.

Because the final Lua script does not conveniently fit on an A4 page size, I have put the script on an A3 page at the very end of this document. You can look at the script now, or you can just continue on to the results.

## 6.4 FEMM results and comparison




The output csv (comma separated value) file was read into MS Excel. We could either believe the result or try to find a way of checking it. I used the K & J magnetics calculator

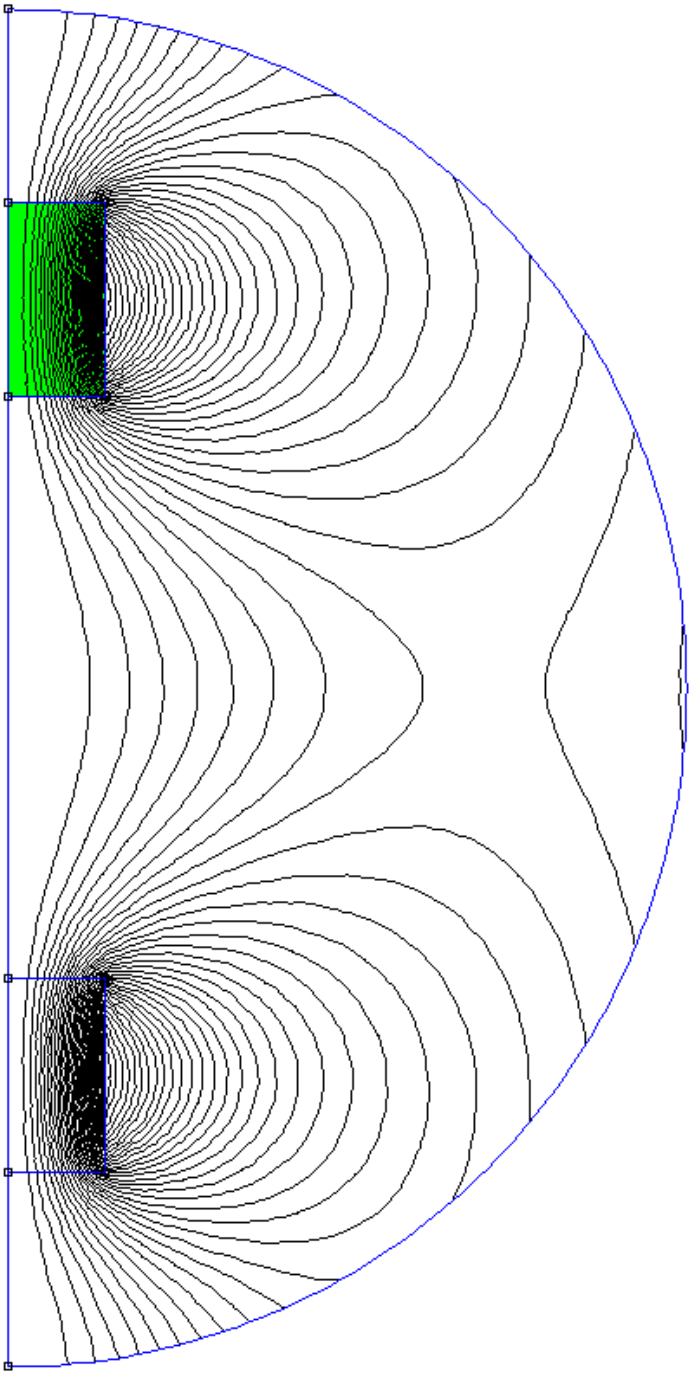
http://www.kjmagnetics.com/calculator.asp

which claims to be based on extensive measurements. If I multiply the K & J results by 1.25× I get the green line above, which seems like a reasonable fit to the **FEMM** results for the most interesting part of the curve. It would have been much better to get hold of the raw measurements from K & J (or elsewhere), but that exceeds the requirements of this tutorial. Likewise, comparison of the N40 material curves from K & J with the N40 material in the **FEMM** library is outside the scope of this tutorial.

(Note that the K & J calculator results are only given to 2 decimal places, so I jiggled the 25mm point from 0.03 lbf to 0.034 lbf to make a more sensible looking curve.)

Since you will ordinarily be simulating something new, and therefore the performance may be relatively unknown, it is worthwhile hammering home the point about “sanity checking” your results. In the development of the simulation script which follows, the boundary sphere radius was initially set to

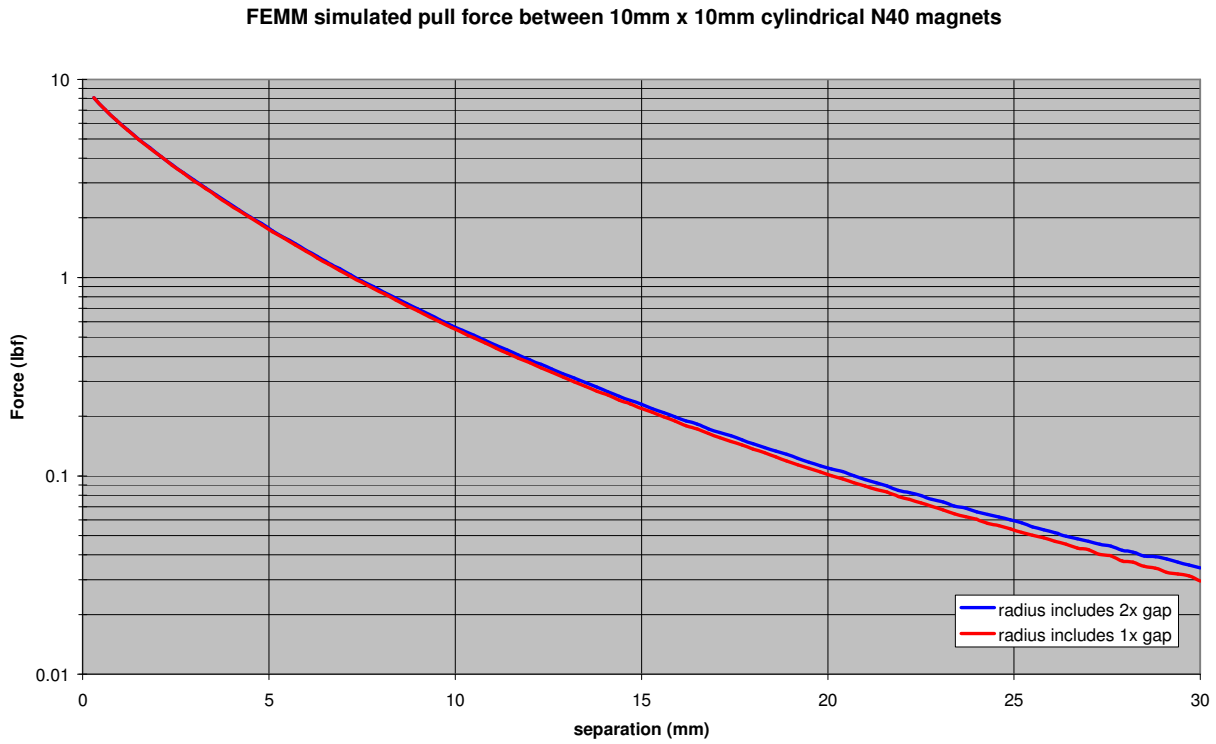
**radius = 2.0 * magnetHeight + 0.5 * gap**



The resulting flux line plot may seem quite reasonable, but it is not. The resulting force plot went negative just past 23mm separation, which is a disastrous prediction. Since the previous curves were plotted on a logarithmic force scale, we could not even plot this result on the same graph!

**radius = 2.0 * magnetHeight + gap** -- plotted below in red

**radius = 2.0 * (magnetHeight + gap )** -- final version in blue




As you might expect, as the magnets get further apart the results diverge from each other more and more due to the proximity of the boundary. In reality, though, nobody is likely to be too upset by the different simulation results at the weak end of the force curve. Nevertheless this does illustrate a way of checking the accuracy of the result by increasing the boundary radius and seeing how stable the result is. You may remember seeing a similar technique used earlier in this tutorial when the meshing was changed to see how tolerant the simulation was to mesh size.

Remember the promises made at the start of the introductory tutorial? The script which follows has one loop to step the gap size, but once you understand the process you can wrap more loops around the gap size loop to change magnet diameter, height and/or material, and then plot groups of force curves. Optimisation follows from that.

Next up is the tutorial simulation script. In the first instance read the comments for each block and try to get a feel for what is being done rather than studying it line by line. Try running it and see what happens. Then try changing things, like the number of points, to get some hands-on experience with it.

Remember that this script is for tutorial purposes. It is not finished, fully verified code.

The code frankly looks horrible in a word/pdf document. You really need to copy/paste the code into a syntax colouring editor such as notepad++ in order to read it more easily.

## 6.5 Lua tutorial script

```
-- Advanced FEMM 4.2 Tutorial by Leslie Green CEng MIEE, Dec 2014
-- This LUA script simulates the pull force vs distance curve between two identical cylindrical magnets

---------------------------------------------------------------------------------------------
-- modify these values to suit your needs
units = "millimeters"        -- should be one of "inches","millimeters","centimeters","mils","meters","micrometers"
forceUnits= "lbf"            -- should be one of "lbf", "Newtons", "kgf"
magnetDia =   10.0
magnetHeight= 10.0
graphPoints = 100                    -- Making this larger will increase simulation time proportionally.
plotRange =   3.0                    -- the range of the gap is calculated for a multiple of the biggest of height and diameter of the magnet
magnetType= "NdFeB 40 MGOe"  -- has to be the EXACT name of a material in the materials library
---------------------------------------------------------------------------------------------

showconsole()                -- does nothing if the console is already displayed
clearconsole()               -- clears both the input and output windows for a fresh start.
remove("c:\\FEMMresult.csv")       -- get rid of the old data file, if it exists

newdocument(0)               -- the 0 specifies a magnetics problem
mi_hidegrid()

-- define the problem type, equivalent to the top level menu    Problem
mi_probdef(0, units, "axi", 1E-8)

-- adds these materials from the Material Library to the project
mi_getmaterial(magnetType)
mi_getmaterial("Air")

-- this bit is a bit of a pain. Although we have set the problem units above,
-- that has no effect on the calculation of c0 for the Asymptotic Boundary Condition
-- We therefore need to create a scaling factor for c0, dependant on the units
c0_scale=1.0
if units=="micrometers" then c0_scale= 1000000.0 end
if units=="millimeters" then c0_scale= 1000.0 end
if units=="centimeters" then c0_scale= 100.0 end
if units=="meters"      then c0_scale= 1.0 end                              -- this is the easy one!
if units=="inches"      then c0_scale= 1.0/0.0254 end
if units=="mils"        then c0_scale= 1000.0/0.0254 end

forceScale = 1.0
if forceUnits=="Newtons" then forceScale= 1.0         end        -- the natural units of the weighted stress tensor
if forceUnits=="lbf"     then forceScale= 0.2248089   end
if forceUnits=="kgf"     then forceScale= 0.1019716   end

outFile, why = openfile("c:\\FEMMresult.csv","w")                -- note that Lua can return multiple values from a function
if outFile==nil then print("Failed to create result file: ", why) end

-- to remove uncertainty, write out a header in the results file
write(outFile, "# Magnet Material=", magnetType, "\n")
write(outFile, "# Magnet Diameter=", magnetDia, " ", units, "\n")
write(outFile, "# Magnet Height=", magnetHeight, " ", units, "\n")
write(outFile, "# Data format is ...", "\n")
write(outFile, "gap in ", units, ", force in ", forceUnits, "\n")

-- this is the main loop that does all the real work

for n= 1, graphPoints do                        -- don't start from zero or you will get a zero gap
        print( "step ", n, " of ", graphPoints)                                    -- announce the step number to show progress

        if magnetDia > magnetHeight then
        gap = (magnetDia * plotRange * n)/graphPoints                    -- note: the gap is never set to zero
else
        gap = (magnetHeight * plotRange * n)/graphPoints
end

-- set the size of the solution sphere radius as the larger of the magnet diameter and (twice the magnet height + twice the gap)
radius = 2.0 * (magnetHeight + gap)
if radius < magnetDia        then
  radius = magnetDia
end
mi_zoom(-radius*0.1, -radius*1.05, radius*1.5, radius*1.05)     -- set the window to a nice size for the problem

-- as the radius changes we need to change the boundary condition to suit
mi_deleteboundprop("ABC")
c1=0
c0= c0_scale/(uo*radius)
mi_addboundprop("ABC",0,0,0,0,0,0,c0,c1,2)                       -- create the Asymptotic Boundary Condition for the problem

-- draw the r=0 axis and the outer boundary
mi_addnode(0,-radius)
mi_addnode(0, radius)
mi_addsegment(0,-radius,0,radius)
mi_addarc(0,-radius,0,radius,180,1)
mi_selectarcsegment(0,radius)
mi_setarcsegmentprop(1,"ABC",0,0)            -- make sure we set the Asymptotic boundary condition for the problem
mi_refreshview()
```

```
-- draw the upper magnet ------------------------------------------------------
mi_addnode(0.0,gap/2.0)                                                 -- bottom left
mi_addnode(0.0, magnetHeight + gap/2.0)                                 -- top left
mi_addsegment(0.0,gap/2.0,0.0,magnetHeight + gap/2.0)
mi_addnode(magnetDia/2.0, magnetHeight + gap/2.0)                       -- top right
mi_addsegment(0.0,magnetHeight + gap/2.0,magnetDia/2.0,magnetHeight + gap/2.0)
mi_addnode(magnetDia/2.0, gap/2.0)                                      -- bottom right
mi_addsegment(magnetDia/2.0, magnetHeight + gap/2.0, magnetDia/2.0, gap/2.0)
mi_addsegment(magnetDia/2.0, gap/2.0, 0.0, gap/2.0)
mi_refreshview()

-- draw the lower magnet ------------------------------------------------------
mi_addnode(0.0, -gap/2.0)                                               -- top left
mi_addnode(0.0, -magnetHeight -gap/2.0)                                 -- bottom left
mi_addsegment(0.0, -gap/2.0, 0.0, -magnetHeight - gap/2.0)
mi_addnode(magnetDia/2.0, -magnetHeight - gap/2.0)                      -- bottom right
mi_addsegment(0.0,-magnetHeight -gap/2.0, magnetDia/2.0, -magnetHeight - gap/2.0)
mi_addnode(magnetDia/2.0, -gap/2.0)                                     -- top right
mi_addsegment(magnetDia/2.0, -magnetHeight - gap/2.0, magnetDia/2.0, -gap/2.0)
mi_addsegment(magnetDia/2.0, -gap/2.0, 0.0, -gap/2.0)
mi_refreshview()

-- add and set block properties -----------------------------------------------
if (magnetDia/2.0) > magnetHeight then
        magMesh = magnetHeight/20.0
else
        magMesh = magnetDia / 40.0
end
if (gap/3.0) < (radius/50.0) then
        airMesh = (gap/3.0)
else
        airMesh = radius/50.0
end

mi_clearselected()
mi_addblocklabel(magnetDia/4.0,  (magnetHeight+gap)/2.0)                -- upper magnet
mi_selectlabel  (magnetDia/4.0,  (magnetHeight+gap)/2.0)
mi_setblockprop(magnetType, 0, magMesh, "", 90, 0)

mi_clearselected()
mi_addblocklabel(magnetDia/4.0, -(magnetHeight+gap)/2.0)                -- lower magnet
mi_selectlabel  (magnetDia/4.0, -(magnetHeight+gap)/2.0)
mi_setblockprop(magnetType, 0, magMesh, "", 90, 0)

mi_clearselected()
mi_addblocklabel(magnetDia/10.0, radius*0.9)                            -- air
mi_selectlabel  (magnetDia/10.0, radius*0.9)
mi_setblockprop("Air", 0, airMesh, "", 0, 0)

mi_clearselected()
mi_refreshview()

------------------ meshing and analysis
mi_saveas("temp21354.fem")          -- you need to save before creating a mesh for some reason
--mi_createmesh()                   -- not strictly necessary, but helpful to see what is going on  --- UNCOMMENT for debugging
mi_refreshview()
mi_analyze()
mi_loadsolution()               -- brings up the post-processor window of the solution

mo_selectblock(magnetDia/4.0,  (magnetHeight+gap)/2.0)
force = -forceScale * mo_blockintegral(19)  -- 19 is the z part of steady-state weighted stress tensor force
write(outFile, gap, ",", force, "\n")
        ----------------------------  get ready to start all over again by deleting all the nodes and labels
        if n==graphPoints then break end    -- leave a nice picture up at the end of the run
        mo_close()                                                      -- close the post processor output window
        mi_purgemesh()
        mi_selectcircle(0,0,radius*1.1,0)
        mi_selectcircle(0,0,radius*1.1,1)
        mi_selectcircle(0,0,radius*1.1,2)
        mi_selectcircle(0,0,radius*1.1,3)
        mi_deleteselected();
        mi_refreshview()
end

result, why= flush(outFile)
if result==nil then print("Failed to flush file: ", why) end

result, why= closefile(outFile)
if result==nil then print("Failed to close file: ", why) end
```