

Finite Element Method Magnetics

Version 3.3

User's Manual

March 17, 2003

David Meeker
dmeeker@ieee.org
<http://femm.berlios.de>
©2002

Acknowledgements

Thanks to the following people for their valuable contributions to FEMM:

- Si Hang, for writing fast point location routines that greatly increase the speed at which FEMM evaluates line integrals;
- Robin Cornelius, for adding Lua extensions for FEMM for scripting / batch run capabilities;
- Keith Gregory, for his detailed help and comments.
- Stefan Engstrom, for figuring out how to run FEMM on linux machines via wine;
- Ian Stokes-Rees, for writing an excellent introductory tutorial;
- Peter Krc and Frank Lenning for contributing some special-purpose batch-run code;
- The regular posters on the FEMM mail list for their great suggestions, comments and advice;
- Victor Petoukhov, for his insightful questions;
- Martin Furlan, for finding several bugs and helping to work through the fine points of force and torque computation;
- Anders Dahlberg, for finding several bugs and making a number of suggestions as to features that would improve FEMM's capabilities and ease of use;
- Jonathan Shewchuk, for creating `triangle`, the excellent mesh generator that FEMM uses;
- Eric Maslen, for counseling me to release FEMM as freeware.

Contents

1	Introduction	5
2	Overview	6
2.1	Relevant Maxwell's Equations	6
2.1.1	Magnetostatic Problems	6
2.1.2	Harmonic Problems	7
2.2	Boundary Conditions	8
2.3	Finite Element Analysis	9
3	Preprocessor	11
3.1	Preprocessor Drawing Modes	11
3.2	Keyboard and Mouse Commands	11
3.3	View Manipulation	13
3.4	Grid Manipulation	14
3.5	Edit	15
3.6	Problem Definition	16
3.7	Definition of Properties	17
3.7.1	Point Properties	18
3.7.2	Boundary Properties	19
3.7.3	Materials Properties	21
3.7.4	Materials Library	25
3.7.5	Circuit Properties	26
3.8	Spawned Tasks	27
3.9	DXF Import/Export	28
4	Postprocessor	29
4.1	Postprocessor modes	29
4.2	View and Grid Manipulation	29
4.3	Keyboard Commands	30
4.4	Mouse Actions	30
4.5	Contour Plot	30
4.6	Density Plot	31
4.7	Line Plots	31
4.8	Line Integrals	33
4.9	Block Integrals	34
4.10	Force/Torque Calculation	37
4.10.1	Lorentz Force/Torque	37
4.10.2	Weighted Stress Tensor Volume Integral	38
4.10.3	Maxwell Stress Tensor Line Integral	38
4.11	Exporting of Graphics	41
4.12	Circuit Results	42
4.13	Miscellaneous Useful View Commands	42

5	Lua Scripting Documentation	44
5.1	What Lua Scripting?	44
5.2	Preprocessor Lua Command Set	44
5.2.1	Object Add/Remove Commands	44
5.2.2	Geometry Selection Commands	46
5.2.3	Object Labeling Commands	46
5.2.4	Problem Commands	47
5.2.5	File Commands	47
5.2.6	Mesh Commands	48
5.2.7	Editing Commands	48
5.2.8	Zoom Commands	49
5.2.9	View Commands	49
5.2.10	Object Properties	50
5.2.11	Miscellaneous	53
5.3	Post Processor Command Set	53
5.3.1	Data Extraction Commands	53
5.3.2	Selection Commands	57
5.3.3	Zoom Commands	57
5.3.4	View Commands	58
5.3.5	Miscellaneous	59
6	Numerical Methods	60
6.1	Finite Element Formulation	60
6.2	Linear Solvers	60
6.3	Field Smoothing	61
A	Appendix	65
A.1	Modeling Permanent Magnets	65
A.2	Bulk Lamination Modeling	68
A.3	Open Boundary Problems	71
A.3.1	Truncation of Outer Boundaries	71
A.3.2	Asymptotic Boundary Conditions	71
A.3.3	Kelvin Transformation	74
A.4	Nonlinear Time Harmonic Formulation	76

1 Introduction

FEMM is a suite of programs for solving static and low frequency problems in magnetics. The programs currently address problems on two-dimensional planar and axisymmetric domains. FEMM is divided into three parts:

1. Preprocessor (`femme.exe`). This is a CAD-like program for laying out the geometry of the problem to be solved and defining material properties and boundary conditions. Autocad DXF files can be imported to facilitate the analysis of existing geometries.
2. Solver (`fkern.exe`). The solver takes a set of data files that describe problem and solves the relevant Maxwell's equations to obtain values for the magnetic field through the solution domain.
3. Postprocessor (`femmview.exe`). This is a graphical program that displays the resulting fields in the form of contour and density plots. The program also allows the user to inspect the field at arbitrary points, as well as evaluate a number of different integrals and plot various quantities of interest along user-defined contours.

Two additional programs are also called to perform specialized tasks. These are:

- `triangle.exe`. Triangle breaks down the solution region into a large number of triangles, a vital part of the finite element process. This program was written by Jonathan Shewchuk, and is available from his Carnegie-Mellon University web page, or from Netlib.
- `femmpplot.exe` This small program is used to display various 2-D plots. It also allows the user to save and view any files in the Extended Metafile (`.emf`) format.

The Lua scripting language is also integrated into the pre- and post-processors. Lua allow "batch" runs to be performed without user interaction. In addition, all edit boxes in the user interface are parsed by Lua, allowing equations or mathematical expressions to be entered into any edit box in lieu of a numerical value. In any edit box in FEMM, a selected piece of text can be evaluated by Lua via a selection on the right mouse button click menu.

The purpose of this document is to give a brief explanation of the kind of problems solved by FEMM and to provide a fairly detailed documentation of the programs' use.

2 Overview

The goal of this section is to give the user a brief description of the problems that FEMM solves. This information is not really crucial if you are not particularly interested in the approach that FEMM takes to formulating the problems. You can skip most of *Overview*, but take a look at Section 2.2. This section contains some important pointers about assigning enough boundary conditions to get a solvable problem.

Some familiarity with magnetism and Maxwell's equations is assumed, since a review of this material is beyond the scope of this manual. However, the author has found several references that have proved useful in understanding the derivation and solution of Maxwell's equations in various situations. A very good introductory-level text is Plonus's *Applied electromagnetics* [1]. A good intermediate-level review of Maxwell's equations, as well as a useful analogy of magnetism to similar problems in other disciplines is contained in Hoole's *Computer-aided analysis and design of electromagnetic devices* [2]. For an advanced treatment, the reader has no recourse but to refer to Jackson's *Classical electrodynamics* [3].

2.1 Relevant Maxwell's Equations

For the low-frequency problems addressed by FEMM, only a subset of Maxwell's equations are required. By definition, low-frequency problems are those problems in which displacement currents can be ignored. Displacement currents are typically relevant only at radio frequencies.

2.1.1 Magnetostatic Problems

Magnetostatic problems are problems in which the fields are time-invariant. In this case, the *field intensity* (H) and *flux density* (B) must obey:

$$\nabla \times H = J \tag{1}$$

$$\nabla \cdot B = 0 \tag{2}$$

subject to a constitutive relationship between B and H for each material:

$$B = \mu H \tag{3}$$

If a material is nonlinear (*e.g.* saturating iron or alnico magnets), the permeability, μ is actually a function of B :

$$\mu = \frac{B}{H(B)} \tag{4}$$

FEMM goes about finding a field that satisfies (1)-(3) via a *magnetic vector potential* approach. Flux density is written in terms of the vector potential, A , as:

$$B = \nabla \times A \tag{5}$$

Now, this definition of B always satisfies (2). Then, (1) can be rewritten as:

$$\nabla \times \left(\frac{1}{\mu(B)} \nabla \times A \right) = J \tag{6}$$

For a linear isotropic material (and assuming the Coulomb gauge, $\nabla \cdot A = 0$), eq. (6) reduces to:

$$-\frac{1}{\mu}\nabla^2 A = J \quad (7)$$

FEMM retains the form of (6), so that magnetostatic problems with a nonlinear B - H relationship can be solved.

In the general 3-D case, A is a vector with three components. However, in the 2-D planar and axisymmetric cases, two of these three components are zero, leaving just the component in the “out of the page” direction.

The advantage of using the vector potential formulation is that all the conditions to be satisfied have been combined into a single equation. If A is found, B and H can then be deduced by differentiating A . In addition, the form of (6), and elliptic partial differential equation, arises in the study of many different types of engineering phenomenon. There are a large number of tools that have been developed over the years to solve this particular problem.

2.1.2 Harmonic Problems

If the field is time-varying, eddy currents can be induced in materials with a non-zero conductivity. Several other Maxwell’s equations related to the electric field distribution must also be accommodated. Denoting the *electric field intensity* as E and the *current density* as J , E and J obey the constitutive relationship:

$$J = \sigma E \quad (8)$$

The induced electric field then obeys:

$$\nabla \times E = -\frac{\partial B}{\partial t} \quad (9)$$

Substituting the vector potential form of B into (9) yields:

$$\nabla \times E = -\nabla \times \dot{A} \quad (10)$$

In the case of 2-D problems, (10) can be integrated to yield:

$$E = -\dot{A} - \nabla V \quad (11)$$

and the constitutive relationship, (8) employed to yield:

$$J = -\sigma \dot{A} - \sigma \nabla V \quad (12)$$

Substituting into (6) yields the partial differential equation:

$$\nabla \times \left(\frac{1}{\mu(B)} \nabla \times A \right) = -\sigma \dot{A} + J_{src} - \sigma \nabla V \quad (13)$$

where J_{src} represents the applied current sources. The ∇V term is an additional voltage gradient that, in 2-D problems, is constant over a conducting body. FEMM uses this voltage gradient in some harmonic problems to enforce constraints on the current carried by conductive regions.

FEMM considers (13) for the case in which the field is oscillating at one fixed frequency. For this case, a *phasor transformation* [2] yields a steady-state equation that is solved for the amplitude and phase of A . This transformation is:

$$A = \text{Re} [a(\cos \omega t + j \sin \omega t) =] = \text{Re} [ae^{j\omega t}] \quad (14)$$

in which a is a complex number. Substituting into (13) and dividing out the complex exponential term yields the equation that FEMM actually solves for harmonic magnetic problems:

$$\nabla \times \left(\frac{1}{\mu(B)} \nabla \times a \right) = -j\omega\sigma a + \hat{J}_{src} - \sigma \nabla V \quad (15)$$

in which \hat{J}_{src} represents the phasor transform of the applied current sources.

Strictly speaking, the permeability μ should be constant for harmonic problems. However, FEMM retains the nonlinear relationship in the harmonic formulation, allowing the program to approximate the effects of saturation on the phase and amplitude of the fundamental of the field distribution. There are a number of subtleties to the nonlinear time harmonic formulation—this formulation is addressed in more detail in Appendix A.4.

FEMM also allows for the inclusion of complex and frequency-dependent permeability in time harmonic problems. These features allow the program to model materials with thin laminations and approximately model hysteresis effects.

2.2 Boundary Conditions

Some discussion of boundary conditions is necessary so that the user will be sure to define an adequate number of boundary conditions to guarantee a unique solution. Boundary conditions for FEMM come in three flavors:

- *Dirichlet*. In this type of boundary condition, the value of A is explicitly defined on the boundary, *e.g.* $A = 0$. The most common use of Dirichlet-type boundary conditions is to define $A = 0$ along a boundary to keep flux from crossing the boundary.
- *Neumann*. This boundary condition specifies the normal derivative of A along the boundary. Usually, $\partial A / \partial n = 0$ is defined along a boundary to force flux to pass the boundary at exactly a 90° angle to the boundary. This sort of boundary condition is consistent with an interface with a very highly permeable metal.
- *Robin*. The Robin boundary condition is sort of a mix between Dirichlet and Neumann, prescribing a relationship between the value of A and its normal derivative at the boundary. An example of this boundary condition is:

$$\frac{\partial A}{\partial n} + cA = 0$$

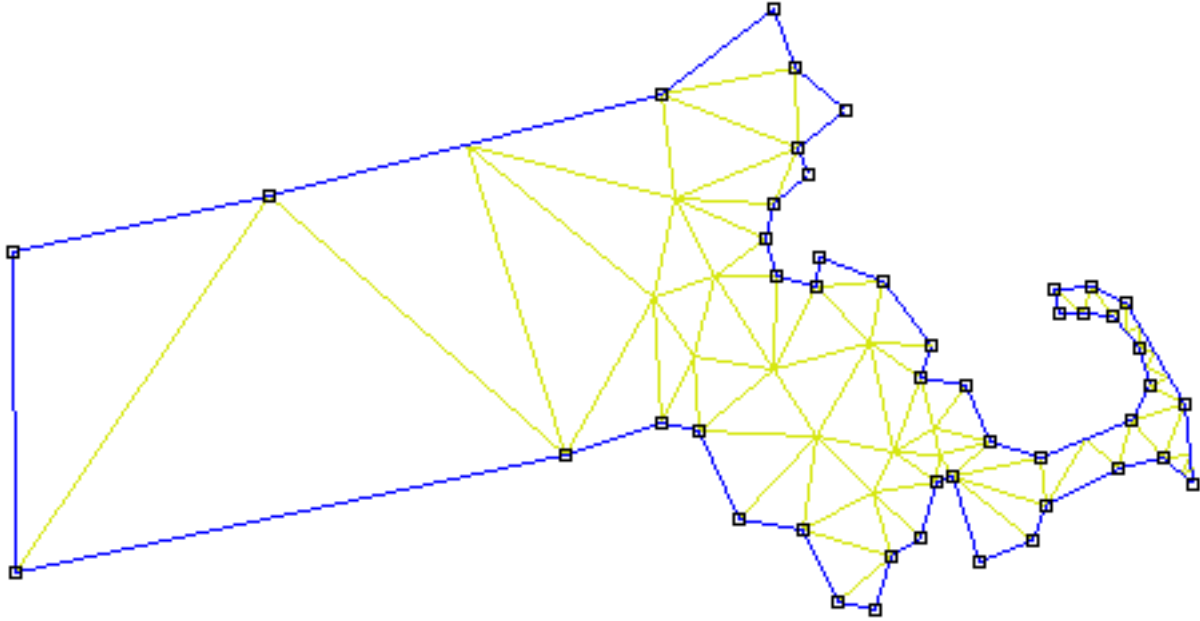


Figure 1: Triangulation of Massachusetts

This boundary condition is most often used by FEMM in eddy current problems on interfaces with bodies with small skin-depth eddy currents.

If no boundary conditions are explicitly defined, each boundary defaults to a $\partial A/\partial n = 0$ Neumann boundary condition. However, a non-derivative boundary condition must be defined somewhere so that the problem has a unique solution.

For axisymmetric problems, $A = 0$ is enforced on the line $r = 0$. In this case, a valid solution can be obtained without explicitly defining any boundary conditions, as long as part of the boundary of the problem lies along $r = 0$.

2.3 Finite Element Analysis

Although the differential equations that describe A appear relatively compact, it is very difficult to get closed-form solutions for all but the simplest geometries. That’s where finite element analysis comes in. The idea of finite elements is to break the problem down into a large number regions, each with a simple geometry (*e.g.* triangles). For example, Figure 1 shows a map of the Massachusetts broken down into triangles. Over these simple regions, the “true” solution for A is approximated by a very simple function. If enough small regions are used, the approximate A closely matches the exact A .

The advantage of breaking the domain down into a number of small elements is that the magnetics problem becomes transformed from a small but difficult to solve problem into a big but relatively easy to solve problem. Specifically, triangulation of the problem results in a linear algebra problem with perhaps tens of thousands of unknowns. However, techniques exist that allow the computer to solve for all the unknowns in only seconds.

FEMM uses triangular elements. Over each element, the solution is approximated by a linear interpolation of the values of A at the three vertices of the triangle. The linear algebra

problem is formed by choosing A on the basis of minimizing the total energy of the problem.

3 Preprocessor

The preprocessor is used for drawing the problems geometry, defining materials, and defining boundary conditions.

Drawing a valid geometry usually consists of four (though not necessarily sequential) tasks:

- Drawing the endpoints of the lines and arc segments that make up a drawing.
- Connecting the endpoints with either line segments or arc segments
- Adding “Block Label” markers into each section of the model to define material properties and mesh sizing for each section.
- Specifying boundary conditions on the outer edges of the geometry.

This section will describe exactly how one goes about performing these tasks and creating a problem that can be solved.

3.1 Preprocessor Drawing Modes

The key to using the preprocessor is that the preprocessor is always in one of five modes: the *Point* mode, the *Segment* mode, *Arc Segment* mode, the *Block* mode, or the *Group* mode. The first four of these modes correspond to the four types of entities that define the problems geometry: nodes that define all corners in the solution geometry, line segments and arc segments that connect the nodes to form boundaries and interfaces, and block labels that denote what material properties and mesh size are associated with each solution region. When the preprocessor is in a one of the first four drawing modes, editing operations take place only upon the selected type of entity. The fifth mode, the group mode, is meant to glue different objects together into parts so that entire parts can be manipulated more easily.

One can switch between drawing modes by clicking the appropriate button on the Drawing Mode portion of the toolbar. This section of the toolbar is pictured in Figure 2. The



Figure 2: Drawing Mode toolbar buttons.

buttons correspond to Point, Line Segment, Arc Segment, Block Label, and Group modes respectively. The default drawing mode when the program begins is the Point mode.

3.2 Keyboard and Mouse Commands

Although most of the tasks that need to be performed are available via the toolbar, some important functions are invoked only through the use of “hot” keys. A summary of these keys and their associated functions is contained in Table 1.

Point Mode Keys	
Key	Function
Space	Edit the properties of selected point(s)
Tab	Display dialog for the numerical entry of coordinates for a new point
Escape	Unselect all points
Delete	Delete selected points

Line/Arc Segment Mode Keys	
Key	Function
Space	Edit the properties of selected segment(s)
Escape	Unselect all segments and line starting points
Delete	Delete selected segment(s)

Block Label Mode Keys	
Key	Function
Space	Edit the properties of selected block labels(s)
Tab	Display dialog for the numerical entry of coordinates for a new label
Escape	Unselect all block labels
Delete	Delete selected block label(s)

Group Mode Keys	
Key	Function
Space	Edit group assignment of the selected objects
Escape	Unselect all
Delete	Delete selected block label(s)

View Manipulation Keys	
Key	Function
Left Arrow	Pan left
Right Arrow	Pan right
Up Arrow	Pan up
Down Arrow	Pan down
Page Up	Zoom in
Page Down	Zoom out
Home	Zoom “natural”

Table 1: FEMME hot keys

Likewise, specific functions are associated with mouse button input. The user employs the mouse to create new object, select objects that have already been created, and inquire

about object properties. Table 2 is a summary of the mouse button click actions.

Point Mode	
Action	Function
Left Button Click	Create a new point at the current mouse pointer location
Right Button Click	Select the nearest point
Right Button DblClick	Display coordinates of the nearest point

Line/Arc Segment Mode	
Action	Function
Left Button Click	Select a start/end point for a new segment
Right Button Click	Select the nearest line/arc segment
Right Button DblClick	Display length of the nearest arc/line segment

Block Label Mode	
Action	Function
Left Button Click	Create a new block label at the current mouse pointer location
Right Button Click	Select the nearest block label
Right Button DblClick	Display coordinates of the nearest block label

Group Mode	
Action	Function
Right Button Click	Select the group associated with the nearest object

Table 2: FEMME Mouse button actions

3.3 View Manipulation

Generally, the user needs to size or move the view of the problem geometry displayed on the screen. Most of the view manipulation commands are available via buttons on the preprocessor toolbar. The functionality of can generally also be accessed via the ‘View Manipulation Keys’ listed in Table 1. The View Manipulation toolbar buttons are pictured in Figure 3. The meaning of the View Manipulation toolbar buttons are:



Figure 3: View Manipulation toolbar buttons.

- The arrows on the toolbar correspond to moving the view in the direction of the arrow approximately 1/2 of the current screen width.

- The “blank page” button scales the screen to the smallest possible view that displays the entire problem geometry.
- The “+” and “-” buttons zoom the current view in and out, respectively.
- The “page with magnifying glass” button allows the view to be zoomed in on a user-specified part of the screen. To use this tool, first push the toolbar button. Then, move the mouse pointer to one of the desired corners of the “new” view. Press and hold the left mouse button. Drag the mouse pointer to the opposite diagonal corner of the desired “new” view. Last, release the left mouse button. The view will zoom in to a window that best fits the user’s desired window.

Some infrequently used view commands are also available, but only as options off of the **Zoom** selection of the main menu. This menu contains all of the manipulations available from the toolbar buttons, plus the options **Keyboard**, **Status Bar**, and **Toolbar**.

The **Keyboard** selection allows the user to zoom in to a window in which the window’s corners are explicitly specified by the user via keyboard entry of the corners’ coordinates. When this selection is chosen, a dialog pops up prompting for the locations of the window corners. Enter the desired window coordinates and hit “OK”. The view will then zoom to the smallest possible window that bounds the desired window corners. Typically, this view manipulation is only done as a new drawing is begun, to initially size the view window to convenient boundaries.

The **Status Bar** selection can be used to hide or show the one-line status bar at the bottom of the Femme window. Generally, it is desirable for the toolbar to be displayed, since the current location of the mouse pointer is displayed on the status line.

The **Toolbar** selection can be used to hide or show the toolbar buttons. The toolbar is not fundamentally necessary to running Femme, because any selection on the toolbar is also available via selections off of the main menu. If more space on the screen is desired, this option can be chosen to hide the toolbar. Selecting it a second time will show the toolbar again. It may be useful to note that the toolbar can be undocked from the main screen and made to “float” at a user-defined location on screen. This is done by pushing the left mouse button down on an area of the toolbar that is not actually a button, and then dragging the toolbar to its desired location. The toolbar can be docked again by moving it back to its original position.

3.4 Grid Manipulation

To aid in drawing your geometry, a useful tool is the Grid. When the grid is on, a grid of light blue pixels will be displayed on the screen. The spacing between grid points can be specified by the user, and the mouse pointer can be made to “snap” to the closest grid point.

The easiest way to manipulate the grid is through the used of the Grid Manipulation toolbar buttons. These buttons are pictured in Figure 4. The left-most button in Figure 4 shows and hides the grid. The default is that the button is pushed in, showing the current grid. The second button, with an icon of an arrow pointing to a grid point, is the “snap to grid” button. When this button is pushed in, the location of the mouse pointer is rounded



Figure 4: Grid Manipulation toolbar buttons.

to the nearest grid point location. By default, the “snap to grid” button is not pressed. The right-most button brings up the Grid Properties dialog. This dialog is shown in Figure 5.

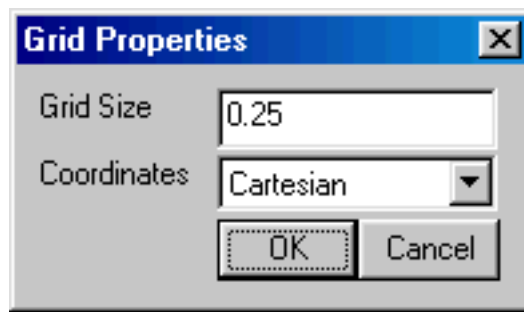


Figure 5: Grid Properties dialog.

The Grid Properties dialog has an edit box for the user to enter the desired grid sizing. When the box appears, the number in this edit box is the current grid size. The edit box also contains a drop list that allows the user to select between Cartesian and Polar coordinates. If Cartesian is selected, points are specified by their (x,y) coordinates for a planar problem, or by their (r,z) coordinates for an axisymmetric problem. If Polar is selected, points are specified by an angle and a radial distance from the origin. The default is Cartesian coordinates.

3.5 Edit

Several useful tasks can be performed via the **Edit** menu off of the main menu.

Perhaps the most frequently used is the **Undo** command. Choosing this selection undoes the last addition or deletion that the user has made to the model’s geometry.

For selecting many objects quickly, the **Select Group** command is useful. This command allows the user to select objects of the current type located in an arbitrary rectangular box. When this command is selected, move the mouse pointer to one corner of the region that is to be selected. Press and hold the left mouse button. Then, drag the mouse pointer to the opposite diagonal corner of the region. A red box will appear, outlining the region to be selected. When the desired region has been specified, release the left mouse button. All objects of the current type completely contained within the box will become selected.

Any objects that are currently selected can be moved, copied, or pasted. To move or copy selected objects, simply choose the corresponding selection off of the main menu’s **Edit** menu. A dialog will appear prompting for an amount of displacement or rotation.

3.6 Problem Definition

The definition of problem type is specified by choosing the **Problem** selection off of the main menu. Selecting this option brings up the Problem Definition dialog, shown in Figure 6

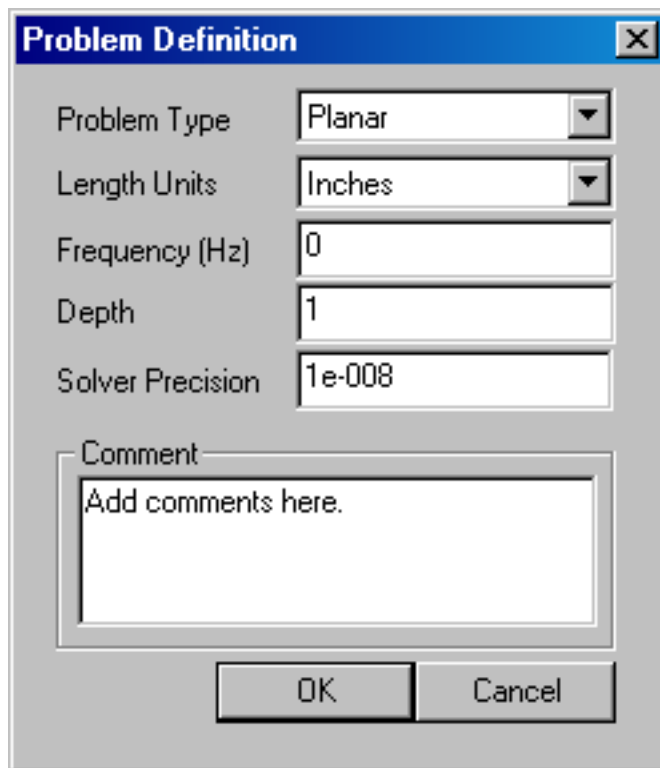


Figure 6: Problem Definition dialog.

The first selection is the **Problem Type** drop list. This drop box allows the user to choose from a 2-D planar problem (the **Planar** selection), or an axisymmetric problem (the **Axisymmetric** selection).

Next is the **Length Units** drop list. This box identifies what unit is associated with the dimensions prescribed in the model's geometry. Currently, the program supports inches, millimeters, centimeters, meters, mils, and μ meters.

The first edit box in the dialog is **Frequency (Hz)**. For a magnetostatic problem, the user should choose a frequency of zero. If the frequency is non-zero, the program will perform a harmonic analysis, in which all field quantities are oscillating at this prescribed frequency. The default frequency is zero.

The second edit box is the **Depth** specification. If a Planar problem is selected, this edit box becomes enabled. This value is the length of the geometry in the “into the page” direction. This value is used for scaling integral results in the post processor (*e.g.* force, inductance, etc.) to the appropriate length. The units of the **Depth** selection are the same as the selected length units. For files imported from version 3.2, the **Depth** is chosen so that the depth equals 1 meter, since in version 3.2, all results from planar problems are reported per meter.

The third edit box is the **Solver Precision** edit box. The number in this edit box specifies the stopping criteria for the linear solver. The linear algebra problem could be represented by:

$$Mx = b$$

where M is a square matrix, b is a vector, and x is a vector of unknowns to be determined. The solver precision value determines the maximum allowable value for $\|b - Mx\|/\|b\|$. The default value is 10^{-8} .

Lastly, there is an optional **Comment** edit box. The user can enter in a few lines of text that give a brief description of the problem that is being solved. This is useful if the user is running several small variations on a given geometry. The comment can then be used to identify the relevant features for a particular geometry.

3.7 Definition of Properties

To make a solvable problem definition, the user must identify boundary conditions, block materials properties, and so on. The different types of properties defined for a given problem are defined via the **Properties** selection off of the main menu.

When the **Properties** selection is chosen, a drop menu appears that has selections for **Materials**, **Boundary**, **Point**, and **Circuits**. When any one of these selections is chosen, the dialog pictured in Figure 7 appears. This dialog is the manager for a particular type of

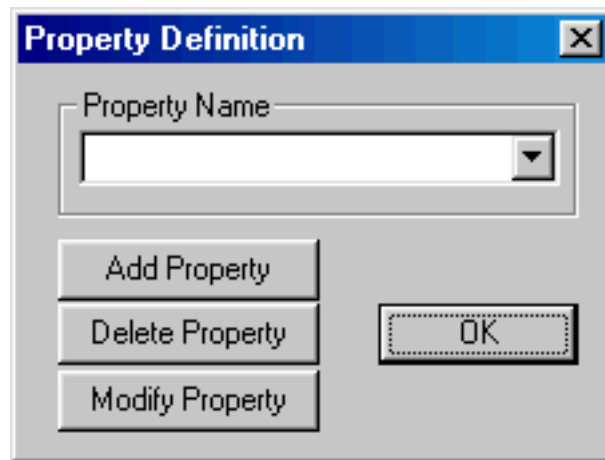


Figure 7: Property Definition dialog box

properties. All currently defined properties are displayed in the **Property Name** drop list at the top of the dialog. At the beginning of a new model definition, the box will be blank, since no properties have yet been defined. Pushing the **Add Property** button allows the user to define a new property type. The **Delete Property** button removes the definition of the property currently in view in the **Property Name** box. The **Modify Property** button allows the user to view and edit the property currently selected in the **Property Name** box. Specifics for defining **Point**, **Segment**, and **Block** properties are addressed in the following subsections.

In general, a number of these edit boxes prompt the user for both real and imaginary components for entered values. If the problem you are defining is magnetostatic (zero frequency), enter the desired value in the box for the real component, and leave a zero in the box for the imaginary component. The reason that Femme uses this formalism is to obtain a relatively smooth transition from static to time-harmonic problems. Consider the definition of the *Phasor transformation* in Eq. 14. The phasor transformation assumes that all field values oscillate in time at a frequency of ω . The phasor transformation takes the cosine part of the field value and represents it as the real part of a complex number. The imaginary part represents the magnitude of the sine component, 90° out of phase. Note what happens as the frequency goes to zero:

$$\lim_{\omega \rightarrow 0} (a_{re} \cos \omega t - a_{im} \sin \omega t) = a_{re} \quad (16)$$

Therefore, the magnetostatic ($\omega = 0$) values are just described by the real part the specified complex number.

3.7.1 Point Properties

If a new point property is added or an existing point property modified, the Nodal Property dialog box appears. This dialog box is pictured in Figure 8

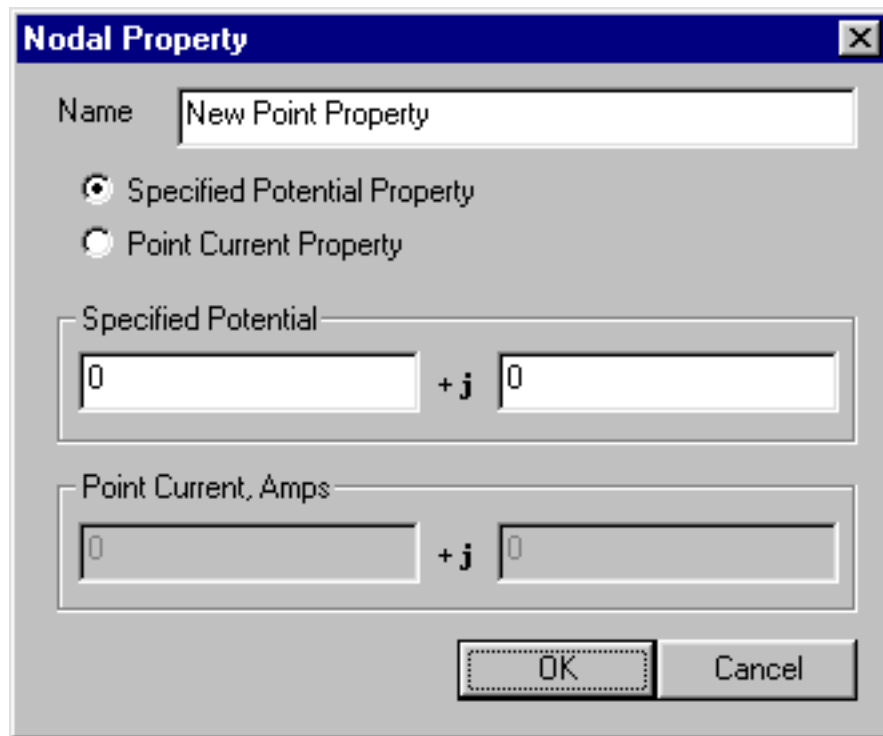


Figure 8: Nodal Property dialog.

The first selection is the **Name** edit box. The default name is “New Point Property,” but this name should be changed to something that describes the property that you are defining.

Next are edit boxes for defining the vector potential, A , at a given point, or prescribing a point current, J , at a given point. The two definitions are mutually exclusive. Therefore, if there are nonzero values in either of the J boxes, the program assumes that a point current is being defined. Otherwise, it is assumed that a point vector potential is being defined.

There are two edit boxes for the definition of the real and imaginary parts of A , the magnetic vector potential. The units of A are understood to be Weber/Meter. Typically, A needs to be defined as some particular values (usually zero) at some point in the solution domain for problems with derivative boundary conditions on all sides. This is the typical use of defining a point vector potential.

Lastly, there are two edit boxes for the definition of a point current, J . The units for the point current are understood to be in Amperes.

3.7.2 Boundary Properties

The Boundary Property dialog box is used to specify the properties of line segments or arc segments that are to be boundaries of the solution domain. When a new boundary property is added or an existing property modified, the Boundary Property dialog pictured in Figure 9 appears.

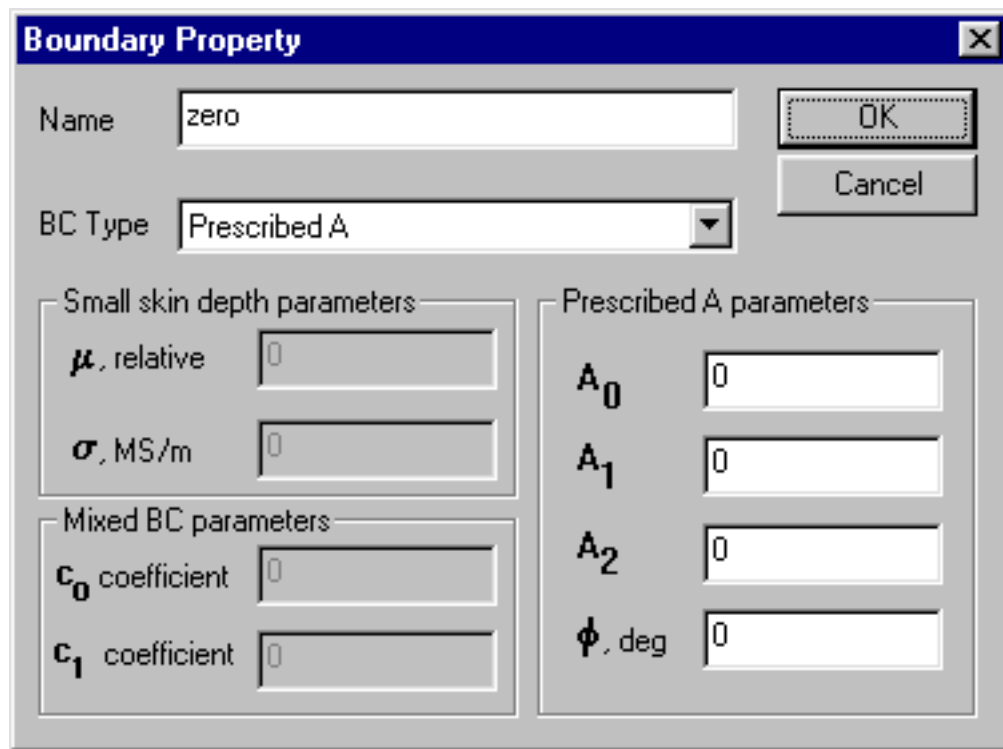


Figure 9: Boundary Property dialog.

The first selection in the dialog is the Name of the property. The default name is “New Boundary,” but you should change this name to something more descriptive of the boundary that is being defined.

The next selection is the **BC Type** drop list. This specifies the boundary condition type. Currently, FEMM supports the following types of boundaries:

- **Prescribed A** With this type of boundary condition, the vector potential, A , is prescribed along a given boundary. This boundary condition can be used to prescribe the flux passing normal to a boundary, since the normal flux is equal to the tangential derivative of A along the boundary. The form for A along the boundary is specified via the parameters A_0 , A_1 , A_2 and ϕ in the **Prescribed A parameters** box. If the problem is planar, the parameters correspond to the formula:

$$A = (A_0 + A_1x + A_2y)e^{j\phi} \quad (17)$$

If the problem type is axisymmetric, the parameters correspond to:

$$A = (A_0 + A_1r + A_2z)e^{j\phi} \quad (18)$$

- **Small Skin Depth** This boundary condition denotes an interface with a material subject to eddy currents at high enough frequencies such that the skin depth in the material is very small. A good discussion of the derivation of this type of boundary condition is contained in [2]. The result is a Robin boundary condition with complex coefficients of the form:

$$\frac{\partial A}{\partial n} + \left(\frac{1+j}{\delta}\right)A = 0 \quad (19)$$

where the n denotes the direction of the outward normal to the boundary and δ denotes the skin depth of the material at the frequency of interest. The skin depth, δ is defined as:

$$\delta = \sqrt{\frac{2}{\omega\mu_r\mu_o\sigma}} \quad (20)$$

where μ_r and σ are the relative permeability and conductivity of the thin skin depth eddy current material. These parameters are defined by specifying μ and σ in the **Small skin depth parameters** box in the dialog. At zero frequency, this boundary condition degenerates to $\partial A/\partial n = 0$ (because skin depth goes to infinity).

- **Mixed** This denotes a boundary condition of the form:

$$\left(\frac{1}{\mu_r\mu_o}\right)\frac{\partial A}{\partial n} + c_oA + c_1 = 0 \quad (21)$$

The parameters for this class of boundary condition are specified in the **Mixed BC parameters** box in the dialog. By the choice of coefficients, this boundary condition can either be a Robin or a Neumann boundary condition. There are two main uses of this boundary condition:

1. By carefully selecting the c_0 coefficient and specifying $c_1 = 0$, this boundary condition can be applied to the outer boundary of your geometry to approximate an unbounded solution region. For more information on open boundary problems, refer to Appendix A.3.

2. The Mixed boundary condition can be used to set the field intensity, H , that flows parallel to a boundary. This is done by setting c_0 to zero, and c_1 to the desired value of H in units of Amp/Meter. Note that this boundary condition can also be used to prescribe $\partial A/\partial n = 0$ at the boundary. However, this is unnecessary—the 1st order triangle finite elements give a $\partial A/\partial n = 0$ boundary condition by default.
- **Strategic Dual Image** This is sort of an “experimental” boundary condition that I have found useful for my own purposes from time to time. This boundary condition mimics an “open” boundary by solving the problem twice: once with a homogeneous Dirichlet boundary condition on the SDI boundary, and once with a homogeneous Neumann condition on the SDI boundary. The results from each run are then averaged to get the open boundary result. This boundary condition should only be applied to the outer boundary of a circular domain in 2-D planar problems. Through a method-of-images argument, it can be shown that this approach yields the correct open-boundary result for problems with no iron (*i.e.* just currents or linear magnets with unit permeability in the solution region).
 - **Periodic** This type of boundary condition is applied to either two segments or two arcs to force the magnetic vector potential to be identical along each boundary. This sort of boundary is useful in exploiting the symmetry inherent in some problems to reduce the size of the domain which must be modeled. The domain merely needs to be periodic, as opposed to obeying more restrictive $A = 0$ or $\partial A/\partial n = 0$ line of symmetry conditions. Another useful application of periodic boundary conditions is for the modeling of “open boundary” problems, as discussed in Appendix A.3.3. Often, a periodic boundary is made up of several different line or arc segments. A different periodic condition must be defined for each section of the boundary, since each periodic BC can only be applied to a line or arc and a corresponding line or arc on the remote periodic boundary.
 - **Antiperiodic** The antiperiodic boundary condition is applied in a similar way as the periodic boundary condition, but its effect is to force two boundaries to be the negative of one another. This type of boundary is also typically used to reduce the domain which must be modeled, *e.g.* so that an electric machine might be modeled for the purposes of a finite element analysis with just one pole.

3.7.3 Materials Properties

The **Block Property** dialog box is used to specify the properties to be associated with block labels. The properties specified in this dialog have to do with the material that the block is composed of, as well as some attributes about how the material is put together (laminated). When a new material property is added or an existing property modified, the **Block Property** dialog pictured in Figure 10 appears.

As with Point and Boundary properties, the first step is to choose a descriptive name for the material that is being described. Enter it in the **Name** edit box in lieu of “New Material.”

Next decide whether the material will have a linear or nonlinear B-H curve by selecting the appropriate entry in the **B-H Curve** drop list.

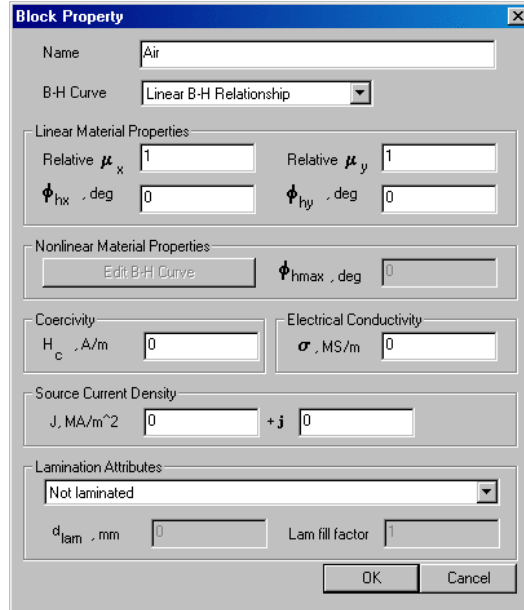


Figure 10: Block Property dialog.

If **Linear B-H Relationship** was selected from the drop list, the next group of **Linear Material Properties** parameters will become enabled. FEMM allows you to specify different relative permeabilities in the vertical and horizontal directions (μ_x for the x- or horizontal direction, and μ_y for the y- or vertical direction). There are also boxes for ϕ_{hx} and ϕ_{hy} , which denote the hysteresis lag angle corresponding to each direction.

If **Nonlinear B-H Curve** was selected from the drop list, the **Nonlinear Material Properties** parameter group becomes enabled. To enter in points on your B-H curve, hit the **Edit B-H Curve** button. When the button is pushed a dialog appears that allows you to enter in B-H data (see Figure 11). The information to be entered in these dialogs is usually obtained by picking points off of manufacturer’s data sheets. For obvious reasons, you must enter the same number of points in the “B” (flux density) column as in the “H” (field intensity) column. To define a nonlinear material, you must enter *at least* three points, and you should enter ten or fifteen to get a good fit.

After you are done entering in your B-H data points, it is a good idea to view the B-H curve to see that it looks like it is “supposed” to. This is done by pressing the **Plot B-H Curve** button in the B-H data dialog. You should see a B-H curve that looks something like the curve pictured in Figure 12. The boxes in the plot represent the locations of the entered B-H points, and the line represents a cubic spline derived from the entered data. Since FEMM interpolates between your B-H points using cubic splines, it is possible to get a bad curve if you haven’t entered an adequate number of points. “Weird” B-H curves can result if you have entered too few points around relatively sudden changes in the B-H curve. FEMM “takes care of” bad B-H data (*i.e.* B-H data that would result in a cubic spline fit that is not single-valued) by repeatedly smoothing the B-H data using a three-point moving average filter until a fit is obtained that is single-valued. This approach is robust in the sense that it always yields a single-valued curve, but the result might be a poor match to the

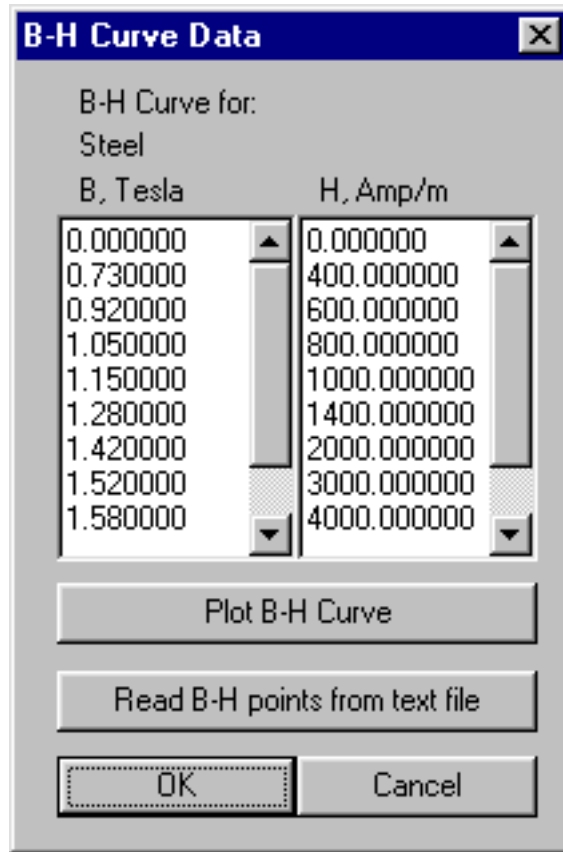


Figure 11: B-H data entry dialog.

original B-H data. Adding more data points in the sections of the curve where the curvature is high helps to eliminate the need for smoothing.

It may also be important to note that FEMM extrapolates linearly off the end of your B-H curve if the program encounters flux density/field intensity levels that are out of the range of the values that you have entered. This extrapolation may make the material look more permeable than it “really” is at high flux densities. You may have to be careful to enter enough B-H values to get an accurate solution in highly saturated structures (so that the program is interpolating between your entered data points, rather than extrapolating).

Also in the nonlinear parameters box is a parameter, ϕ_{hmax} . For nonlinear problems, the hysteresis lag is assumed to be proportional to the effective permeability. At the highest effective permeability, the hysteresis angle is assumed to reach its maximal value of ϕ_{hmax} . This idea can be represented by the formula:

$$\phi_h B = \left(\frac{\mu_{eff}(B)}{\mu_{eff,max}} \right) \phi_{hmax} \quad (22)$$

The next entry in the dialog is H_c . If the material is a permanent magnet, you should enter the magnet’s coercivity here in units of Amperes per meter. There are some subtleties involved in defining permanent magnet properties (especially nonlinear permanent magnets). Please refer to the Appendix A.1 for a more thorough discussion of the modeling of permanent magnets in FEMM.

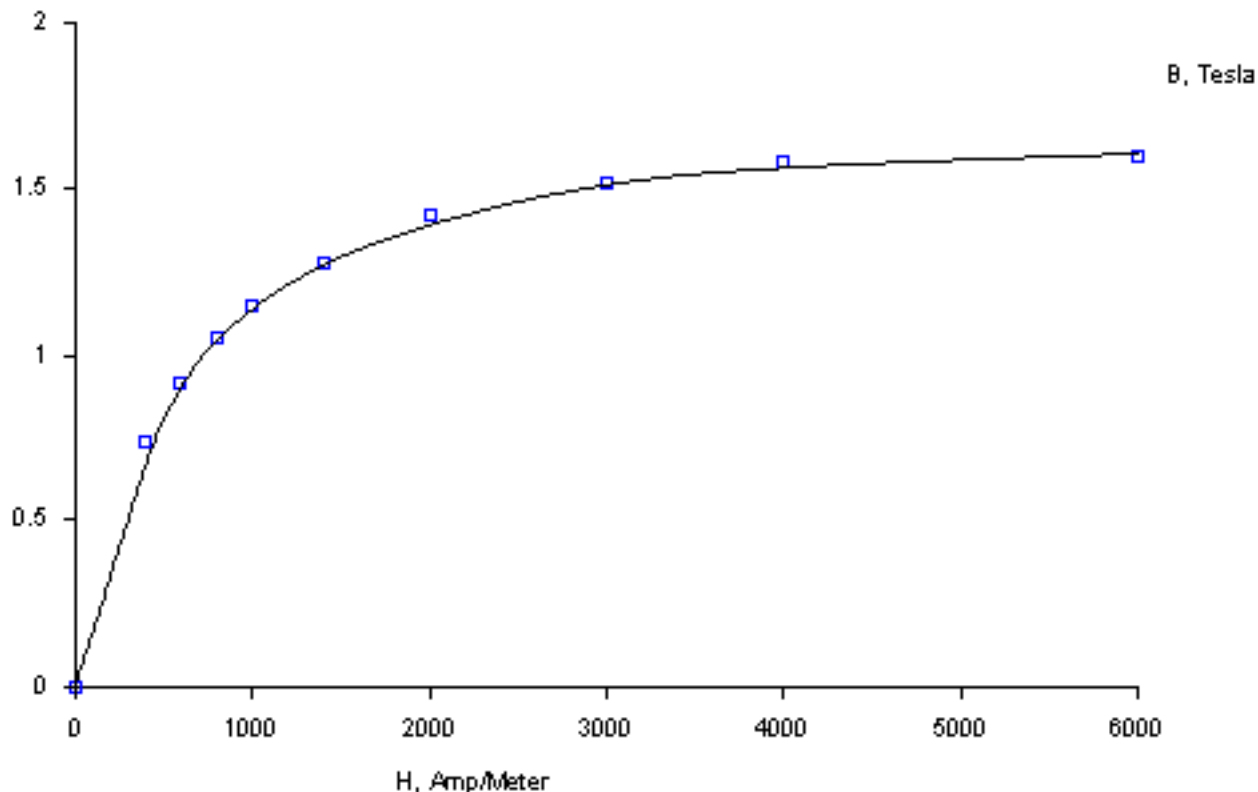


Figure 12: Sample B-H curve plot.

The next entries, the real and imaginary parts of J , represent the applied current density in the block. The usual rules for quantities with real and imaginary parts apply to J .

The σ edit box denotes the electrical conductivity of the material in the block. This value is generally only used in time-harmonic (eddy current) problems. The units for conductivity are 10^6 Seymens/Meter (equivalent to $10^6(\Omega * \text{Meters})^{-1}$). For reference, copper at room temperature has a conductivity of 58 MS/m; a good silicon steel for motor laminations might have a conductivity of as low as 2 MS/m. Commodity-grade transformer laminations are more like 9 MS/m. You should note that conductivity generally has a strong dependence upon temperature, so you should choose your values of conductivity keeping this caveat in mind. Lastly, bulk sections of coil should be defined to have a conductivity of zero for eddy current problems—thin wires impede the formation of eddy currents inside the wires until high frequencies. You will get spurious results if a nonzero conductivity is used for bulk coils.

Another edit box that is used only for harmonic problems is the ϕ_h edit box. This stands for the hysteresis lag angle. A simple, but surprisingly effective, model for hysteresis in harmonic problems is to assume that hysteresis creates a constant phase lag between B and H that is independent of frequency. This is exactly the same as assuming that hysteresis loop has an elliptical shape. Since the hysteresis loop is not exactly elliptical, the perceived hysteresis angle will vary somewhat for different amplitudes of excitation. The hysteresis angle is typically not a parameter that appears on manufacturer's data sheets; you have to

identify it yourself from a frequency sweep on a toroidal coil with a core composed of the material of interest. For most laminated steels, the hysteresis angle lies between 0° and 20° [4]. This same reference also has a very good discussion of the derivation and application of the fixed phase lag model of hysteresis.

The d_{lam} edit box represents the thickness of the laminations used for this type of material. If the material is not laminated, enter 0 in this edit box. Otherwise, enter the thickness of *just the iron part* (not the iron plus the insulation) in this edit box in units of millimeters.

Associated with the lamination thickness edit box is the **Lam fill factor** edit box. This is the fraction of the core that is filled with iron. For example, if you had a lamination in which the iron was 12.8 mils thick, and the insulation between laminations was 1.2 mils thick, the fill factor would be:

$$\text{Fill Factor} = \frac{12.8}{1.2 + 12.8} = 0.914$$

The last selection is the **Lam orientation** drop list. This information is used to denote the direction in which the material is laminated. The various selections in this list are illustrated in Figure 13. Currently, the laminations are constrained to run along a particular

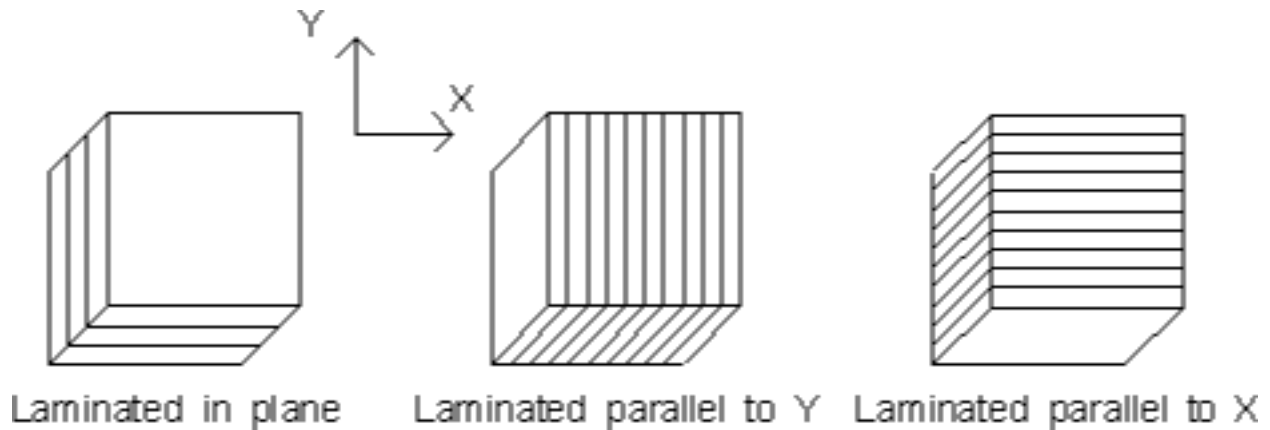


Figure 13: Different lamination orientation options.

axis.

The lamination thickness, fill factor, and lamination orientation parameters are used to implement a bulk model of laminated material. The result of this model is that one can account for laminations with hysteresis and eddy currents in harmonic problems. For magnetostatic problems, one can approximate the effects of nonlinear laminations without the necessity of modeling the individual laminations separately. This bulk lamination model is discussed in more detail in the Appendix (Section A.2).

3.7.4 Materials Library

Since one kind of material might be needed in several different models, FEMM has a built-in library of Block Property definitions. The user can access and maintain this library through the **Properties | Materials Library** selection off of the main menu. When this option

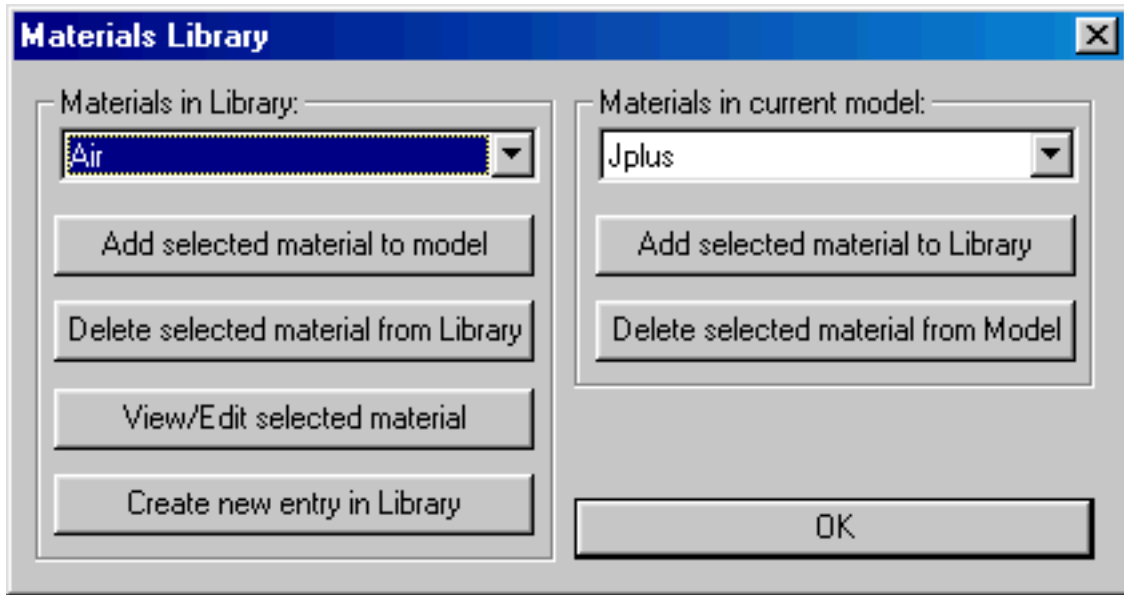


Figure 14: Materials Library dialog.

is selected, the `Materials Library` dialog pictured in Figure 14 appears. The options on this dialog allow the user to exchange Block Property definitions between the current model and the materials library.

The materials library should be located in the same directory as the FEMM executable files, under the filename `mlibrary.dat`. If you move the materials library, `femme` will not be able to find it.

3.7.5 Circuit Properties

The definition of “Circuit” properties is a “new” addition to version 3 of the program. The purpose of the circuit properties is mainly to allow the user to apply constraints on the total amount of current flowing in one or more blocks.

For magnetostatic problems, one could alternatively apply a source current density over the conductor of interest and achieve similar results. For eddy current problems, however, the “circuit” properties are much more useful—they allow the user to apply constraints on the *total* current (*i.e.* the source current plus eddy currents), whereas source current densities are just one component of the currents for harmonic problems.

By applying circuit properties, one can also enforce connectivity in eddy current problems. By default, all objects in eddy current problems are “shorted together at infinity”—that is, there is nothing to stop induced currents from returning in other sections of the domain that might not be intended to be physically connected. By applying a zero net current density “circuit” constraint to each physical “part” in the geometry, the connectivity of each part is enforced and all is forced to be conserved inside the part of interest.

The dialog for entering circuit properties is pictured in Figure 15. Instead of imposing constraints on current, the circuit properties can also be used to apply a prescribed voltage gradient over a block or group of blocks.

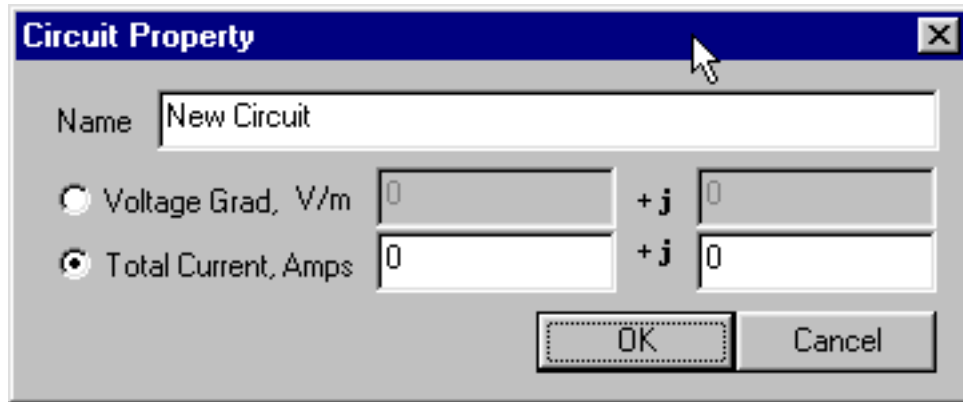


Figure 15: Circuit Property dialog

3.8 Spawned Tasks

To actually mesh the model, analyze the model, and view the results, the femme editor must spawn external programs. These tasks are most easily performed by the toolbar buttons pictured in Figure 16



Figure 16: Toolbar buttons for spawning external tasks.

The first of these buttons (with the “yellow mesh” icon) runs the mesh generator. The solver actually automatically calls the mesh generator to make sure that the mesh is up to date, so you never *have* to call the mesher from within femme. However, it is almost always important to get a look at the mesh and see that it “looks right.” When the mesh generation button is pressed, the mesher is called. While the mesher is running, an entry labeled “triangle” will appear on the Windows taskbar. Triangle is actually a console application that runs in a minimized window. After the geometry is triangulated, the finite element mesh is loaded into memory and displayed underneath the defined nodes, segments, and block labels as a set of yellow lines.

If you have a very large model, just keeping all of the mesh information in core can take up a significant amount of memory. If you are about to analyze a very large problem, it might be a good idea to choose the **Mesh | Purge Mesh** option off of the main menu. When this option is selected, the mesh is removed from memory, and the memory that it occupied is freed for other uses.

The second button, with the “hand-crank” icon, executes the solver, `fkern.exe`. Before `fkern` is actually run, the Triangle is called to make sure the mesh is up to date. Then, `fkern` is called. When `fkern` runs, it opens up a console window to display status information to the user. However, `fkern` requires no user interaction while it is running. When `fkern` is finished analyzing your problem, the console window will disappear. The time that `fkern` requires is highly dependent on the problem being solved. Solution times can range from less than a second to several hours, depending upon the size and complexity of the problem.

Generally, linear magnetostatic problems take the least amount of time. Harmonic problems take slightly more time, because the answer is in terms of complex numbers. The complex numbers effectively double the number of unknowns as compared to a magnetostatic problem with the same mesh. The slowest problems to analyze are nonlinear magnetostatic problems, since multiple iterations must be used to converge on the final solution. However, nonlinear problems almost never take more than 10 iterations. Later iterations in nonlinear problems are usually quite fast compared to the first iteration or two because the later iterations can be initialized with an approximate solution that is very close to the “actual” solution.

For users who have a technical interest in what is “under the hood” in `fkern`, some details are provided in the Appendix (Section 6).

The “big magnifying glass” icon is used to run the postprocessor once the analysis is finished. A detailed description of the postprocessor addressed in Section 4.

3.9 DXF Import/Export

For interfacing with CAD programs and other finite element packages, `femme` supports the import and export of the AutoCAD dxf file format. Specifically, the dxf interpreter in `femme` was written to the dxf revision 13 standards. Only 2D dxf files can be imported in a meaningful way.

To import a dxf file, select **Import DXF** off of the **File** menu. A dialog will appear after the file is selected asking for a tolerance. This tolerance is the maximum distance between two points at which the program considers two points to be the same. The default value is usually sufficient. For some files, however, the tolerance needs to be increased to import the file correctly. `Femme` does not understand all the possible tags that can be included in a dxf file; instead, it simply strips out the commands involved with drawing lines, circles, and arcs. All other information is simply ignored. Generally, dxf import is a useful feature. It allows the user to draw an initial geometry using their favorite CAD package. Once the geometry is laid out, the geometry can be imported into `femme` and detailed for materials properties and boundary conditions.

Do not despair if `femme` takes a while to import dxf files (especially large dxf files). The reason that `femme` can take a long time to import dxf files is that a lot of consistency checking must be performed to turn the dxf file into a valid FEMM geometry. For example, large dxf files might take up to a minute or two to import.

The current `femme` geometry can be exported in dxf format by selecting the **Export DXF** option off of the **File** menu. The dxf files generated from `femme` can then be imported into CAD programs to aid in the mechanical detailing of a finalized magnetic design, or imported into other finite element or boundary element programs.

4 Postprocessor

The executable `femview.exe` is the postprocessor used to view solutions generated by the `fkern` solver. This program can either be run on its own, from the “Start” menu (to view previously solved problems), or spawned from within `femme` to view a newly generated solution. Data files for `femview` have the `.ans` prefix.

4.1 Postprocessor modes

Similar to the preprocessor, the postprocessor always operates in one of three modes, depending upon the task to be performed. These modes are:

- *Point Values Mode* In this mode, the user can click on various points in the solution region. Local field values are then listed in the `Femview Output` window.
- *Contour Mode* This mode allows the user to define arbitrary contours in the solution region. Once a contour is defined, plots of field quantities can be produced along the contour, and various line integrals can be evaluated along the contour.
- *Block Mode* The Block Mode lets the user define a subdomain in the solution region. Once the block has been defined, a variety of area and volume integrals can be taken over the defined subdomain. Integrals include stored energy (inductance), various kinds of losses, total current in the block, and so on.

The current postprocessor mode is controlled via the Analysis Mode toolbar buttons, shown in Figure 17. The buttons denote, respectively, Point Values mode, Contour Mode,



Figure 17: Analysis Mode toolbar buttons

and Block Mode. The depressed button denotes the current mode. The default mode when `femview` starts is Point Values mode.

4.2 View and Grid Manipulation

The aspects of the current view and of the current grid are regulated via the use of toolbar buttons. The view is manipulated by the following toolbar buttons: and the grid settings are



Figure 18: View Manipulation toolbar buttons.

manipulated by these grid manipulation toolbar buttons: The grid and view manipulation work in exactly the same fashion as these same features in the preprocessor. Refer to Section 3.4 for a detailed description of grid manipulation, and to Section 3.3 for view manipulation.



Figure 19: Grid Manipulation toolbar buttons.

4.3 Keyboard Commands

Unlike the preprocessor, femmview is not very dependent on keyboard commands.

In the Point Values mode, there is only one relevant keypress. In this mode, the Tab key allows the user to enter the coordinates of a specific point. After the coordinates of a point are entered, the field values at that point are displayed in the `Femmview Output` window.

In the Contours, there are four relevant keys. Pressing the Escape key wipes out any current contour or block definition. Pressing the Delete removes the last point added to the current contour or block edge. Pressing the Shift key allows the user to turn the last segment in the prescribed contour from a straight line into an arc. A dialog pops up after the key is pressed that prompts for attributes of the the desired arc. Last, pressing the Tab keys allows the user to numerically enter the coordinates of a point to be included in the current contour.

In the Block mode, the Escape and Delete keys have the same definition as in Contours mode. In the Block mode, the Tab key does not do anything, since all points on the contour must also be Points defining the model's geometry.

4.4 Mouse Actions

In contrast, the operation of the postprocessor is very dependent upon input from the mouse.

In the Point Values mode, a Left Button Click is used to display field values at the current mouse location. If Snap to Grid is enabled, values are displayed at the closes grid point instead.

In the Contours mode, mouse clicks are used to define the contour. A Left Button Click adds the closest Point in the model's geometry. Via a Right Button Click, the current mouse pointer location is added to the contour. A contour appears as a red line on the screen.

Blocks are defined in Block mode in a fashion very similar to the way in which contours are defined. A block is defined by by drawing a contour around the region of interest. The contour appears as a green line on the femmview screen. When the ends of the contour meet, the block is defined. All elements enclosed by the contour (all elements that form the block) then turn green in the femmview window.

A Left Button Click attempts to add the nearest Point in the input geometry to the Block's contour. However, a block can only be defined along line and arc segments from the input geometry. Each node on the boundary of the block must be selected in order to form the Block boundary. In Block mode, the Right mouse button has no function.

4.5 Contour Plot

One of the most useful ways to get a subjective feel for a magnetics finite element solution is by plotting the "flux lines." These are the streamlines along which flux flows in the finite



Figure 20: Graph Mode toolbar buttons.

element geometry. Where flux lines are close together, the flux density is high. In FEMM's vector potential formulation, flux lines are simply plots of the level contours of the vector potential, A , in planar problems, or level contours of $2\pi rA$ in axisymmetric problems.

For harmonic problems, the contours are a little more subtle— A has both real and imaginary components. In this case, femmview allows the user to plot contours of either the real or the imaginary part of A . Real contours appear black, and Imaginary contours appear as grey.

By default, a set of 19 flux lines are plotted when a solution is initially loaded into femmview. The number and type of flux lines to be plotted can be altered using the Contours Plot icon in the Graph Mode section of the toolbar (see Figure 20). The Contour Plot icon is the icon with the black contours. When this button is pressed, a dialog pops up, allowing the choice of the number of contours (between 4 and 100 are allowed), and which contours to plot (either real, imaginary, or none).

In the contour plot dialog, a check box is also present titled “Show stress tensor mask”. If this box is checked, the contour lines associated with the last Weighted Stress Tensor integration are also displayed, by default as orange flux lines.

4.6 Density Plot

Density plots are also a useful way to get a quick feel for the flux density in various parts of the model. By default, a flux density plot is not displayed when femmview first starts. However, the plot can be displayed by pressing the middle button in the Graph Mode section of the toolbar (see Figure 20). A dialog the pops up that allows the user to turn density plotting on. If the solution is to a harmonic problem, the user can choose to plot either the magnitude of the flux density or just the real or imaginary part of the flux density.

The flux density at each point is classified into one of 12 contours distributed evenly between either the minimum and maximum flux densities or user-specified bounds. An example plot of an air-cored coil with both contour and density plotting turned on is shown in Figure 21.

4.7 Line Plots

When femmview is in Contours Mode, various field values of interest can be plotted along the defined contour. A plot of a field value defined contour is performed by pressing the “graphed function” icon in the Plot and Integration group of toolbar buttons, shown in Figure 22. When this button is pressed, the X-Y Plot dialog (see Figure 23) appears with a drop list containing the types of line plots available. Choose the desired type of plot and press “OK.”

After “OK” is pressed, the program computes the desired values along the defined contour. These values are then plotted using the `femmpLOT` program, which is called automati-

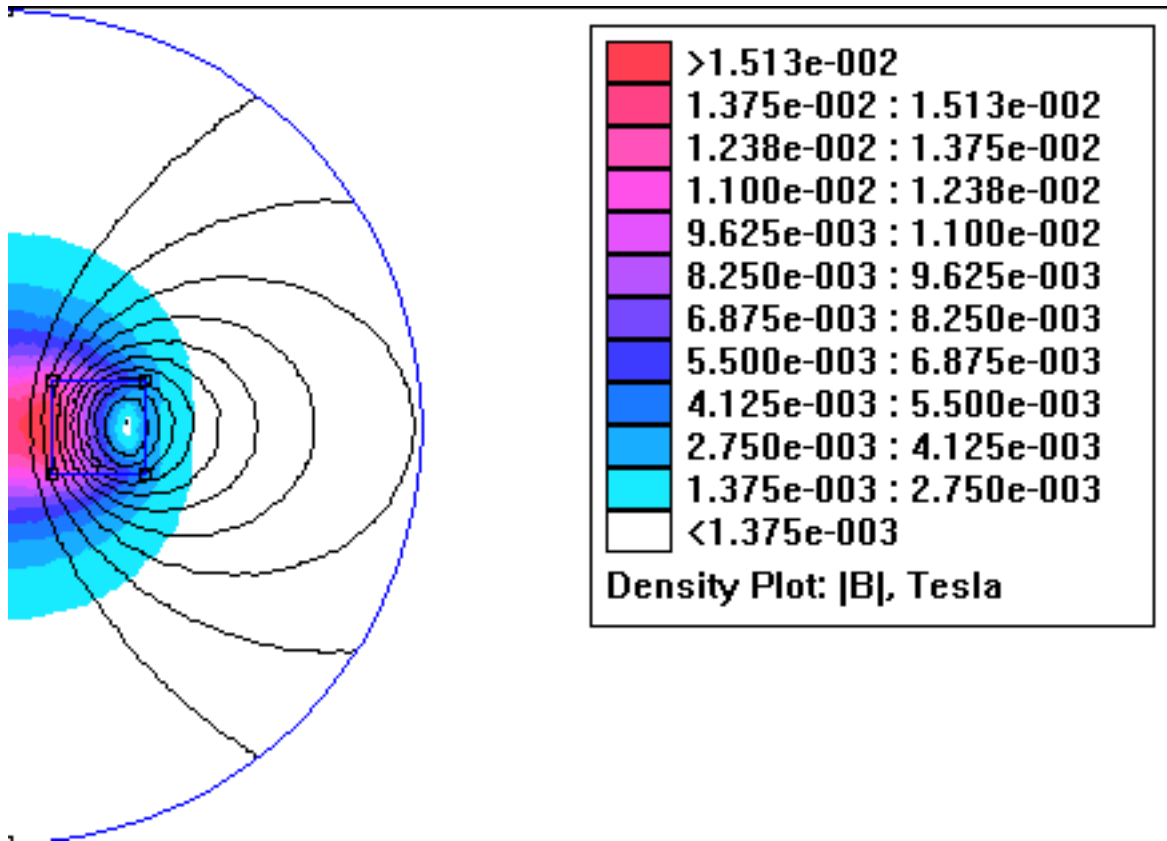


Figure 21: Solution for an air-cored coil with both Contour Lines and Density Plot



Figure 22: Line Plot and Integration toolbar buttons

cally to display the plot.

By default, the `Write data to text file` box is not checked. If the user selects this option, the file selection dialog will appear and prompt for a filename to which to write the data. The data is written in two-column text format. If `Write data to text file` is selected, a femmplot window will not appear.

Currently, the type of line plots supported are:

- Vector potential along the contour;
- Magnitude of the flux density along the contour;
- Component of flux density normal to the contour;
- Component of flux density tangential to the contour;
- Magnitude of the field intensity along the contour;
- Component of field intensity normal to the contour;

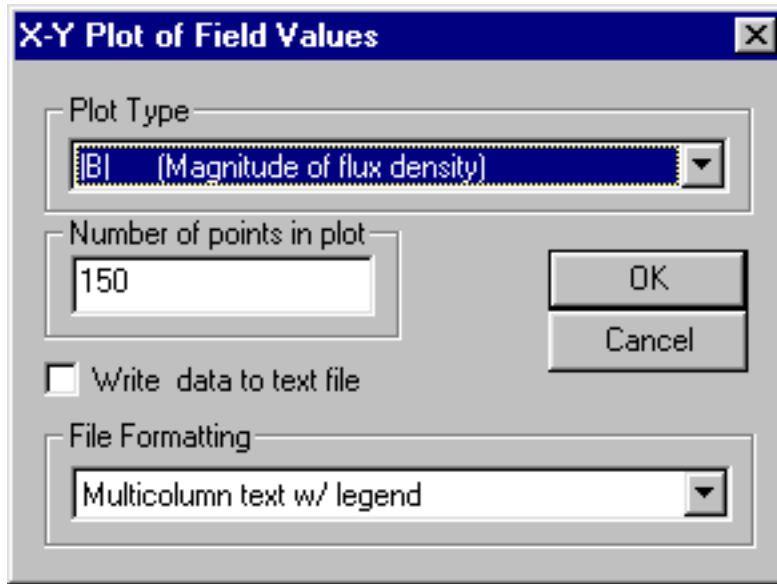


Figure 23: X-Y Plot dialog.

- Component of field intensity tangential to the contour;

In all of these plots, the direction of the normal is understood to be as shown in Figure 24. The tangential direction is understood to be the direction in which the contour was defined.

In certain cases, the quantity to be plotted can be ambiguous. This can occur, for example, if a plot of the tangential field intensity is requested on a contour running along an interface between air and a piece of iron. In this case, there is a discontinuity in the tangential field intensity, and the value of this quantity is different on each side of the interface. Femmview resolves the conflict by always evaluating the plots at a differentially small distance to the “normal” side of the line. Therefore, by defining the same contour but reversing the order in which the points are specified, plots of the quantity of interest on each side of a boundary can be obtained.

4.8 Line Integrals

Once a contour has been specified in Contours mode, Line Integrals can be performed along the specified contour. These integrals are performed by evaluating a large number of points at evenly spaced along the contour and integrating using a simple trapezoidal-type integration scheme.

To perform an integration, press the “integral” icon on the toolbar (as shown in Figure 22). A small dialog will appear with a drop list. Choose the desired integral from the drop list and press OK. The amount of time required to perform the integral will be virtually instantaneous for some types of integrals; however, some types may require several seconds to evaluate. When the evaluation of the integral is completed, the answer appears on the screen in a pop-up box.

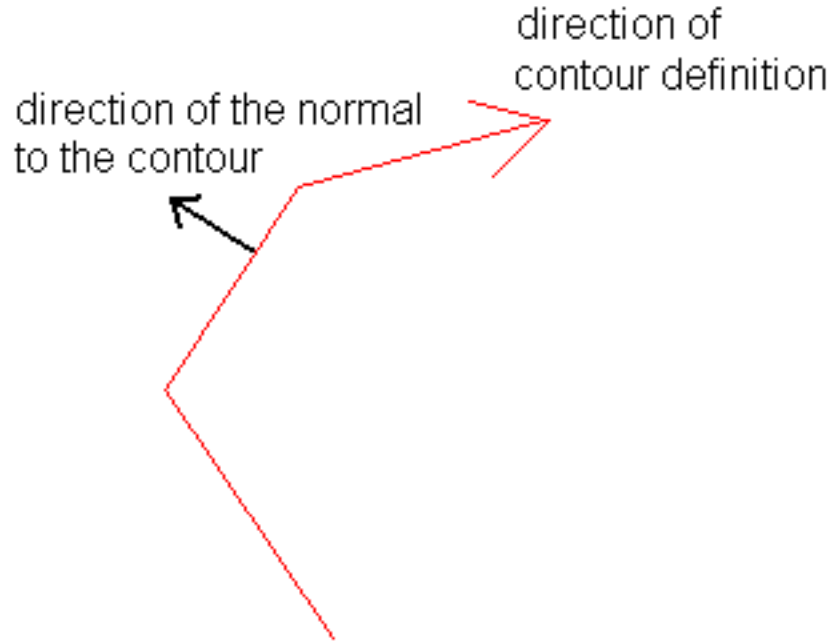


Figure 24: When in doubt plots and integrals taken on this side of a contour.

One “tip” that may aid with the definition of contours of integration is that a curved integration contour can be defined in a fairly painless fashion by hitting the Shift key. Hitting the Shift key tells the program to turn the last segment in the defined integration contour into an arc segment. A dialog then pops up that prompts for the desired attributes of the arc.

The line integrals currently supported are:

- **B.n.** This integral returns the total flux passing normal to the contour. This integral is useful for determining the total flux in a bulk flux path. This result might then be compared to predictions from a simpler magnetic circuit model, for example.
- **H.t.** The integral of the tangential field intensity along the contour yields the magnetomotive force drop between the endpoints of the contour. Again, this integral is useful for comparison to or validation of magnetic circuit models.
- **Contour Length.** This integral returns the length of the defined contour in meters.
- **Force from stress tensor.** This integral totals the force produced on the contour derived from Maxwell’s stress tensor. Deriving meaningful force results requires some care in the choice of integration path; refer to Section 4.10 for a detailed discussion of force and torque calculation.
- **Torque from stress tensor.** This selection integrates the torque about the point (0,0) inferred from Maxwell’s stress tensor. Again, some guidelines must be followed to get accurate torque results (see Section 4.10).

- **B.n²**. This selection evaluates the integral of the square of the normal flux along the line. This integral is not so commonly used, but it has been useful in the past for some specialized purposes, like determining the RMS amplitude of a periodic flux distribution.

4.9 Block Integrals

Once a closed contour has been specified in Block mode and the block appears highlighted in green, Block Integrals over the specified area. These integrals are performed by analytically integrating the specified kernel over each element in the defined region, and summing the results for all elements.

To perform an integration, press the “integral” icon on the toolbar (as shown in Figure 22). A small dialog will appear with a drop list. Choose the desired integral from the drop list and press OK. Generally, volume integrals take several seconds to evaluate, especially on dense meshes. Be patient. When the evaluation of the integral is completed, the answer appears on the screen in a pop-up box.

The block integrals currently supported are:

- **A.J** This integral is performed to evaluate inductance for linear problems. Generally, the self-inductance of a coil is:

$$L_{self} = \frac{\int A \cdot J dV}{i^2} \quad (23)$$

where i is the current flowing through the coil.

- **A** This integral can be used to evaluate mutual inductances between coils. Similar to the formula for self inductance, mutual inductance is:

$$L_{mutual} = \frac{\int A_1 \cdot J_2 dV_2}{i_1 i_2} \quad (24)$$

where A_1 is the component of A produced by the first coil, J_2 is the current in the second coil, and i_1 and i_2 are the current in the first and second coils, respectively. dV_2 is meant to denote that the integral is taken over the volume of the second coil. We can rearrange (24) into a somewhat simpler form by noting that $n_2 * i_2 = J_2 * a_2$. That is, the total amps times turns for the second coil equals the current density in the second coil times the second coil’s cross-section area. Substituting for J_2 in (24) yields:

$$L_{mutual} = \frac{n_2}{i_1 a_2} \left(\int_{J_{2+}} A_1 dV_2 - \int_{J_{2-}} A_1 dV_2 \right) \quad (25)$$

where the first bracketed term in (25) is the contribution from the turns of coil 2 that are pointed out of the page and the second term is the contribution from the turns of coil 2 that are pointed into the page. To evaluate mutual inductance with FEMM, one substitutes values into (25). First, run the model with only “coil 1” turned on. Then, integrate A over the volume in which the second coil lies (although the second coil is not turned on). For planar problems, you will typically have to make two separate

integrations—one over the region where the turns in “coil 2” are pointed out of the page (*i.e.* that part of the coil in which a positive current results in current flowing in the out-of-the-page direction), and one over the region in which the turns in “coil 2” are pointed into the page. Add these two results together to get the total $A_1 dV_2$ integral. Lastly, multiply the integral result times $n_2/(i_1 a_2)$ to get mutual inductance.

- **Magnetic field energy** This selection calculates the energy stored in the magnetic field in the specified region. This integral can be used as an alternate method of getting inductance for problems that are linear (at least not heavily saturated). Denoting E as the energy stored in the magnetic field, inductance can be obtained by solving the formula:

$$E = \frac{Li^2}{2} \quad (26)$$

In the case of nonlinear materials, the energy is computed via:

$$W = \int \left(\int_0^B H(B') dB' \right) dV \quad (27)$$

to take proper account of the energy under nonlinear conditions

- **Magnetic field coenergy** For linear problems, coenergy is numerically the same as energy. For nonlinear problems, coenergy is defined as:

$$W_c = \int \left(\int_0^H B(H') dH' \right) dV \quad (28)$$

Coenergy can be used in an alternative method of force and torque computation. To compute force via coenergy, currents are held constant, and the position of the object upon which the force is desired is perturbed slightly. The force can then be estimated by:

$$F = \frac{W_c(p + \delta) - W_c(p)}{\delta} \quad (29)$$

where p denotes the initial position, $p + \delta$ denotes the perturbed position, and δ is the magnitude of the perturbation. The component of force determined in this way acts along the direction of the perturbation—one has to perform two such operations to get both horizontal and vertical components of the force.

- **Hyst. and/or Laminated eddy current losses.** This selection is typically used to obtain the core losses produced in laminated iron sections in harmonic problems.
- **Resistive losses** This selection integrates the $i^2 R$ losses due to currents flowing in the “ z ” direction (or θ direction, if you are evaluating an axisymmetric problem).
- **Block cross-section area**
- **Total losses** This selection totals the losses from all possible loss mechanisms that might apply over the given block. This is especially useful for finding losses in a region that might enclose several different types of materials with different loss mechanisms.

- **Lorentz force (JxB)** Lorentz force is the force produced by a magnetic field acting upon a current:

$$F_{Lorentz} = \int J \times B dV \quad (30)$$

Many devices (*e.g.* voice coil actuators) produce forces in a fashion that can be evaluated using this integral.

- **Lorentz torque (rxJxB)** This selection computes the torque about (0,0) resulting from Lorentz forces.
- **Integral of B over block** This integral can be useful in computing Lorentz forces. Since Lorentz force is $J \times B$, the force that would be produced if a coil were placed in a certain part of the solution domain can be inferred by integrating B , and then scaling times an arbitrarily chosen current density to get force.
- **Total current** This integral returns the total specified currents in the given block.
- **Block Volume** For axisymmetric problems, this selection returns the volume swept out by the selected block.
- **Force via Weighted Stress Tensor** New in version 3.3, the Weighted Stress Tensor block integrals automatically compute a weighting function over the finite element mesh that allows all possible air elements to contribute to the stress tensor integration. This approach is essentially identical to the weighted stress tensor approach described in [5] and/or [6].

To compute the force on a region or set of regions, the user selects the blocks upon which force result is desired and selects the **Force via Weighted Stress Tensor** integral. The program then computes the weighting function by solving an additional Laplace equation over the air surrounding the blocks upon which the force is to be computed. It may take a few seconds to compute the weighting function—progress is indicated by a progress bar that is displayed while the weighting function is being computed. The stress tensor is then evaluated as a volume integration, and the results are displayed. The results are typically more accurate than the Maxwell Stress Tensor line integral, since in some sense, all possible contours have been averaged to yield the Weighted Stress Tensor force result.

If the user is interested in the contours along which the integral was performed, the "stress tensor mask" box can be checked in the contour plot dialog. A set of orange (by default) lines will be displayed that.

- **Torque via Weighted Stress Tensor** This integral is torque version of the **Force via Weighted Stress Tensor** integral. Instead of force, torque about (0,0) is computed using the same weighting function approach.

4.10 Force/Torque Calculation

Ultimately, the estimation of magnetically produced forces and torques is often the goal of a finite element analysis. This section discusses some of the different methods of deducing forces and torques using FEMM.

4.10.1 Lorentz Force/Torque

If one is attempting to compute the force on a collection of currents in a region containing *only* materials with a unit relative permeability, the volume integral of Lorentz torque is always the method to employ. Lorentz force results tend to be very accurate. However, again, they are only applicable for the forces on conductors of with unit permeability (*e.g.* coils in a voice coil actuator).

4.10.2 Weighted Stress Tensor Volume Integral

New in version 3.3, this volume integral greatly simplifies the computation of forces and torques. Merely select the blocks upon which force or torque are to be computed and evaluate the integral. No particular “art” is required in getting good force or torque results (as opposed to the Stress tensor line integral), although results tend to be more accurate with finer meshing around the region upon which the force or torque is to be computed.

One limitation of the Weighted Stress Tensor integral is that the regions upon which the force is being computed must be entirely surrounded by air and/or abutting a boundary. In cases in which the desired region abuts a non-air region, force results may be deduced from differentiation of coenergy—see (29).

4.10.3 Maxwell Stress Tensor Line Integral

The indiscriminate use Maxwell’s Stress Tensor can result in bad predictions forces and torques. The goal of this section is to explain how to set up problems and properly choose integration paths so that good estimations of force and torque might be obtained via stress tensor methods. Generally, you should not use the Stress Tensor line integral to compute forces and torques if it can be avoided (*i.e.* use the volume integral version instead).

Maxwell’s stress tensor prescribes a force per unit area produced by the magnetic field on a surface. The differential force produced is:

$$dF = \frac{1}{2} (H(B \cdot n) + B(H \cdot n) - (H \cdot B)n) \quad (31)$$

where n denotes the direction normal to the surface at the point of interest. The net force on an object is obtained by creating a surface totally enclosing the object of interest and integrating the magnetic stress over that surface.

While an integration of (31) theoretically gives the magnetic force on an object, numerical problems arise when trying to evaluate this integral on a finite element mesh made of first-order triangles. Though the solution for vector potential A is relatively accurate, the distributions of B and H are an order less accurate, since these quantities are obtained by differentiating the trial functions for A . That is, A is described by a linear function over each element, but B and H are piece-wise constant over each element. Errors in B and H can be

particularly large in elements in which the exact solution for B and H changes rapidly—these areas are just not well approximated by a piece-wise approximation. Specifically, large errors can arise in the tangential components of B and H in elements adjacent to boundaries between materials of different permeabilities. The worst errors arise on this sort of interface at corners, where the exact solution for B is nearly a singularity.

The upshot is that if stress tensor is evaluated on the interface between two different materials, the results will be particularly erroneous. However, the stress tensor has the property that, for an exact solution, the same result is obtained regardless of the path of integration, as long as that path encircles the body of interest and passes only through air (or at least, every point in the contour is in a region with a constant permeability). This implies that the stress tensor can be evaluated over a contour a few elements away from the surface of an object—where the solution for B and H is much more accurate. Much more accurate force results will be obtained by integrating along the contour a few elements removed from any boundary or interface. The above discussion is the rationale for the first guideline for obtaining forces via stress tensor:

Never integrate stress stress tensor along an interface between materials. Always define the integration contour as a closed path around the object of interest with the contour displaced several elements (at least two elements) away from any interfaces or boundaries.

As an example of a properly defined contour, consider Figure 25. This figure represents a horseshoe magnet acting on a block of iron. The objective is to obtain the magnetic forces acting upon the iron block. The red line in the figure represents the contour defined for the integration. The contour was defined running clockwise around the block, so that the normal to the contour points outward. Always define your contour in a clockwise direction to get the correct sign. Note that the contour is well removed from the surface of the block, and the contour only passes through air. To aid in the definition of a closed contour, grid and the “snap to grid” were turned on, and the corners of the contour are grid points that were specified by right mouse button clicks.

The second rule of getting good force results is:

Always use as fine a mesh as possible in problems where force results are desired.

Even though an integration path has been chosen properly (away from boundaries and interfaces), some significant error can still arise if a coarse mesh is used. Note that (31) is composed of B^2 terms – this means that stress tensor is one order worse in accuracy than B . The only way to get that accuracy back is to use a fine mesh density. A good way to proceed in finding a mesh that is “dense enough” is to solve the problem on progressively finer meshes, evaluating the force on each mesh. By comparing the results from different mesh densities, you can get an idea of the level of accuracy (by looking at what digits in the answer that change between various mesh densities). You then pick the smallest mesh density that gives convergence to the desired digit of accuracy.

For torque computations, all the same rules apply as for force computations (*i.e.* define integration contours away from boundaries and interfaces, and use a dense mesh). Several beta testers have been using FEMM to obtain torques produced by motors and generators.

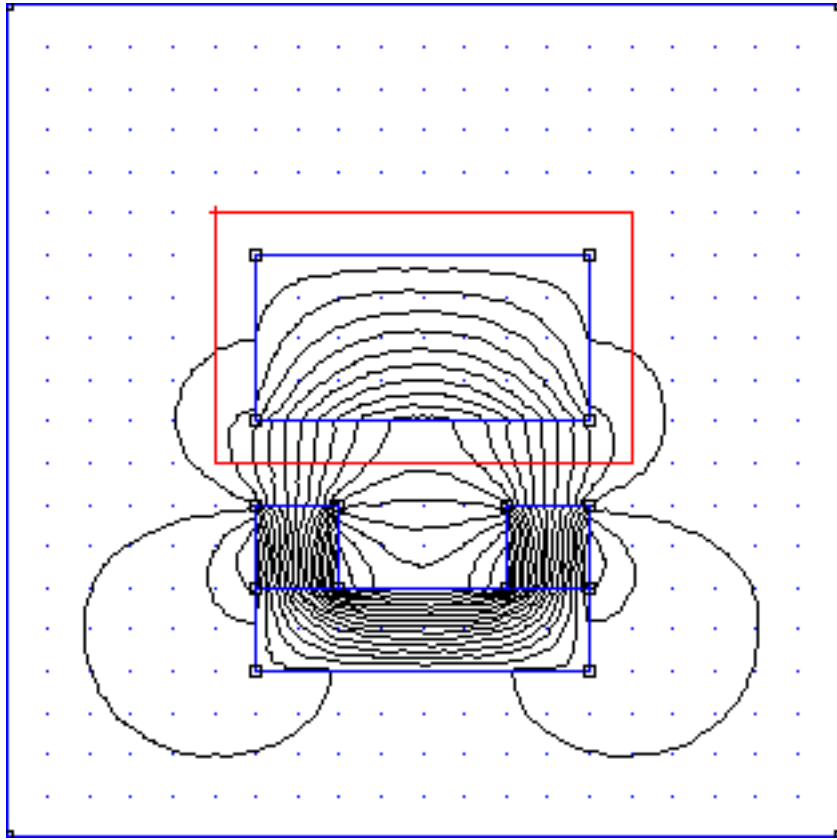


Figure 25: Properly defined contour for integration of Maxwell's Stress Tensor

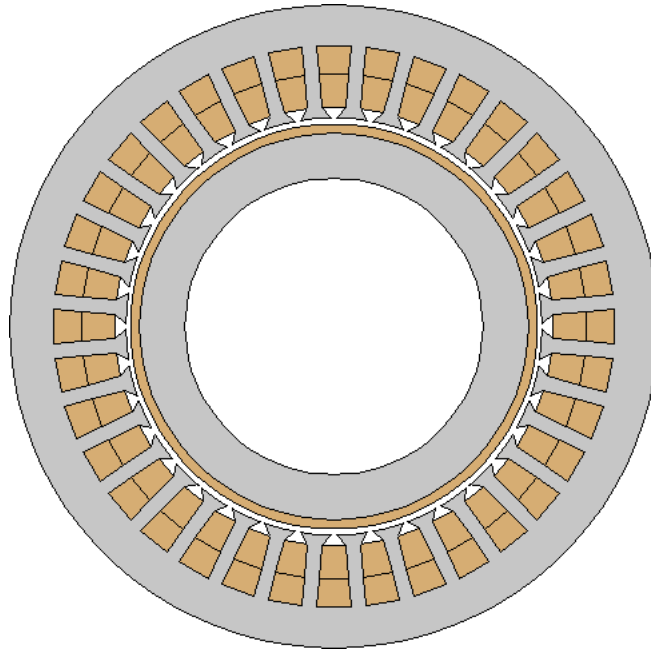


Figure 26: Example three-phase, six pole induction motor

In these machines, there are some steps that one can take in the definition of the model's geometry to make torque computation easier. Consider the motor pictured in Figure 26. This picture represents a three-phase induction motor wound so as to have six poles. The rotor is a conductive sheet attached to a laminated iron journal. To obtain the starting torque of the motor, one can impose three-phase currents in the winding and run a harmonic analysis at 60 Hz. The torque is then obtained by integrating the torque derived from Maxwell's stress tensor along a line running through the center of the air gap between the rotor and the stator.

A close-up of the input geometry near some of the tooth faces is shown in Figure 27. To aid in the evaluation of the torque, additional arc segments have been defined that run through the center of the air gap. This contour can be selected by clicking on the endpoints of the arcs with the left mouse button. This is much easier than trying to define a contour by explicitly specifying a lot of points in the air gap via right mouse button clicks. Note that the mesh density has been chosen such that there are four rows of elements in the thickness of the air gap, so that the integration contour (through the center of the air gap) is no closer than two rows of elements to either side of the gap. The mesh shown in Figure 27 represents the coarsest possible mesh that obeys the guidelines for obtaining good force/torque results, requiring about 62000 elements to mesh the entire motor. The solution, along with the integration contour shown in red, is pictured in Figure 28.

4.11 Exporting of Graphics

Ultimately, you will probably want to export graphics from FEMM for inclusion in reports and so on. It is possible to get what you are seeing on the screen onto disk in several different graphics formats.

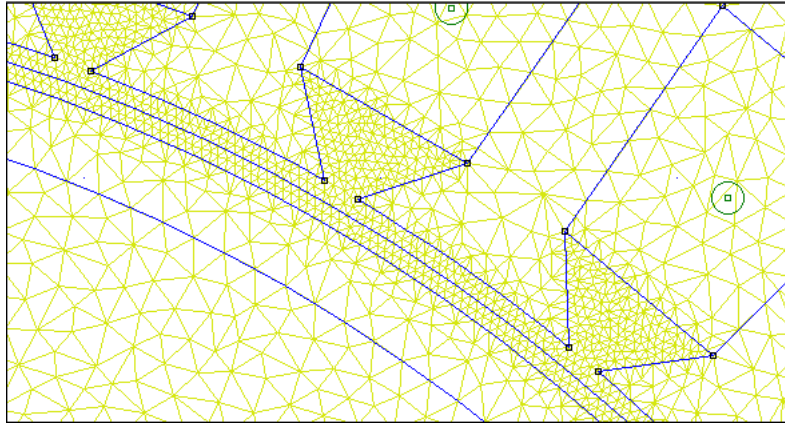


Figure 27: Input geometry in the region of the air gap.

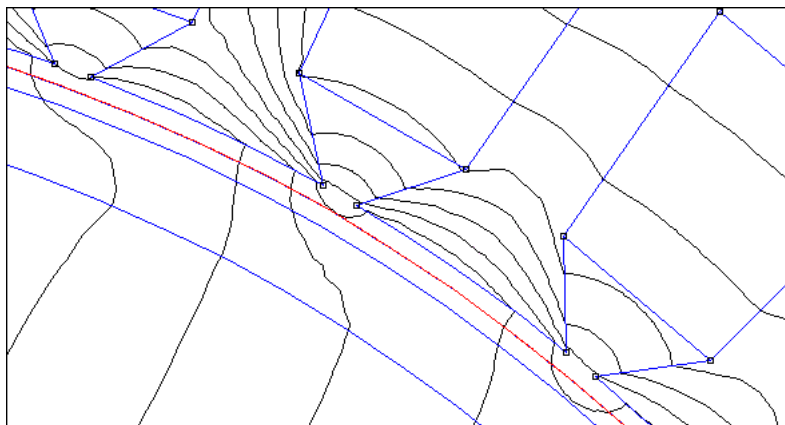


Figure 28: Solution with integration contour defined in the region of the air gap.

Probably the easiest way to get graphics out of femmview is to use the **Copy as Bitmap** or **Copy as Metafile** selections off of the main menu's **Edit** list. These command takes whatever is currently in the femmview window and copies it to the clipboard as a Device Independent Bitmap (.bmp format) and Extended Metafile (.emf format), respectively. The clipboard data can then be pasted directly into most applications (*e.g.* Word, MS Paint, etc).

L^AT_EXafficionados typically find PostScript to be the most useful type of graphics output. FEMM does not support postscript output directly, but it is still relatively easy to create postscript figures with FEMM. To obtain a postscript version of the current view, you first must set up a postscript printer driver that outputs to **File:**. This is done via the following steps:

1. Choose **Settings/Printers** off of the Windows Start menu. A window containing the list of currently defined printers will appear.
2. Double click on the **Add Printer** icon in this list. The **Add Printer Wizard** will appear on the screen.
3. Choose **Local Printer**; hit **Next**;
4. A list of printers will appear. Choose a postscript printer off of this list. The Apple Laserwriter II NT is a good choice.
5. Select **FILE:** as the port which will be used with this printer.
6. Accept the defaults for all remaining questions.

Now, when you want a postscript picture of the currently displayed screen, just choose **File/Print** off of femmview's main menu. As the printer, choose the postscript printer that you have previously defined. When you print to this printer, you will be prompted for a file name, and graphic will be written as a postscript figure to the specified file name.

4.12 Circuit Results

If "circuit" properties are used to specify the excitation, a useful biproduct is ready access to the impedance of the circuit. To view the circuit results, select **View—Circuit Props** off of the femmview main menu. A dialog, as pictured in Figure 29 will appear. There is a drop list on the dialog, from which the user selects the circuit for which results are desired. When a circuit is selected, the voltage drop, total current, and impedance associated with that circuit are displayed. For circuits that are imposed on non-conductive blocks or circuits with a zero total current, only a subset of this information is displayed.

4.13 Miscellaneous Useful View Commands

There are some additional entries on the femmview **View** menu that might be useful to you from time to time. These are:

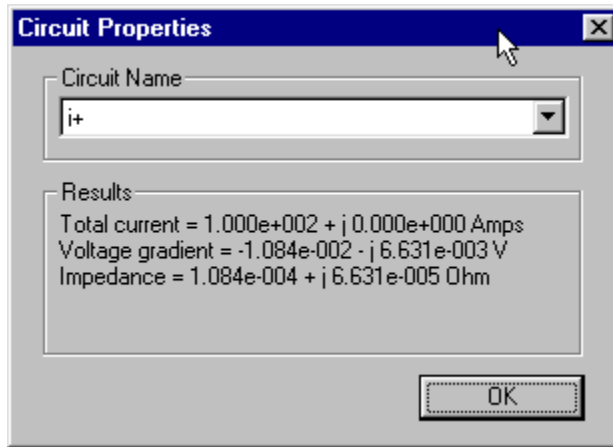


Figure 29: Circuit results dialog.

- **Smoothing** By default, a smoothing algorithm is applied to the flux density solution. Because first-order triangles are used as trial functions for vector potential, the resulting flux density and field intensity distributions are piece-wise constant in each element. The smoothing algorithm uses a nearest neighbor interpolation to obtain linear B and H distributions over each element. The smoothed solution generally looks better on the screen, and somewhat increases the accuracy of B and H near the vertices of each element. However, if you want to toggle smoothing, this can be done by selecting the **Smoothing** option.
- **Show Points** Especially when making graphics for reports, presentations, etc, it may be desirable to hide the small boxes on the screen that denote input node points. The **Show Points** option allows the user to toggle whether or not the input points are shown.
- **ToolBar** Use this toggle to hide and show the floating toolbar.
- **Point Props** Use this toggle to hide and show the floating dialog box used to display point property information.

5 Lua Scripting Documentation

5.1 What Lua Scripting?

The Lua extension language has been used to add scripting/batch processing facilities. The preprocessor and postprocessor can either run Lua scripts through a selection on the Files menu, or Lua commands can be entered in directly to the Lua Console Window in either program.

Lua is a complete, open-source scripting language. Source code for Lua, in addition to detailed documentation about programming in Lua, can be obtained from the Lua homepage at <http://www.lua.org>. Because the scripting files are text, they can be edited with any text editor (*e.g.* notepad).

In addition to the standard Lua command set, a number of FEMM-specific functions have been added for manipulating files in both the pre- and post-processor. These commands are described in the following sections.

Lua scripts are invoked by selecting the **Open Lua Script** selection off of the File menu of either the pre- or post-processor. A file selection dialog then appears, and the selected Lua script file is executed.

5.2 Preprocessor Lua Command Set

A number of different commands are available in the preprocessor. Two naming conventions can be used: one which separates words in the command names by underscores, and one that eliminates the underscores. A list of alternate, equivalent preprocessor scripting function names is shown in Table 3.

5.2.1 Object Add/Remove Commands

- `addnode(x,y)` Add a new node at x,y
- `addsegment(x1,y1,x2,y2)` Add a new line segment from node closest to (x1,y1) to node closest to (x2,y2)
- `addblocklabel(x,y)` Add a new block label at (x,y)
- `addarc(x1,y1,x2,y2,angle,maxseg)` Add a new arc segment from the nearest node to (x1,y1) to the nearest node to (x2,y2) with angle 'angle' divided into 'maxseg' segments.
- `deleteselected` Delete all selected objects.
- `deleteselectednodes` Delete selected nodes.
- `deleteselectedlabels` Delete selected block labels.
- `deleteselectedsegments` Delete selected segments.
- `deleteselectedarcsegments` Delete selected arcs.

open_femm_file	openfemmfile
save_femm_file	savefemmfile
create_mesh	createmesh
show_mesh	showmesh
purge_mesh	purgemesh
prob_def	probdef
analyse	analyze
run_post	runpost
add_node	addnode
add_block_label	addblocklabel
add_segment	addsegment
add_arc	addarc
select_node	selectnode
select_label	selectlabel
select_segment	selectsegment
select_arcsegment	selectarcsegment
clear_selected	clearselected
set_node_prop	setnodeprop
set_block_prop	setblockprop
set_segment_prop	setsegmentprop
set_arcsegment_prop	setarcsegmentprop
delete_selected	deleteselected
delete_selected_nodes	deleteselectednodes
delete_selected_labels	deleteselectedlabels
delete_selected_segments	deleteselectedsegments
delete_selected_arcsegments	deleteselectedarcsegments
zoom_natural	zoomnatural

zoom_out	zoomout
zoom_in	zoomin
add_material	addmaterial
add_point_prop	addpointprop
add_circ_prop	addcircprop
add_bound_prop	addboundprop
modify_material	modifymaterial
modify_bound_prop	modifyboundprop
modify_point_prop	modifypointprop
modify_circ_prop	modifycircprop
delete_material	deletematerial
delete_bound_prop	deleteboundprop
delete_circuit	deletecircuit
delete_point_prop	deletpointprop
move_rotate	moverotate
move_translate	movetranslate
copy_rotate	copyrotate
copy_translate	copytranslate
set_edit_mode	seteditmode
select_group	selectgroup
new_document	newdocument
save_bitmap	savebitmap
save_metafile	savemetafile
exit_pre	exitpre
add_bh_point	addbhpoint
clear_bh_points	clearbhpoints
refresh_view	refreshview
message_box	messagebox
grid_snap	gridsnap
show_grid	showgrid
hide_grid	hidegrid
set_grid	setgrid

Table 3: Alternate preprocessor scripting function names

5.2.2 Geometry Selection Commands

- `clearselected()` Clear all selected nodes, blocks, segments and arc segments.
- `selectsegment(x,y)` Select the line segment closest to (x,y)
- `selectnode(x,y)` Select the node closest to (x,y)
- `selectlabel(x,y)` Select the label closest to (x,y)
- `selectarcsegment(x,y)` Select the arc segment closest to (x,y)
- `selectgroup(n)` Select the n^{th} group of nodes, segments, arc segments and blocklabels. This function will clear all previously selected elements and leave the editmode in 4 (group)

5.2.3 Object Labeling Commands

- `setnodeprop("propname", groupno)` Set the selected nodes to have the nodal property "propname" and group number groupno. Note the property must exist before calling this function.
- `setblockprop("blockname", automesh, meshsize, "incircuit", magdirection, group)` Set the selected block labels to have the properties:
 - Block property "blockname".
 - `automesh`: 0 = mesher defers to mesh size constraint defined in `meshsize`, 1 = mesher automatically chooses the mesh density.
 - `meshsize`: size constraint on the mesh in the block marked by this label.
 - Block is a member of the circuit named "incircuit" (Note this circuit name must already exist)
 - The magnetization is directed along an angle in measured in degrees denoted by `magdirection`
 - A member of group number `group`
- `setsegmentprop("propname", elementsize, automesh, hide, group)` Set the selected segments to have:
 - Boundary property "propname"
 - Local element size along segment no greater than `elementsize`
 - `automesh`: 0 = mesher defers to the element constraint defined by `elementsize`, 1 = mesher automatically chooses mesh size along the selected segments
 - `hide`: 0 = not hidden in post-processor, 1 == hidden in post processor
 - A member of group number `group`

- `setarcsegmentprop(maxsegdeg, "propname", hide, group)` Set the selected arc segments to:
 - Meshed with elements that span at most `maxsegdeg` degrees per element
 - Boundary property `"propname"`
 - `hide`: 0 = not hidden in post-processor, 1 == hidden in post processor
 - A member of group number `group`

5.2.4 Problem Commands

- `probdef(frequency,units,type,precision,(depth))` changes the problem definition. Set `frequency` to the desired frequency in Hertz. The `units` parameter specifies the units used for measuring length in the problem domain. Valid `"units"` entries are `"inches"`, `"millimeters"`, `"centimeters"`, `"mils"`, `"meters"`, and `"micrometers"`. Set `problemtyp` to `"planar"` for a 2-D planar problem, or to `"axi"` for an axisymmetric problem. The `precision` parameter dictates the precision required by the solver. For example, entering `1E-8` requires the RMS of the residual to be less than 10^{-8} . A fifth parameter, representing the depth of the problem in the into-the-page direction for 2-D planar problems, can also be specified.

- `analyse(flag)` runs `fkern` to solve the problem. The `flag` parameter controls whether the `fkern` window is visible or minimized. For a visible window, either specify no value for `flag` or specify 0. For a minimized window, `flag` should be set to 1.

- `runpost("filename")` starts the post processor and instructs `femmview` to execute the lua file `"filename"`. The current solution will also be passed to `femmview` and loaded.

The file name must be in a C `printf` format, so for a backslash, use two backslashes (*i.e.* `\\`). If the file name contains a space (*e.g.* file names like `c:\program files\stuff`) you must enclose the file name in (extra) quotes by using a `\"` sequence. For example:
`runpost("\"c:\\program files\\femm30\\bin\\testpost.lua\"")`

Several additional parameters can be included in the `runpost` call. A limited number of variables can be passed to the postprocessor by adding a extra parameters of the form `-lua-var=variable=value`, *e.g.*:

```
run_post("c:\\my-lua-script.lua", "-lua-var=filename=myfilename")
```

All variables are passed as strings so lua internal conversion routines must be employed to obtain numbers from the strings.

The postprocessor window can also be minimized by including the parameter `"-windowhide"`, *e.g.*:

```
run_post("c:\\myluascrip.lua", "-windowhide")
```

5.2.5 File Commands

- `savefemmfile("filename")` saves the file with name `"filename"`. Note if you use a path you must use two backslashes *e.g.* `"c:\\temp\\myfemmfile.fem"`

- `openfemmfile("filename")` opens the file with name "filename". Note that if you use a path you must use two backslashes (*e.g.* "c:\\temp\\myfemmfile.fem"). If the file name contains a space (*e.g.* file names like c:\program files\stuff) you must enclose the file name in (extra) quotes by using a \" sequence. For example:
`runpost("\\c:\\program files\\femm30\\bin\\testpost.lua\\")`
- `newdocument()` clears everything to a new blank document.

5.2.6 Mesh Commands

- `createmesh()` runs triangle to create a mesh. Note that this is not a necessary precursor of performing an analysis, as `analyse()` will make sure the mesh is up to date before running an analysis. The number of elements in the mesh is pushed back onto the lua stack.
- `showmesh()` shows the mesh.
- `purgemesh()` clears the mesh out of both the screen and memory.

5.2.7 Editing Commands

- `copyrotate(bx, by, angle, copies, (editaction))`
 - `bx, by` – base point for rotation
 - `angle` – angle by which the selected objects are incrementally shifted to make each copy. `angle` is measured in degrees.
 - `copies` – number of copies to be produced from the selected objects.
- `copytranslate(dx, dy, copies, (editaction))`
 - `dx,dy` – distance by which the selected objects are incrementally shifted.
 - `copies` – number of copies to be produced from the selected objects.
 - `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4-group
- `moverotate(bx,by,shiftangle (editaction))`
 - `bx, by` – base point for rotation
 - `shiftangle` – angle in degrees by which the selected objects are rotated.
 - `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4-group
- `movetranslate(dx,dy,(editaction))`
 - `dx,dy` – distance by which the selected objects are shifted.
 - `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4-group

- `scale(bx,by,scalefactor,(editaction))`
 - `bx`, `by` – base point for scaling
 - `scalefactor` – a multiplier that determines how much the selected objects are scaled
 - `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4–group
- `mirror(x1,y1,x2,y2,(editaction))` mirror the selected objects about a line passing through the points `(x1,y1)` and `(x2,y2)`. Valid `editaction` entries are 0 for nodes, 1 for lines (segments), 2 for block labels, 3 for arc segments, and 4 for groups.
- `seteditmode(editmode)` Sets the current editmode to:
 - "nodes" - nodes
 - "segments" - line segments
 - "arcsegments" - arc segments
 - "blocks" - block labels
 - "group" - selected group

This command will affect all subsequent uses of the other editing commands, if they are used WITHOUT the `editaction` parameter.

5.2.8 Zoom Commands

- `zoomnatural()` zooms to a “natural” view with sensible extents.
- `zoomout()` zooms out by a factor of 50%.
- `zoomin()` zoom in by a factor of 200%.
- `zoom(x1,y1,x2,y2)` Set the display area to be from the bottom left corner specified by `(x1,y1)` to the top right corner specified by `(x2,y2)`.

5.2.9 View Commands

- `showgrid()` Show the grid points.
- `hidegrid()` Hide the grid points points.
- `grid_snap("flag")` Setting `flag` to "on" turns on snap to grid, setting `flag` to "off" turns off snap to grid.
- `setgrid(density,"type")` Change the grid spacing. The `density` parameter specifies the space between grid points, and the `type` parameter is set to "cart" for cartesian coordinates or "polar" for polar coordinates.
- `refreshview()` Redraws the current view.

5.2.10 Object Properties

- `addmaterial("materialname", mu_x, mu_y, H_c, Jr, Ji, Cduct, Lam_d, Theta_h, lam_fill, LamType)` adds a new material with called "materialname" with the material properties:
 - `mu_x` Relative permeability in the x- or r-direction.
 - `mu_y` Relative permeability in the y- or z-direction.
 - `H_c` Permanent magnet coercivity in Amps/Meter.
 - `Jr` Real (in phase) portion of the applied source current density in Amps/mm².
 - `Ji` Imaginary (out of phase) portion of the applied source current density in Amps/mm².
 - `Cduct` Electrical conductivity of the material in MS/m.
 - `Lam_d` Lamination thickness in millimeters.
 - `Theta_h` Hysteresis lag angle in degrees.
 - `Lam_fill` Fraction of the volume occupied per lamination that is actually filled with iron (Note that this parameter defaults to 1 the `femme` preprocessor dialog box because, by default, iron completely fills the volume)
 - `Lamtype` Set to
 - * 0 – Not laminated or laminated in plane
 - * 1 – laminated x or r
 - * 2 – laminated y or z
- `addbhpoint("blockname",b,h)` Adds a B-H data point the the material specified by "blockname". The point to be added has a flux density of `b` in units of Teslas and a field intensity of `h` in units of Amps/Meter.
- `clearbhpoints("blockname")` Clears all B-H data points associated with the material specified by "blockname".
- `addpointprop("pointpropname",a_re,a_im,j_re,j_im)` adds a new point property of name "pointpropname" with either a specified potential `a_im`, `a_re` in units Webers/Meter or a point current `j_im`, `j_re` in units of Amps. Set the unused parameter pairs to 0.
- `addboundprop("boundpropname", A0, A1, A2, Phi, Mu, Sig, c0, c1, BdryFormat)` adds a new boundary property with name "boundpropname"
 - For a "Prescribed A" type boundary condition, set the `A0`, `A1`, `A2` and `Phi` parameters as required. Set all other parameters to zero.
 - For a "Small Skin Depth" type boundary condtion, set the `Mu` to the desired relative permeability and `Sig` to the desired conductivity in MS/m. Set `BdryFormat` to 1 and all other parameters to zero.

- To obtain a “Mixed” type boundary condition, set `C1` and `C0` as required and `BdryFormat` to 2. Set all other parameters to zero.
 - For a “Strategic dual image” boundary, set `BdryFormat` to 3 and set all other parameters to zero.
 - For a “Periodic” boundary condition, set `BdryFormat` to 4 and set all other parameters to zero.
 - For an “Anti-Periodic” boundary condition, set `BdryFormat` to 5 set all other parameters to zero.
- `addcircprop("circuitname", i_re, i_im, dvolt_re, dvolt_im, circuittype)` adds a new circuit property with name "circuitname" with either a prescribed voltage gradient or a prescribed total current. Set the unused property pair to zero. The `circuittype` parameter is 0 for prescribed current and 1 for prescribed voltage gradient.
 - `deletematerial("materialname")` deletes the material named "materialname".
 - `deleteboundprop("boundpropname")` deletes the boundary property named "boundpropname".
 - `deletecircuit("circuitname")` deletes the circuit named `circuitname`.
 - `deletepointprop("pointpropname")` deletes the point property named "pointpropname"
 - `modifymaterial("BlockName", propnum, value)` This function allows for modification of a material’s properties without redefining the entire material (*e.g.* so that current can be modified from run to run). The material to be modified is specified by "BlockName". The next parameter is the number of the property to be set. The last number is the value to be applied to the specified property. The various properties that can be modified are listed below:

propnum	Symbol	Description
0	BlockName	Name of the material
1	μ_x	x (or r) direction relative permeability
2	μ_y	y (or z) direction relative permeability
3	H_c	Coercivity, Amps/Meter
4	J_r	Real part of current density, MA/m ²
5	J_i	Imaginary part of current density, MA/m ²
6	σ	Electrical conductivity, MS/m
7	d_{lam}	Lamination thickness, mm
8	θ_h	Hysteresis lag angle, degrees
9	LamFill	Iron fill fraction
10	LamType	0 = None/In plane, 1 = parallel to x, 2=parallel to y

- `modifyboundprop("BdryName", propnum, value)` This function allows for modification of a boundary property. The BC to be modified is specified by "BdryName". The next parameter is the number of the property to be set. The last number is the value to

be applied to the specified property. The various properties that can be modified are listed below:

propnum	Symbol	Description
0	BdryName	Name of boundary property
1	A_0	Prescribed A parameter
2	A_1	Prescribed A parameter
3	A_2	Prescribed A parameter
4	ϕ	Prescribed A phase
5	μ	Small skin depth relative permeability
6	σ	Small skin depth conductivity, MS/m
7	c_0	Mixed BC parameter
8	c_1	Mixed BC parameter
9	BdryFormat	Type of boundary condition: 0 = Prescribed A 1 = Small skin depth 2 = Mixed 3 = Strategic Dual Image 4 = Periodic 5 = Antiperiodic

- `modifypointprop("PointName",propnum,value)` This function allows for modification of a point property. The point property to be modified is specified by "PointName". The next parameter is the number of the property to be set. The last number is the value to be applied to the specified property. The various properties that can be modified are listed below:

propnum	Symbol	Description
0	PointName	Name of the point property
1	A_{re}	Real part of nodal potential, Weber/Meter
2	A_{im}	Imaginary part of nodal potential Weber/Meter
3	J_{re}	Real part of nodal current, Amps
4	J_{im}	Imaginary part of nodal current, Amps

- `modifycircprop("CircName",propnum,value)` This function allows for modification of a circuit property. The circuit property to be modified is specified by "CircName". The next parameter is the number of the property to be set. The last number is the value to be applied to the specified property. The various properties that can be modified are listed below:

propnum	Symbol	Description
0	CircName	Name of the circuit property
1	i_{re}	Real part of total current
2	i_{im}	Imaginary part of total current
3	δv_{re}	Real part of voltage gradient
4	δv_{im}	Imaginary part of voltage gradient
5	CircType	0 = Prescribed current, 1 = Prescribed voltage gradient

5.2.11 Miscellaneous

- `savebitmap("filename")` saves a bitmapped screenshot of the current view to the file specified by "filename", subject to the `printf`-type formatting explained previously for the `savefemmfile` command.
- `savemetafile("filename")` saves a metafile screenshot of the current view to the file specified by "filename", subject to the `printf`-type formatting explained previously for the `savefemmfile` command.
- `messagebox("message")` displays the "message" string to the screen in a pop-up message box.
- `refreshview()` Redraws the current view.
- `exitpre()` exits the preprocessor after the lua script has finished executing.
- `print()` This is standard Lua "print" command directed to the output of the Lua console window. Any number of comma-separated items can be printed at once via the print command.
- `prompt("message")` This function allows a Lua script to prompt a user for input. When this command is used, a dialog box pops up with the "message" string on the title bar of the dialog box. The user can enter in a single line of input via the dialog box. `prompt` returns the user's input as a string. If a numerical value is desired, the syntax `tonumber(prompt("message"))` can be used.
- `shownames(flag)` This function allow the user to display or hide the block label names on screen. To hide the block label names, `flag` should be 0. To display the names, the parameter should be set to 1.

5.3 Post Processor Command Set

There are a number of Lua scripting commands designed to operate in the postprocessor. As with the preprocessor commands, these commands can be used with either the underscore naming or with the no-underscore naming convention. The equivalent function names in the two conventions are shown in Table 4

5.3.1 Data Extraction Commands

- `lineintegral(type)` Calculate the line integral for the defined contour

type	name	values 1	values 2	values 3	values 4
0	B.n	total B.n	avg B.n	-	-
1	H.t	total H.t	avg H.t	-	-
2	Contour length	surface area	-	-	-
3	Stress Tensor Force	DC r/x force	DC y/z force	2× r/x force	2× y/z force
4	Stress Tensor Torque	DC torque	2× torque	-	-
5	(B.n)^2	total (B.n)^2	avg (B.n)^2	-	-

get_point_values	getpointvalues	hide_density_plot	hidedensityplot
exit_post	exitpost	show_density_plot	showdensityplot
add_contour	addcontour	hide_contour_plot	hidecountourplot
bend_contour	bendcontour	show_contour_plot	showcountourplot
clear_contour	clearcontour	show_points	showpoints
line_integral	lineintergral	hide_points	hidepoints
select_block	selectblock	grid_snap	gridsnap
group_select_block	groupselectblock	set_grid	setgrid
clear_block	clearblock	get_problem_info	getprobleminfo
block_integral	blockintergral	save_bitmap	savebitmap
zoom_natural	zoomnatural	get_circuit_properties	getcircuitproperties
zoom_in	zoomin	save_metafile	savemetafile
zoom_out	zoomout	refresh_view	refreshview
show_grid	showgrid	select_point	selectpoint
hide_grid	hidegrid	show_point_props	showpointprops
show_mesh	showmesh	hide_point_props	hidepointprops
hide_mesh	hidemesh	message_box	messagebox
set_edit_mode	seteditmode	make_plot	makeplot

Table 4: Alternate postprocessor scripting function names

Returns typically four floating point values as results. The first two values are the total real and imaginary parts of the integral result, the second pair the average real and imaginary parts, *e.g.*:

```
linere, lineim, advre, advim = lineintegral(0)
```

The only exception is integral 3, which evaluates Maxwell's stress tensor. This integral can return up to eight results. For force and torque results, the $2\times$ results are only relevant for problems where $\omega \neq 0$.

- `blockintegral(type)` Calculate a block integral for the selected blocks

Type	Definition
0	$A \cdot J$
1	A
2	Magnetic field energy
3	Hysteresis and/or lamination losses
4	Resistive losses
5	Block cross-section area
6	Total losses
7	Total current
8	Integral of B_x (or B_r) over block
9	Integral of B_y (or B_z) over block
10	Block volume
11	x (or r) part of steady-state Lorentz force
12	y (or z) part of steady-state Lorentz force
13	x (or r) part of $2\times$ Lorentz force
14	y (or z) part of $2\times$ Lorentz force
15	Steady-state Lorentz torque
16	$2\times$ component of Lorentz torque
17	Magnetic field coenergy
18	x (or r) part of steady-state weighted stress tensor force
19	y (or z) part of steady-state weighted stress tensor force
20	x (or r) part of $2\times$ weighted stress tensor force
21	y (or z) part of $2\times$ weighted stress tensor force
22	Steady-state weighted stress tensor torque
23	$2\times$ component of weighted stress tensor torque

Returns 2 parameters for the real and imaginary components, *e.g.*:

```
re, im = blockintegral(10)
```

- `getpointvalues(X,Y)` Get the values associated with the point at x,y RETURN values in order

Symbol	Definition
Are	real part of A or flux ϕ
Aim	imaginary part of A or flux ϕ
B1re	real part of B_x if planar, B_r if axisymmetric
B1im	imag part of B_x if planar, B_r if axisymmetric
B2re	real part of B_y if planar, B_z if axisymmetric
B2im	imag part of B_y if planar, B_z if axisymmetric
Sig	conductivity σ
E	stored energy density
H1re	real part of H_x if planar, H_r if axisymmetric
H1im	imag part of H_x if planar, H_r if axisymmetric
H2re	real part of H_y if planar, H_z if axisymmetric
H2im	imag part of H_y if planar, H_z if axisymmetric
Jere	real part of eddy current density
Jeim	imag part of eddy current density
Jsre	real part of source current density
Jsim	imag part of source current density
Mu1re	real part of μ_x if planar, μ_r if axisymmetric
Mu1im	imag part of μ_x if planar, μ_r if axisymmetric
Mu2re	real part of μ_y if planar, μ_z if axisymmetric
Mu2im	imag part of μ_y if planar, μ_z if axisymmetric
Pe	Power density dissipated through ohmic losses
Ph	Power density dissipated by hysteresis

Example: To catch all values at (0.01,0) use

```
Are, Aim, B1re, B1im, B2re, B2im, Sig, E, H1re, H1im, H2re, H2im, Jere, Jeim, Jsre, Jsim, Mu1re, Mu1im, Mu2re, Mu2im, Pe, Ph = getpointvalues(0.01,0)
```

For magnetostatic problems, all imaginary quantities are zero.

- `makeplot(PlotType, NumPoints, Filename, FileFormat)` Allows Lua access to the X-Y plot routines. If only `PlotType` or only `PlotType` and `NumPoints` are specified, the command is interpreted as a request to plot the requested plot type to the screen. If, in addition, the `Filename` parameter is specified, the plot is instead written to disk to the specified file name as an extended metafile. If the `FileFormat` parameter is also, the command is instead interpreted as a command to write the data to disk to the specified file name, rather than display it to make a graphical plot. Valid entries for `PlotType` are:

PlotType	Definition
0	Potential
1	$ B $
2	$B \cdot n$
3	$B \cdot t$
4	$ H $
5	$H \cdot n$
6	$H \cdot t$
7	J_{eddy}
8	$J_{source} + J_{eddy}$

Valid file formats are

FileFormat	Definition
0	Multi-column text with legend
1	Multi-column text with no legend
2	Mathematica-style formatting

For example, if one wanted to plot $B \cdot n$ to the screen with 200 points evaluated to make the graph, the command would be:

```
makeplot(2,200)
```

If this plot were to be written to disk as a metafile, the command would be:

```
makeplot(2,200,"c:\\temp\\myfile.emf")
```

To write data instead of a plot to disk, the command would be of the form:

```
makeplot(2,200,"c:\\temp\\myfile.txt",0)
```

- `getprobleminfo()` Returns info on problem description. Returns two values:

Return value	Definition
1	problem type
2	frequency in Hz

- `getcircuitproperties("circuit")` Used primarily to obtain impedance information associated with circuit properties. Properties are returned for the circuit property named "circuit". Six values are returned by the function. In order, these parameters are:

- `totalcurrent_re` Real component of total current carried by the circuit.
- `totalcurrent_im` Imaginary component of total current carried by the circuit.
- `vgrad_re` Real component of the voltage gradient in the circuit. Meaningful only when the circuit property is applied to blocks with nonzero conductivity.
- `vgrad_im` Imaginary part of the voltage gradient in the circuit. Meaningful only when the circuit property is applied to blocks with nonzero conductivity.
- `impd_re` Real part of the impedance driven by the circuit.

- `impd_im` Imaginary part of the impedance driven by the circuit.
- `currentdensity_re` Current density in the circuit. Only relevant is the circuit property is applied to one or more blocks with zero conductivity (otherwise, the current density is not necessarily uniform)
- `currentdensity_im` Imaginary part of current density in the circuit

5.3.2 Selection Commands

- `seteditmode(mode)` Sets the mode of the postprocessor to point, contour, or area mode. Valid entries for `mode` are "point", "contour", and "area".
- `selectblock(x,y)` Select the block that contains point (x,y).
- `groupselectblock(n)` Selects all of the blocks that are labeled by block labels that are members of group n. If no number is specified (*i.e.* `groupselectblock()`), all blocks are selected.
- `addcontour(x,y)` Adds a contour point at (x,y). If this is the first point then it starts a contour, if there are existing points the contour runs from the previous point to this point. The `addcontour` command has the same functionality as a right-button-click contour point addition when the program is running in interactive mode.
- `bendcontour(angle,anglestep)` Replaces the straight line formed by the last two points in the contour by an arc that spans `angle` degrees. The arc is actually composed of many straight lines, each of which is constrained to span no more than `anglestep` degrees. The `angle` parameter can take on values from -180 to 180 degrees. The `anglestep` parameter must be greater than zero. If there are less than two points defined in the contour, this command is ignored.
- `selectpoint(x,y)` Adds a contour point at the closest input point to (x,y). If the selected point and a previous selected points lie at the ends of an arcsegment, a contour is added that traces along the arcsegment. The `selectpoint` command has the same functionality as the left-button-click contour point selection when the program is running in interactive mode.
- `clearcontour()` Clear a previously defined contour
- `clearblock()` Clear block selection

5.3.3 Zoom Commands

- `zoomnatural()` Zoom to the natural boundaries of the geometry.
- `zoomin()` Zoom in one level.
- `zoomout()` Zoom out one level.
- `zoom(x1,y1,x2,y2)` Zoom to the window defined by lower left corner (x1,y1) and upper right corner (x2,y2).

5.3.4 View Commands

- `showmesh()` Show the mesh.
 - `hidemesh()` Hide the mesh.
 - `showpoints()` Show the node points from the input geometry.
 - `hidepoints()` Hide the node points from the input geometry.
 - `smooth("flag")` This function controls whether or not smoothing is applied to the B and H fields, which are naturally piece-wise constant over each element. Setting `flag` equal to "on" turns on smoothing, and setting `flag` to "off" turns off smoothing.
 - `showgrid()` Show the grid points.
 - `hidegrid()` Hide the grid points points.
 - `grid_snap("flag")` Setting `flag` to "on" turns on snap to grid, setting `flag` to "off" turns off snap to grid.
 - `setgrid(density,"type")` Change the grid spacing. The `density` parameter specifies the space between grid points, and the `type` parameter is set to "cart" for cartesian coordinates or "polar" for polar coordinates.
 - `hidedensityplot()` hides the flux density plot.
 - `showdensityplot(legend,gscale,upper_B,lower_B,type)` Shows the flux density plot with options:
 - `legend` Set to 0 to hide the plot legend or 1 to show the plot legend.
 - `gscale` Set to 0 for a colour density plot or 1 for a grey scale density plot.
 - `upper_B` Sets the upper display limit for the density plot.
 - `lower_B` Sets the lower display limit for the density plot.
 - `type` Type of density plot to display. Valid entries are "mag", "real", and "imag" for magnitude, real component, and imaginary component of B , respectively. Alternatively, current density can be displayed by specifying "jmag", "jreal", and "jimag" for magnitude, real component, and imaginary component of J , respectively.
- if `legend` is set to -1 all parameters are ignored and default values are used *e.g.*
`show_density_plot(-1)`
- `hidecontourplot()` Hides the contour plot.
 - `showcontourplot(numcontours,lower_A,upper_A,type)` shows the A contour plot with options:
 - `numcontours` Number of A equipotential lines to be plotted.

- upper_A Upper limit for A contours.
- lower_A Lower limit for A contours.

If numcontours is -1 all parameters are ignored and default values are used, *e.g.* `show_contour_plot(-1)`

- `showpointprops()` Displays the floating Point Properties display window.
- `hidepointprops()` Hides the floating Point Properties display window.

5.3.5 Miscellaneous

- `savebitmap("filename")` saves a bitmapped screen shot of the current view to the file specified by "filename". Note that if you use a path you must use two backslashes (*e.g.* "c:\\temp\\myfemmfile.fem"). If the file name contains a space (*e.g.* file names like c:\program files\stuff) you must enclose the file name in (extra) quotes by using a \" sequence. For example:
`save_bitmap("\\c:\\temp\\femm30\\bin\\screenshot.bmp\\")`
- `savemetafile("filename")` saves a metafile screenshot of the current view to the file specified by "filename", subject to the `printf`-type formatting explained previously for the `savebitmap` command.
- `messagebox("message")` displays the "message" string to the screen in a pop-up message box.
- `exitpost()` Quit the post-processor.
- `pause()` Waits for the ok button to be pressed, a debug helper.
- `print()` This is standard Lua "print" command directed to the output of the Lua console window. Any number of comma-separated items can be printed at once via the print command.
- `prompt("message")` This function allows a Lua script to prompt a user for input. When this command is used, a dialog box pops up with the "message" string on the title bar of the dialog box. The user can enter in a single line of input via the dialog box. `prompt` returns the user's input as a string. If a numerical value is desired, the syntax `tonumber(prompt("message"))` can be used.
- `shownames(flag)` This function allow the user to display or hide the block label names on screen. To hide the block label names, `flag` should be 0. To display the names, the parameter should be set to 1.

6 Numerical Methods

For those of you interested in what's going on behind the scenes in the `fkern` solver, this section is meant as a brief description of the methods and techniques used by FEMM. References are cited as applicable.

6.1 Finite Element Formulation

All elements were derived using variational formulations (based on minimizing energy, as opposed to Galerkin, least squares residual, and so on). Explanations of the variational approach for 2-D planar problems with first-order triangle elements are widely available in the literature ([7] in particular was referred to during the creation of FEMM).

The axisymmetric case for magnetics, however, is oddly less well addressed. Hoole [2] and Silvester [8] promote solving axisymmetric problems in terms of a modified vector potential. The advantage of the modified vector potential is that closed-form expressions for each term in the element matrices can be formed. An early version of FEMM used this technique, but it is observed to yield relatively larger errors near $r = 0$. With the modified potential formulation, it is also nontrivial to compute the average flux density associated with each element.

FEMM used to use an axisymmetric formulation developed directly from vector potential A interpolated linearly over each element. Although this formulation gives very good results close to $r = 0$, and often does well in general, it is not *always* well-behaved. However, the formulation suggested in [9] gives the same good performance close to $r = 0$, while succeeding in the cases where the direct A formulation breaks down.

6.2 Linear Solvers

For all problems, variations of the iterative Conjugate Gradient solver are used. This technique is appropriate for the sort of problem that FEMM solves, because the matrices are symmetric and very sparse. A row-based storage scheme is used in which only the nonzero elements of the diagonal and upper triangular part of the matrix are solved.

For magnetostatic problems, the preconditioned conjugate gradient (PCG) code is based on the discussion in [8]. Minor modifications are made to this algorithm to avoid computing certain quantities more than once per iteration. Although Silvester promotes the use of the Incomplete Cholesky preconditioner, it is not used in FEMM, because it nearly doubles the storage requirements—for each element of the matrix stored, a corresponding element of the preconditioner must also be stored. Instead, the Symmetric Successive Over-Relaxation (SSOR) preconditioner, as described in [10], is used. The advantage of this preconditioner is that it is built on the fly in a simple way using only the matrix elements that are already in storage. In general, the speed of PCG using SSOR is said to be comparable to the speed of PCG with Incomplete Cholesky.

For harmonic problems, the regular PCG algorithm cannot be used; the matrix that arises in the formulation of harmonic problems is Complex Symmetric (*i.e.* $A = A^T$), rather than Hermitian (*i.e.* $A = A^*$). Curiously, there is very little literature available on iterative solvers for complex symmetric problems, given the number of diverse applications in which

these problems arise. However, there is a very good paper on the solution of linear problems with complex symmetric matrices via various flavors of Conjugate Gradient by Freund [11]. The techniques discussed by Freund allow one to operate directly on the complex symmetric matrix and take advantage of the symmetric structure to minimize the number of computations that must be performed per iteration. Although Freund supports Quasi-Minimum Residual approach, FEMM uses the complex symmetric version of biconjugate gradient also described in [11]. After coding and comparing the the speed of both BCG and QMR, it was found that BCG is somewhat faster due to a relatively smaller number of computations that must be performed per iteration (even though QMR has better convergence properties than BCG).

However, using the algorithms as described by [11], solution times were unacceptably long. To decrease solution times, the complex symmetric BCG algorithm was modified to include the SSOR preconditioner (built in exactly the same way as for magnetostatic problems). Including the SSOR preconditioner in complex symmetric BCG problems usually yields an order of magnitude improvement in speed over no preconditioner.

In all problems, a node renumbering scheme is used. Although the conjugate gradient schemes work well without renumbering, the renumbering seems to roughly halve the solution time. There is an overall advantage to using the renumbering, because the of time required to perform the renumbering is small compared to the time required to run CG or BCG. Although there are many possible approaches to renumbering, FEMM uses the Cuthill-McKee method as described in [2]. Although there are newer schemes that yield a tighter profile, Cuthill-McKee does a relatively good job and requires very little to execute. The renumbering code is a hold-over from an early version of FEMM that employed a banded Gauss Elimination solver in which a good node numbering is essential to good performance. The renumbering speeds up CG and BCG by reducing the error between the SSOR approximation of A^{-1} and the exact A^{-1} . An interesting paper on the effect of the ordering of the unknowns on convergence in conjugate gradient methods is [12].

6.3 Field Smoothing

Since first-order triangles are used by FEMM, the resulting solution for B and H obtained by differentiating A is constant over each element. If the raw B and H are used by the postprocessor, density plots of B and 2-D plots of field quantities along user-defined contours look terrible. Also, the values of B and H aren't so accurate at points in an element away from the element's centroid.

The use of smoothing to recover the accuracy lost by differentiating A is known as *superconvergence*. There are quite a few researchers actively pursuing this area. A good bibliography of current research is on the web at <http://www.isc.tamu.edu/srihari/refer.htm>.

Of the greatest interest to FEMM are so-called "patch recovery" techniques. The basic idea is the the solutions for B are most accurate at the centroid of the triangular element (known as its *Gauss Point*). One desires a continuous profile of B that can be interpolated from nodal values, in the same way that vector potential A can be represented. The problem is, the "raw" solution of B is multivalued at any node point, those values being the different constant values of B in each element surrounding the node point. The general approach to estimating the "true" value of B at any node point is to fit a least-squares plane through

the values of B at the Gauss points of all elements that surround a node of interest, and to take the value of the plane at the node point's location as its smoothed value of B [13].

However, this approach to patch recovery has a lot of shortcomings. For the rather irregular meshes that can arise in finite elements, the least-squares fit problem can be ill-conditioned, or even singular, at some nodes in the finite element mesh. Furthermore, the superconvergence solution can actually be less accurate than the piece-wise constant solution in the neighborhood of boundaries and interfaces.

One can note that the patch recovery method is merely a weighted average of the flux densities in all of the elements surrounding a given node. Instead of a least-squares fit, FEMM simply weights the values of flux density in each adjacent element's Gauss point with a value inversely proportional to the distance from the Gauss point to the node point of interest. Away from boundaries, the results seem to be nearly as good as a least-squares fit. At boundaries and interfaces, the smoothed solution is no worse than the unsmoothed solution.

References

- [1] M. Plonus, *Applied electromagnetics*. McGraw-Hill, 1978.
- [2] S. R. Hoole, *Computer-aided analysis and design of electromagnetic devices*, Elsevier, 1989.
- [3] J. D. Jackson, *Classical electrodynamics*, 2nd ed, Wiley, 1975.
- [4] R. L. Stoll, *The analysis of eddy currents*, Oxford University Press, 1974.
- [5] S. McFee, J. P. Webb, and D. A. Lowther, “A tunable volume integration formulation for force calculation in finite-element based computational magnetostatics,” *IEEE Transactions on Magnetics*, 24(1):439-442, January 1988.
- [6] F. Henrotte, G. Deliege, and K. Hameyer, “The eggshell method for the computation of electromagnetic forces on rigid bodies in 2D and 3D,” CEFC 2002, Perugia, Italy, April 16-18, 2002. (pdf version)
- [7] P. E. Allaire, *Basics of the finite element method*, 1985.
- [8] P. P. Silvester, *Finite elements for electrical engineers*, Cambridge University Press, 1990.
- [9] F. Henrotte *et al*, “A new method for axisymmetric linear and nonlinear problems,” *IEEE Transactions on Magnetics*, MAG-29(2):1352-1355, March 1993.
- [10] C. A. Fletcher, *Computational techniques for fluid dynamics*, Springer-Verlag, 1988.
- [11] R. W. Freund, “Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices,” *SIAM Journal of Scientific and Statistical Computing*, 13(1):425-448, January 1992.
- [12] E. F. D’Azevedo, P. A. Forsyth, and W. Tang, “Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems,” *SIAM J. Matrix Anal. Appl.*, 12(4), July 1992.
- [13] O. C. Zienkiewicz and J. Z. Zhu, “The superconvergent patch recovery and *a posteriori* estimates, part 1: the recovery technique,” *International Journal for Numerical Methods in Engineering*, 33:1331-1364, 1992.
- [14] Q. Chen and A. Konrad, “A review of finite element open boundary techniques for static and quasistatic electromagnetic field problems,” *IEEE Transactions on Magnetics*, 33(1):663-676, January 1997.
- [15] E. M. Freeman and D. A. Lowther, “A novel mapping technique for open boundary finite element solutions to Poissons equation,” *IEEE Transactions on Magnetics*, 24(6):2934-2936, November 1988.

- [16] D. A. Lowther, E. M. Freeman, and B. Forghani, "A sparse matrix open boundary method for finite element analysis," *IEEE Transactions on Magnetics*, 25(4):2810-2812, July 1989.
- [17] E. M. Freeman and D. A. Lowther, "An open boundary technique for axisymmetric and three dimensional magnetic and electric field problems," *IEEE Transactions on Magnetics*, 25(5):4135-4137, September 1989.
- [18] A. G. Jack and B. C. Mecrow, "Methods for magnetically nonlinear problems involving significant hysteresis and eddy currents," *IEEE Transactions on Magnetics*, 26(2):424-429, March 1990.

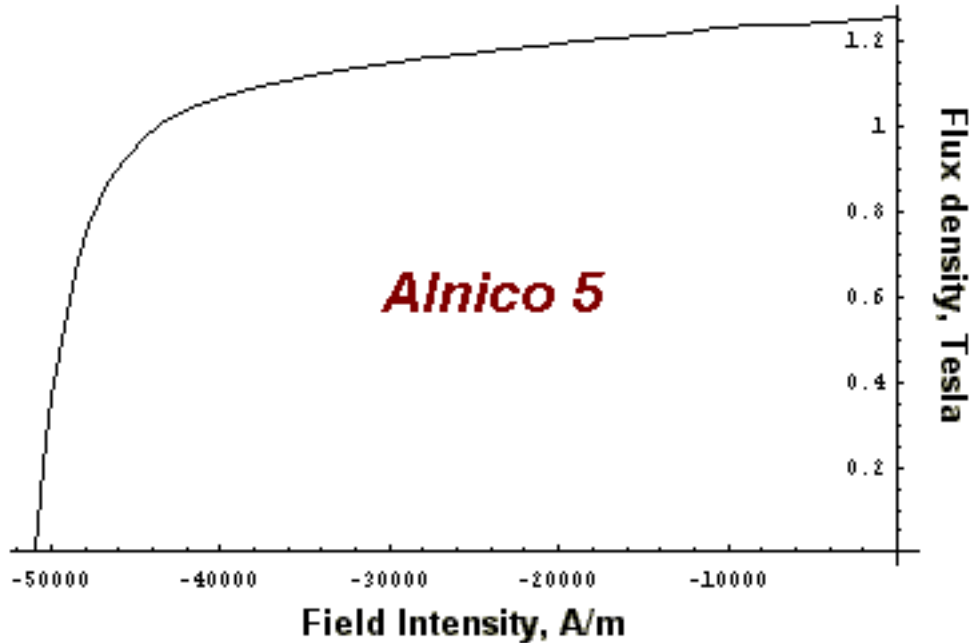


Figure 30: Sample demagnetization curve for Alnico 5

A Appendix

A.1 Modeling Permanent Magnets

FEMM accommodates permanent magnets, but there are some special rules associated with properly modeling them. This appendix will explain how to distill enough information from a manufacturer’s literature to properly define the material in FEMM.

The manufacturer provides information about their material in the form of a demagnetization curve. A sample curve for Alnico 5 is pictured in Figure 30. The task is to get the appropriate information out of the curve put in a FEMM Block Property model.

Magnets can be modeled from several different, but equally valid, points of view. From the perspective finite element analysis, the most useful model is to think of the magnet as a volume of ferromagnetic material surrounded by a thin sheet of current, as shown in Figure 31. From this point of view, the demagnetization curve is what occurs when different amounts of magnetomotive force are applied to a long magnet, acting in the direction opposing the field of the magnet. When enough MMF is applied so that the field is exactly cancelled out, the applied MMF must be exactly the same as the MMF that is driving the magnet. The B-H profile that is traversed on the way to the $B = 0$ point is just the B-H curve of the material inside the magnet.

Using these insights, the permanent magnet can be modeled. The *coercivity* (denoted H_c) of the magnet is the absolute value of the MMF that it takes to bring the the field in the magnet to zero. This value (in units of Amps/Meter) is entered in the H_c box in the Block Property dialog (see Figure 10). If the magnet material is nonlinear, the appropriate values to enter in the B-H data dialog can be obtained by shifting the curve to the right by exactly H_c ,

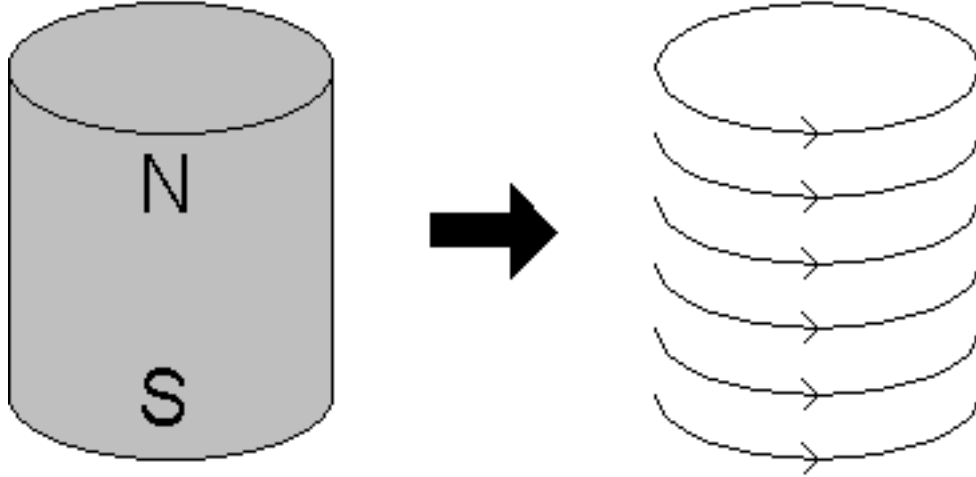


Figure 31: Magnet as an equivalent current sheet.

so that the $B = 0$ point lines up with the origin. For example, the shifted demagnetization curve corresponding to Alnico 5 is pictured in Figure 32. If the demagnetization curve is straight enough to be considered linear, one can obtain the appropriate permeability by taking the slope of the demagnetization curve.

Strong rare-earth materials at room temperature have a very linear demagnetization curve. Usually, a linear model is sufficient for these materials. In addition, these materials have a relative permeability very close to 1. The modeling of these materials can be simplified (while only incurring small errors) by assuming that the permeability is exactly 1. Then, if you know the energy product of the magnet material in units of MGOe (the unit in which the energy product is almost always given), the appropriate H_c can be determined via (32).

$$H_c = \frac{5(10^5)\sqrt{E}}{\pi} \quad (32)$$

where E is the energy product in MGOe.

With alnico magnets, great care must be taken in interpreting the finite element results. Unlike rare-earth magnets, these magnets exhibit a great degree of hysteresis when they are demagnetized. That is, when the flux density is pushed below the “knee” in the demagnetization curve, the flux level does not recover to the previous magnitude when the opposing MMF is removed. This hysteresis is illustrated in Figure 33. This sort of demagnetization and recoil can occur when the magnets are being handled prior to assembly into a device. In a motor, the magnets will demagnetize somewhat when the motor is first started. They will eventually end up running back and forth along a recoil line that is below the “virgin” demagnetization curve. The point is that the modeler cannot be sure exactly where the magnets are operating—an analysis that takes this sort of hysteresis into account is beyond the scope of FEMM. Note, however, that this caution applies only for nonlinear magnets; for practical purposes, rare-earth magnets generally do not exhibit this sort of hysteresis behavior.

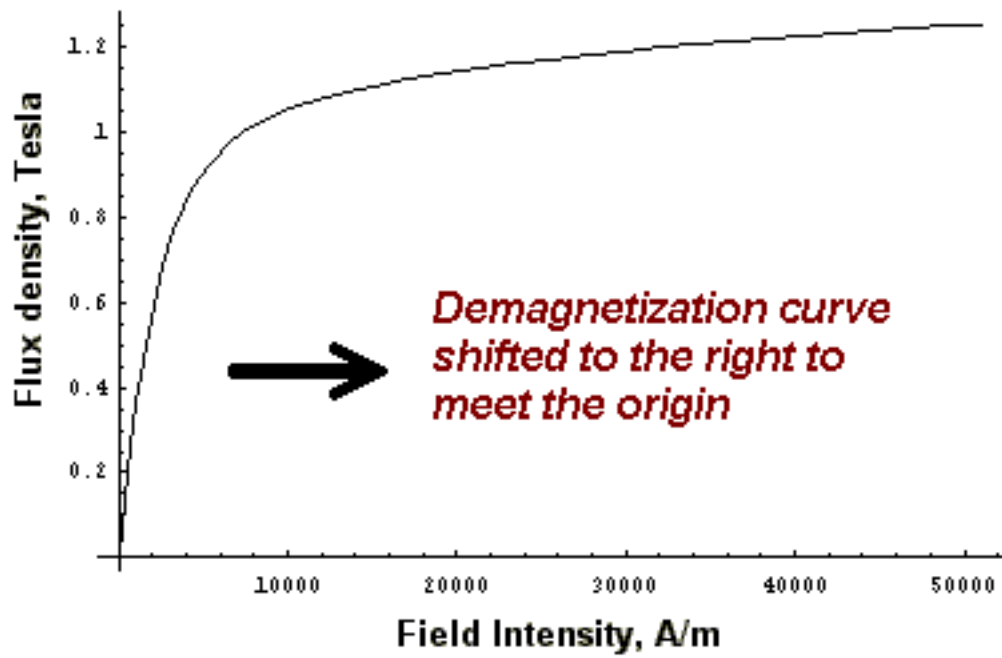


Figure 32: Shifting the B-H curve of a permanent magnet

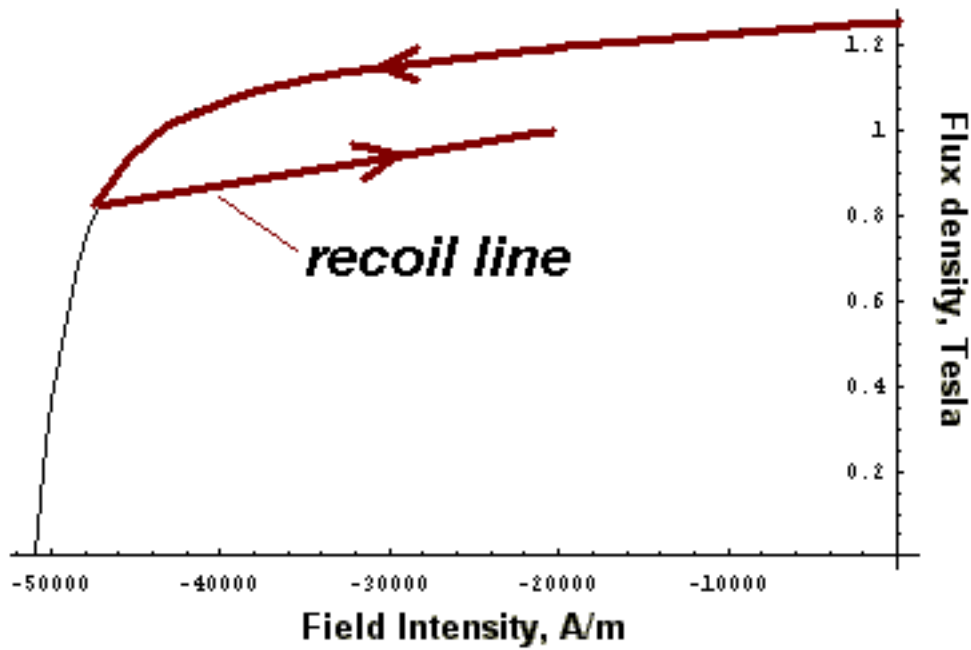


Figure 33: Recoil in partially demagnetized Alnico 5.

A.2 Bulk Lamination Modeling

A great number of magnetic devices employ cores built up out of thin laminations for the purpose of reducing eddy current effects. One way to model these materials within a finite element framework would be to model each discrete lamination (and the insulation between laminations) in the finite element geometry. An alternative is to treat the laminated material as a continuum and derive bulk properties that yield essentially the same results, while requiring a much less elaborate finite element mesh. FEMM has implemented this bulk approach to laminations.

Consider that the flux can flow through the lamination in a combination of two ways: via the “easy” direction down the laminations, or the “hard” way, across the thickness of the laminations. The hard direction is difficult for flux for two reasons. First, the rolling process makes the iron somewhat less permeable than in the easy direction. Second, and most importantly, the flux must traverse the insulation between laminations, which typically has a unit permeability.

The first assumption in deriving the bulk permeability model is that the permeability in the iron itself is isotropic. This isn’t quite true, but almost all of the reluctance in the hard direction results from crossing the gap between laminations. Having a significant error in the hard direction permeability in the iron itself only results in a trivial change in the bulk reluctance in the cross-lamination direction.

Armed with this assumption, a circuit model can be produced for each direction of flux travel. For the easy direction, the circuit model is pictured in Figure 34. There are two reluctances in parallel—one for flux that flows through the iron part of the laminations:

$$R_{ez,fe} = \frac{L}{\mu_r \mu_o c W} \quad (33)$$

and another reluctance for flux that flows through the air between laminations:

$$R_{ez,air} = \frac{L}{\mu_o(1 - c)W} \quad (34)$$

where L and W are the length and width of the path traversed, and c is the fraction of the path filled with iron. Adding these two reluctances in parallel yields:

$$R_{ez} = \frac{L}{((1 - c) + c\mu_r)\mu_o} W \quad (35)$$

Since L and W are arbitrarily chosen, the bulk permeability of the section is:

$$\mu_{ez} = ((1 - c) + c\mu_r)\mu_o \quad (36)$$

For the solution of nonlinear problems, the derivation of the changes to Newton’s method to accomodate the bulk lamination model are greatly simplified if it is assumed that $(1 - c) \ll c\mu_r$. In this case, μ_{ez} can be approximated as:

$$\mu_{ez} \approx c\mu_r\mu_o \quad (37)$$

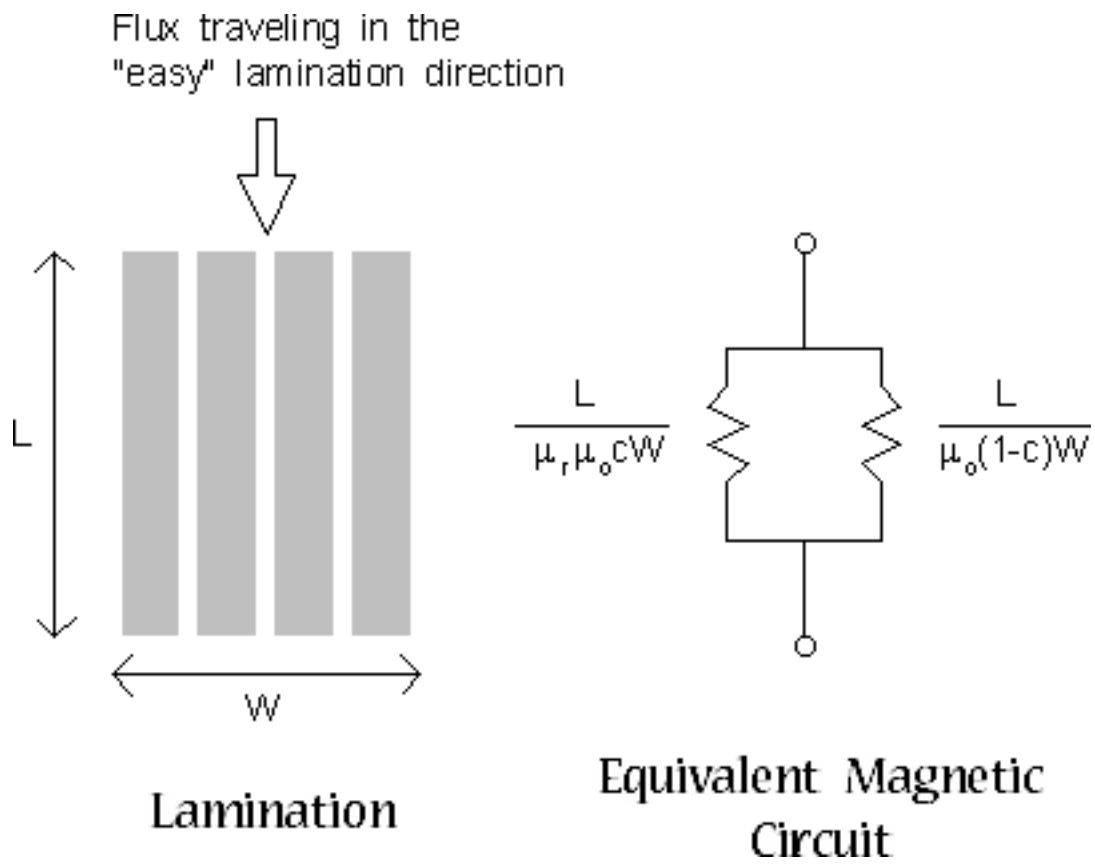


Figure 34: Equivalent circuit for flux in the "easy" direction

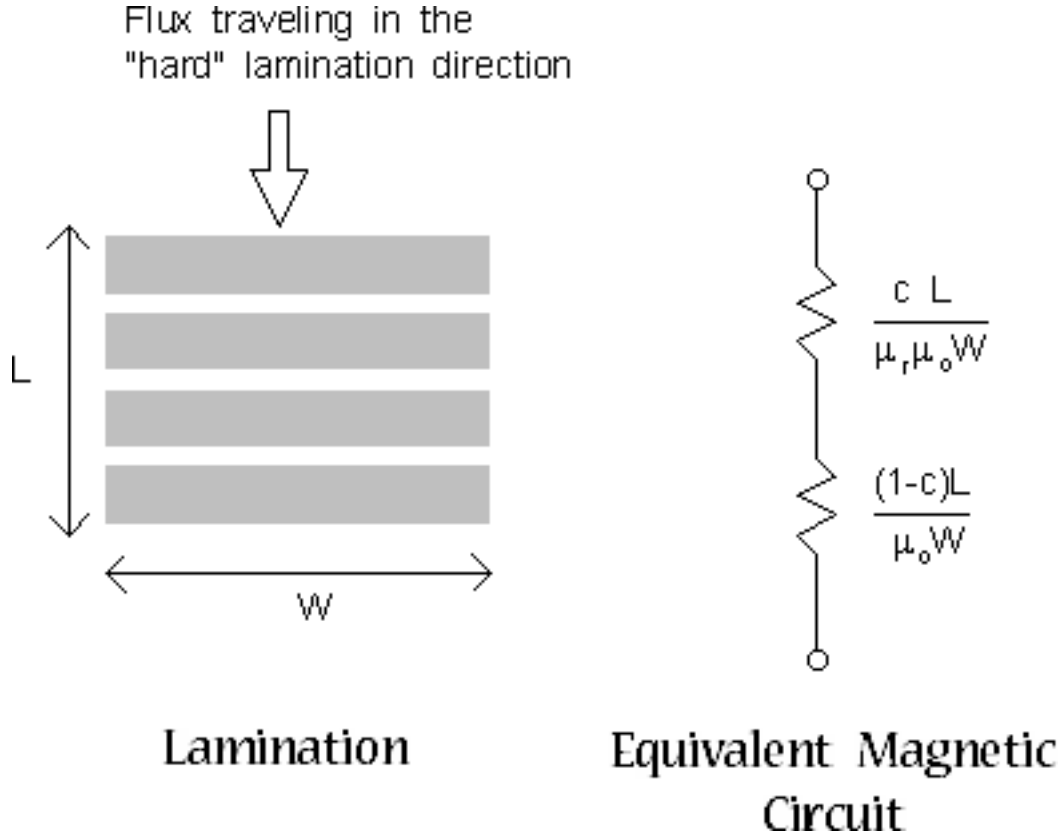


Figure 35: Equivalent circuit for flux in the “hard” direction.

This approximation leads to only trivial errors until the fill factor approaches zero. For example, if $\mu_r = 1000$ with a 90% fill, the difference between (36) and (37) is only about 0.01%.

For the hard direction, a different equivalent circuit, pictured in Figure 35 can be drawn. In this case, the circuit is two reluctances in series, as the flux has to cross the insulation and the lamination in succession. These reluctances are:

$$R_{hard,fe} = \frac{cL}{\mu_r \mu_o W} \quad (38)$$

$$R_{hard,air} = \frac{(1-c)L}{\mu_o W} \quad (39)$$

Adding these two reluctances together in series yields:

$$R_{hard} = \left(\frac{c + (1-c)\mu_r}{\mu_r \mu_o} \right) \frac{L}{W} \quad (40)$$

Since L and W are arbitrary, the bulk permeability in the hard direction is:

$$\mu_{hard} = \frac{\mu_r \mu_o}{c + (1-c)\mu_r} \quad (41)$$

If the material is laminated “in-plane,” all flux is flowing in the easy direction, and (37) is used as the permeability for each element. In problems that are laminated parallel to x or y , (37) and (41) are used as permeabilities in the standard fashion for elements with an anisotropic permeability.

For harmonic problems, eddy currents flow in the laminations, and hysteresis causes additional loss. If the laminations are thin compared to the other dimensions of the geometry, the effects of eddy currents and hysteresis can be encapsulated in a frequency-dependent permeability [4]. In this case, the magnetostatic permeability, μ_r is simply replaced by the frequency-dependent permeability μ_{fd} in (37) and (41):

$$\mu_{fd} = \frac{\mu_r e^{-\frac{j\phi_h}{2}} \tanh \left[e^{-\frac{j\phi_h}{2}} \sqrt{j\omega\sigma\mu_r\mu_o\frac{d}{2}} \right]}{\sqrt{j\omega\sigma\mu_r\mu_o\frac{d}{2}}} \quad (42)$$

In (42), ϕ_h represents a constant phase lag between B and H due to hysteresis, σ is the conductivity of the lamination material, d is the thickness of the iron part of the lamination, and ω is the frequency of excitation in rad/s. Note that the concept of hysteresis-induced lag can be applied to non-laminated materials as well, simply by multiplying the magnetostatic permeability by $e^{-j\phi_h}$ for harmonic problems.

A.3 Open Boundary Problems

Typically, finite element methods are best suited to problems with well-defined, closed solution regions. However, a large number of problems that one might like to address have no natural outer boundary. A prime example is a solenoid in air. The boundary condition that one would *like* to apply is $A = 0$ at $r = \infty$. However, finite element methods, by nature, imply a finite domain. Fortunately, there are methods that can be applied to get solutions that closely approximate the “open boundary” solution using finite element methods.

A.3.1 Truncation of Outer Boundaries

The simplest, but least accurate, way to proceed is to pick an arbitrary boundary “far enough” away from the area of interest and declare either $A = 0$ or $\partial A/\partial n = 0$ on this boundary. According to [14], a rule of thumb is that the distance from the center of the problem to the outer boundary should be at least five times the distance from the center to the outside of the objects of interest. Truncation is the method employed by most magnetics finite element programs, because it requires no additional effort to implement.

The down side to truncation is that get an accurate solution in the region of interest, a volume of air much larger than the region of interest must also be modeled. Usually, this large region exterior to the area of interest can be modeled with a relatively coarse mesh to keep solution times to a minimum. However, some extra time and space is still required to solve for a region in which one has little interest.

A.3.2 Asymptotic Boundary Conditions

A thorough review of open boundary techniques is contained in [14]. Perhaps the simple way to approximate an “open” boundary (other than truncation) described in [14] is to use

asymptotic boundary conditions. The result is that by carefully specifying the parameters for the “mixed” boundary condition, and then applying this boundary condition to a circular outer boundary, the unbounded solution can be closely approximated. An example that employs an asymptotic boundary condition to obtain an unbounded field solution is the `axi1.fem` example included in the distribution.

Consider a 2-D planar problem in polar coordinates. The domain is a circular shell of radius r_o in an unbounded region. As $r \rightarrow \infty$, vector potential A goes to zero. On the surface of the circle, the vector is a prescribed function of θ . This problem has an analytical solution, which is:

$$A(r, \theta) = \sum_{m=1}^{\infty} \frac{a_m}{r^m} \cos(m\theta + \alpha_m) \quad (43)$$

where the a_m and α_m parameters are chosen so that the solution matches the prescribed potential on the surface of the circle.

One could think of this solution as describing the solution exterior to a finite element problem with a circular outer boundary. The solution is described inside the circle via a finite element solution. The trick is to knit together the analytical solution outside the circle to the finite element solution inside the circle.

From inspecting (43), one can see that the higher-numbered harmonic, the faster the magnitude of the harmonic decays with respect to increasing r . After only a short distance, the higher-numbered harmonics decay to the extent that almost all of the open-space solution is described by only the leading harmonic. If n is the number of the leading harmonic, the open-field solution for large, but not infinite, r is closely described by:

$$A(r, \theta) \approx \frac{a_n}{r^n} \cos(n\theta + \alpha_n) \quad (44)$$

Differentiating with respect to r yields:

$$\frac{\partial A}{\partial r} = -\frac{na_n}{r^{n+1}} \cos(n\theta + \alpha_n) \quad (45)$$

If (45) is solved for a_n and substituted into (44), the result is:

$$\frac{\partial A}{\partial r} + \left(\frac{n}{r}\right) A = 0 \quad (46)$$

Now, (46) is a very useful result. This is the same form as the “mixed” boundary condition supported by FEMM. If the outer edge of the solution domain is circular, and the outer finite element boundary is somewhat removed from the area of primary interest, the open domain solution can be closely approximated by applying (46) the circular boundary.

To apply the Asymptotic Boundary Condition, define a new, mixed-type boundary condition. Then, pick the parameters so that:

$$c_0 = \frac{n}{\mu_o r_o} \quad (47)$$

$$c_1 = 0 \quad (48)$$

where r_o is the outer radius of the region in meters (regardless of the working length units), and $\mu_o = 4\pi(10^{-7})$.

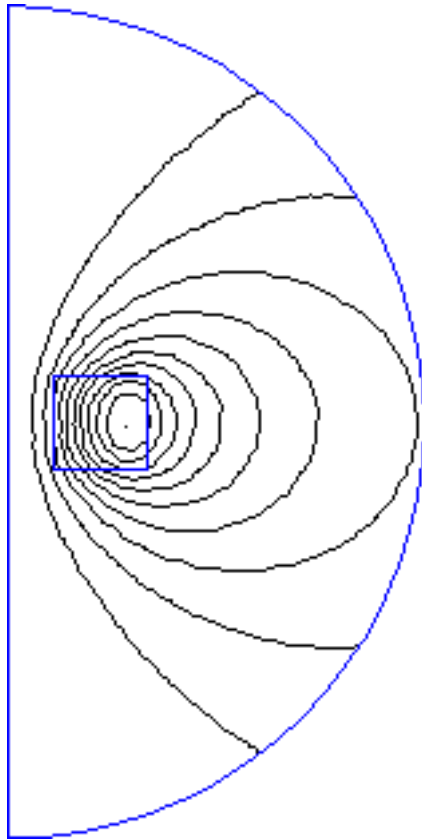


Figure 36: Air-cored coil with “open” boundary condition

Although the above derivation was specifically for 2-D problems, it turns out that when the same derivation is done for the axisymmetric case, the definition of the mixed boundary condition coefficients are exactly the same as (47).

Some care must be used in applying this boundary condition. Most of the time, it is sufficient to take $n = 1$ (*i.e.* the objects in the solution region look like a dipole when viewed from a large distance). However, there are other cases (*e.g.* a 4-pole halbach permanent magnet array) in which the leading harmonic is something other than $n = 1$. You need to use your insight into your specific problem to pick the appropriate n for the leading harmonic. You also must put the objects of interest roughly in the center of the circular finite element domain to minimize the magnitude higher-order field components at the outer boundary.

Although the application of this boundary condition requires some thought on the part of the user, the results can be quite good. Figure 36, corresponding to the `axi1` example, represents the field produced by an air-cored coil in free space. The asymptotic boundary condition has been applied to the circular outer boundary. Inspecting the solution, flux lines appear to cross the circular boundary as if the solution domain were truly unbounded.

A quick note on computational efficiency: applying the absorbing boundary condition imposes no additional computing cost on the problem. The ABC is computationally no more time-consuming to apply than enforcing $A = 0$ at the outer boundary. Solution times for the PCG solver are equivalent in either case. It can also readily be derived that the ABC works exactly the same for harmonics problems. (To see this, just assume that the a_m in

(43) can be complex valued, and follow the same derivation).

A.3.3 Kelvin Transformation

Introduction

A particularly good approach to “open boundary” problems is the Kelvin Transformation, a technique first discussed in the context of computational magnetics in [15] and [16]. The strengths of this technique are:

- the effects of the exterior region are, in theory, exactly modeled by this approach;
- a sparse matrix representation of the problem is retained (unlike FEM-BEM methods, which give the same “exact solution” but densely couples together the boundary nodes).
- requires no “special” features in the finite element solver to implement the technique, other than the ability to apply periodic boundary conditions.

The purposes of this note are to explain what the Kelvin transformation is derived and to show how it is implemented in the context of the FEMM finite element program.

Derivation

In the “far field” region, the material is typically homogeneous (*e.g.* *air* and free of sources. In this case, the differential equation that describes vector potential A is the Laplace equation:

$$\nabla^2 A = 0 \tag{49}$$

If we write (49) in polar notation, A is described by:

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial A}{\partial r} \right) + \frac{1}{r^2} \frac{\partial^2 A}{\partial \theta^2} = 0 \tag{50}$$

Assume that the “near field” region of the problem can be contained in a circle of radius r_o centered at the origin. The far-field region is then everything outside the circle.

One approach to unbounded problems is to attempt to map the unbounded region onto a bounded region, wherein problems can more easily be solved. Specifically, we desire a way to transform the unbounded region outside the circle into a bounded region. One simple way to make such a mapping is to define another variable, R , that is related to r by:

$$R = \frac{r_o^2}{r} \tag{51}$$

By inspecting (51), it can be seen that this relationship maps the exterior region onto a circle of radius r_o .

The next step is to transform (49), the differential equation that the field must satisfy, into the mapped space. That is, (49) must be written in terms of R and θ rather than r and θ . We can evaluate derivatives in terms of R instead of r by employing the chain rule:

$$\frac{\partial}{\partial r} = \frac{\partial}{\partial R} \left(\frac{dR}{dr} \right) = -\frac{\partial}{\partial R} \left(\frac{R}{r_o} \right)^2 \tag{52}$$

Now, we can note that at $r = R = r_o$,

$$\frac{\partial A}{\partial r} = -\frac{\partial A}{\partial R} \quad (53)$$

and we can substitute (52) into (49) to yield, after some algebraic manipulation:

$$\frac{1}{R} \frac{\partial}{\partial R} \left(R \frac{\partial A}{\partial R} \right) + \frac{1}{R^2} \frac{\partial^2 A}{\partial \theta^2} = 0 \quad (54)$$

Eq. (54), the transformed equation for the outer region, has exactly the same form as inner region, only in terms of R rather than r . The implication is that for the 2-D planar problem, the exterior can be modeled simply by creating a problem domain consisting of two circular regions: on circular region containing the items of interest, and an additional circular region to represent the “far field.” Then, periodic boundary conditions must be applied to corresponding edges of the circle to enforce the continuity of A at the edges of the two regions. The is continuity of A at the boundary between the exterior and interior regions. For a finite element formulation consisting of first-order triangles, (53) is enforced automatically at the boundaries of the two regions. The second circular region exactly models the infinite space solution, but does it on a bounded domain—one could always back out the field for any point in space by applying the inverse of (51).

Kelvin Transformation Example – open1.fem

As an example, consider an E-core lamination stack with a winding around it. Suppose that the objective is to determine the field around the E-core in the absence of any flux return path (*i.e.* when the magnetic circuit is open). In this case, the flux is not constrained to flow in a path that is *a priori* well defined, because the laminations that complete the flux path have been removed.

The geometry was chosen arbitrarily, the purpose here being more the procedure than the actual problem. The E-core was chosen to have a 0.5” thick center leg, 0.25” thick outer legs, and a slot depth of 0.75”. The material for the core is linear with a relative permeability of 2500. The coil carries a bulk current density of 2 MA/m². The input geometry is picture in Figure 37.

In Figure 37, the core is placed within a circular region, and a second circular region is drawn next to the region containing the core. Periodic boundary conditions are applied to the arcs that define the boundaries as shown in Figure 37. The way that periodic boundary conditions are implemented in FEMM, each periodic boundary condition defined for the problem is to be applied to two and only two corresponding entities. In this case, each boundary circle is composed of two arcs, so two periodic boundary conditions must be defined to link together each arc with in the domain with the core to its corresponding arc in the domain representing the exterior region.

Also notice that a point has been drawn in the center of the exterior region. A point property has been applied to this point that specifies that $A = 0$ at this reference point. The center of the circle maps to infinity in the analogous open problem, so it makes sense to define, in effect, $A = 0$ at infinity. If no reference point is defined, it is fairly easy to see that

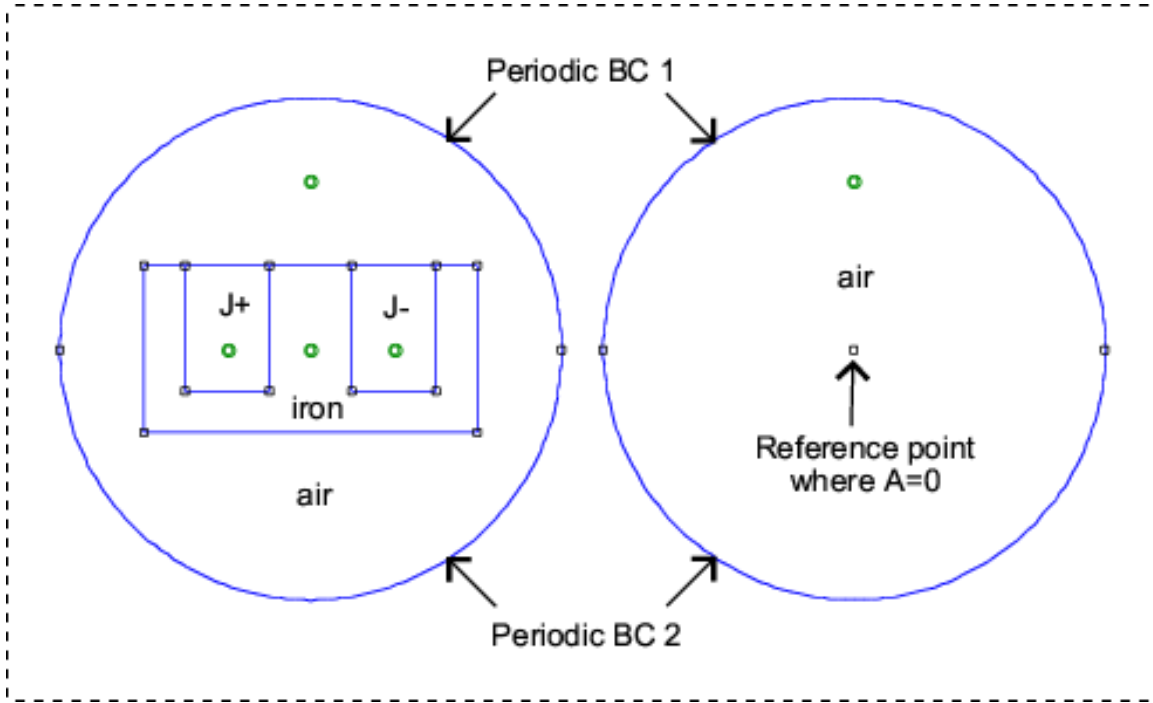


Figure 37: Example input geometry.

the solution is only unique to within a constant. The situation is analogous to a situation where Neumann boundary conditions have been defined on all boundaries, resulting in a non-unique solution for A . Due to the type of solver that FEMM employs, the problem can most likely be solved even if a reference point is not defined. However, defining a reference point eliminates the possibility of numerical difficulties due to uniqueness issues.

The resulting solution is shown in Figure 38. As is the intention, the flux lines appear to cross out of the of the region containing the core as if unaffected by the presence of the boundary. The flux lines reappear in the domain representing the exterior region, completing their flux paths through the exterior region.

A.4 Nonlinear Time Harmonic Formulation

Starting with the the 3.3 version of FEMM, the program includes a “nonlinear time harmonic” solver. In general, the notion of a ”nonlinear time harmonic” analysis is something of a kludge. To obtain a purely sinusoidal response when a system is driven with a sinusoidal input, the system must, by definition, be linear. The nonlinear time harmonic analysis seeks to include the effects of nonlinearities like saturation and hysteresis on the fundamental of the response, while ignoring higher harmonic content. This is a notion similar to “describing function analysis,” a widely used tool in the analysis of nonlinear control systems. There are several subtly different variations of the formulation that can yield slightly different results, so documentation of what has actually been implement is important to the correct interpretation of the results from this solver.

An excellent description of this formulation is contained in [18]. FEMM formulates the

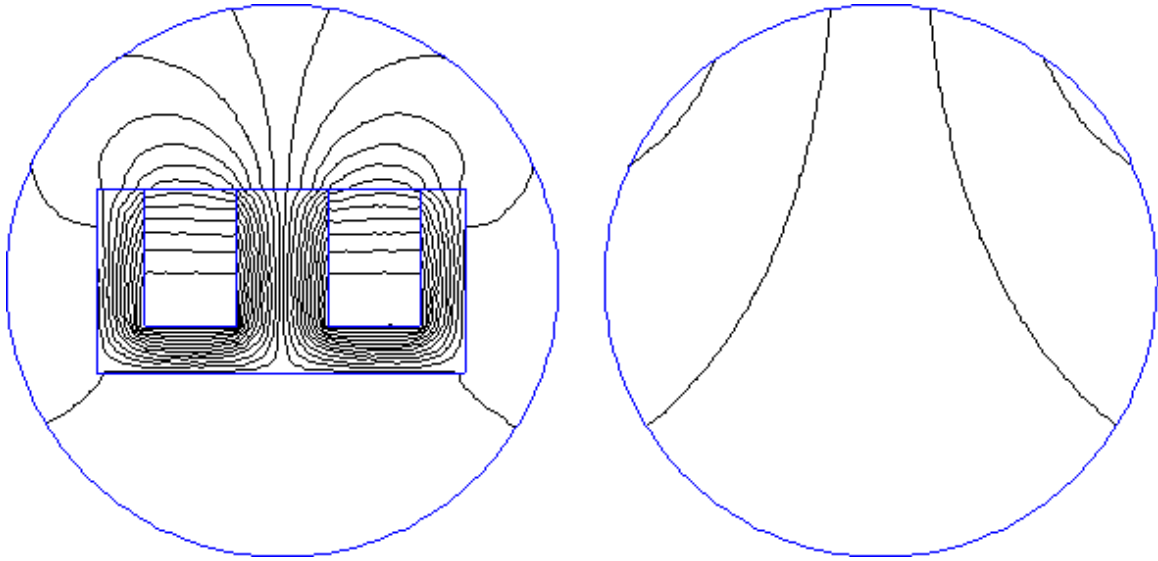


Figure 38: Solved problem.

nonlinear time harmonic problem as described in this paper. Similar to Jack and Mecrow, FEMM derives an apparent BH curve by taking H to be the sinusoidally varying quantity. The amplitude of B is obtained by taking the first coefficient in a Fourier series representation of the resulting B . For the purposes of this Fourier series computation, FEMM interpolates linearly between the user-defined points on the BH curve to get a set of points with the same H values as the input set, but with an adjusted B level. The rationale for choosing H to be the sinusoidal quantity (rather than B) is that choosing B to be sinusoidal shrinks the defined BH curve—the B values stay fixed while the H values become smaller. It then becomes hard to define a BH curve that does not get interpolated. In contrast, with H sinusoidal, the B points are typically larger than the DC flux density levels, creating a curve with an expanded range.

A "nonlinear hysteresis lag" parameter is then applied to the effective BH curve. The lag is assumed to be proportional to the permeability, which gives a hysteresis loss that is always proportional to $|B|^2$. This form was suggested in an old O'Kelly paper (reference?). It has been suggested that that the Steinmetz equation could be used to specify hysteresis lag, but the Steinmetz equation is badly behaved at low flux levels (i.e. one can't solve for a hysteresis lag that produces the Steinmetz $|B|^{1.6}$ form for the loss as B goes to zero.)

For nonlinear in-plane laminations, an additional step is taken to obtain an effective BH curve that also includes eddy current effects. At each H level on the user-defined BH curve, a 1D nonlinear time harmonic finite element problem is solved to obtain the total flux that flows in the lamination as a function of the H applied at the edge of the lamination. Then dividing by the lamination thickness and accounting for fill factor, an effective B that takes into account saturation, hysteresis, and eddy currents in the lamination is obtained for each H .