MathFEMM 1.20

The Mathematica interface to FEMM 4.2
David Meeker
dmeeker@ieee.org

January 26, 2006

■ Installation and Setup

The MathFEMM package is automatically installed with FEMM 4.2. If you have accepted the default install for FEMM (to c:\Program Files\femm42) no further initialization is required. If you have installed FEMM to a different directory, you will need to perform an additional initialization step so that MathFEMM can find the FEMM executable:
1) Start up Mathematica.
2) Load the MathFEMM package from the installation directory with the command:
<<c:\progra~1\femm42\mathfemm\mathfemm.m
3) Run the SetPathToFEMM command to specify the path to the FEMM executable:
SetPathToFEMM["c:\\progra~1\\femm42\\bin\\femm.exe"]
where you'd replace the above path withe the one to femm.exe in your installation.Be sure to use double backslashes in the path specification

USE:To start an MathFEMM session,first load up the MathFEMM package using the command:
<<c:\progra~1\femm42\mathfemm\mathfemm.m
You may need to change the path to mathfemm.m in the above call if you chose to install in a directory other than the default one.Then,use the OpenFEMM[] to automatically FEMM process and connect to it. Subsequent MathFEMM commands are sent to that instance of FEMM.When you are finished sending commands to FEMM,you can close down FEMM with the CloseFEMM[] function.There are a number of example notebooks of various MathFEMM analyses in the examples subdirectory. To run them,simply open up the notebooks and select Kernel|Evaluation|Evaluate Notebook off of the Mathematica main menu. The included notebook Usage.nb describes all MathFEMM commands in detail.

■ Common Commands

OpenFEMM[] starts an external instance of FEMM and establishes an associated MathLink connection.

CloseFEMM[] shuts down the external instance of FEMM and closes the associated MathLink connection.

```
NewDocument[doctype] creates a new preprocessor document and opens up a new preprocessor window
  in FEMM. Specify doctype to be 0 for a magnetics problem, 1 for an electrostatics problem, 2
  for a heat flow problem, or 3 for a current flow problem. Alternate form is Create[doctype]

OpenDocument["filename"] opens a document specified by "filename".

ShowPointProps[] displays the Point Properties dialog

HidePointProps[] hides the Point Properties dialog from view

ShowConsole[] displays the FEMM Lua console

HideConsole[] hides the FEMM Lua console from view

MessageBox["string"] displays a pop-up messagebox with the message "string"

Prompt["string"] displays a pop-up dialog with the message "string" displayed on it
  with an edit box.  The return value of the funciton is the contents of the edit box.

MLPut["string"] sends "string" to FEMM's Lua interpreter via MathLink

MLGet[] receives data from FEMM via MathLink.  The data
  is always in the form of a list, but the number of elements in the list
  is variable.  The items in the list could be either numbers or strings

AWG[awg] returns the wire diameter in mm for the specified AWG gauge

MathFEMM`IEC

IEC[MathFEMM`Private`iec_] := 7.95916 MathFEMM`Private`exp(-0.115197 MathFEMM`Private`iec)
```

■ Electrostatics Input Commands

■ Define, Modify, or Delete Properties

```
EIAddBoundProp["boundpropname", Vs,qs,c0,c1,
  BdryFormat] adds a new boundary property with name "boundpropname"
 For a "Fixed Temperature" type boundary condition, set the Vs
  parameter to the desired voltage and all other parameters to zero.
 To obtain a "Mixed" type boundary condition, set C1 and C0 as
  required and BdryFormat to 1. Set all other parameters to zero.
 To obtain a prescribes surface charge density, set qs to the
  desired charge density in C/m2 and set BdryFormat to 2.
 For a "Periodic" boundary condition, set BdryFormat to 3 and set all other parameters to zero.
 For an "Anti-Perodic" boundary condition, set BdryFormat to 4 set all other parameters to zero.

EIAddConductorProp["conductorname",Vc,qc,conductortype] adds a
  new conductor property with name "conductorname" with either a prescribed
  voltage or a prescribed total charge. Set the unused property to zero. The
  conductortype parameter is 0 for prescribed charge and 1 for prescribed voltage.

EIAddMaterial["materialname",ex,ey,qv] adds a new
  material with called "materialname" with the material properties:
 ex Relative permittivity in the x- or r-direction.
 ey Relative permittivity in the y- or z-direction.
 qv Volume charge density in units of C/m^3
```

EIAddPointProp["pointpropname",Vp,qp] adds a new point property of name "pointpropname" with either a specified potential Vp a point charge density qp in units of C/m.

EIModifyBoundProp["BdryName",propnum,value] 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:
0 – "BdryName" – Name of boundary property
1 – Vs – Fixed Voltage
2 – qs – Prescribed charge density
3 – c0 – Mixed BC parameter
4 – c1 – Mixed BC parameter
5 – BdryFormat – Type of boundary condition (0 = Prescribed V; 1 = Mixed; 2 = Surface charge density; 3 = Periodic; 4 = Antiperiodic)

EIModifyConductorProp["ConductorName",propnum,value] allows for modification of a conductor property. The conductor property to be modified is specified by "ConductorName". 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:
0 – "ConductorName" – Name of the conductor property
1 – Vc – Conductor voltage
2 – qc – Total conductor charge
3 – ConductorType – 0 = Prescribed charge, 1 = Prescribed voltage

EIModifyMaterial["BlockName",propnum,value] 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:
0 – "BlockName" – Name of the material 1 – ex – x (or r) direction relative permittivity 2 – ey – y (or z) direction relative permittivity 3 – qs – Volume charge

EIModifyPointProp["PointName",propnum,value] 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:
0 – PointName – Name of the point property
1 – Vp – Prescribed nodal voltage
2 – qp – Point charge density in C/m

EIDeleteBoundProp["boundpropname"] deletes the boundary property named "boundpropname".

EIDeleteConductor["conductorname"] deletes the conductor property named "conductorname".

EIDeleteMaterial["materialname"] deletes the material property named "materialname".

EIDeletePointProp["pointpropname"] deletes the point property named "pointpropname".

■ Object Drawing Commands

EIAddArc[x1,y1,x2,y2,angle,maxseg] adds an arc to the electrostatics input geometry from the point nearst to {x1,y1} to the point nearest to {x2,y2}. The arc spans a number of degrees specified by angle. Since FEMM approximates arcs by many line segments, the parameter maxseg specifies the maximum number of degrees that is allowed to be spanned by any one segment. Equivalent forms are:
EIAddArc[{x1,y1},{x2,y2},angle,maxseg]
EIAddArc[{{x1,y1},{x2,y2}},angle,maxseg]

EIAddBlockLabel[x,y] adds a block label at the point {x,y}. An equivalent form is:
EIAddBlockLabel[{x,y}]

```
EIAddNode[x,y] adds a new node at {x,y}. An equivalent form is:
 EIAddNode[{x,y}]

EIAddSegment[x1,y1,x2,y2] add a new line segment from node
  closest to {x1,y1} to node closest to {x2,y2}.  Equivalent forms are:
 EIAddSegment[{x1,y1},{x2,y2}]
 EIAddSegment[{{x1,y1},{x2,y2}}]

EIDrawArc[x1,y1,x2,y2,angle,maxseg] adds an arc to the electrostatics input geometry by
  drawing points at {x1,y1} and {x2,y2} and then connecting them with an arc segment.
  The arc spans a number of degrees specified by angle. Since FEMM approximates
  arcs by many line segments, the parameter maxseg specifies the maximum number of
  degrees that is allowed to be spanned by any one segment. Equivalent forms are:
 EIDrawArc[{x1,y1},{x2,y2},angle,maxseg]
 EIDrawArc[{{x1,y1},{x2,y2}},angle,maxseg]

EIDrawLine[x1,y1,x2,y2] adds points at {x1,y1} and {x2,y2} and
  then adds a segment connecting these two points.  Equivalent forms are:
 EIDrawLine[{x1,y1},{x2,y2}]
 EIDrawLine[{{x1,y1},{x2,y2}}]

EIDrawPolygon[{{x1,y2},...,{xn,yn}}] adds new node points at
  every listed point and then draws a closed figure that connects the points

EIDrawPolyLine[{{x1,y2},...,{xn,yn}}] draws a multi-segment line by adding each
  of the points in the list and then adding segments that join the listed points.

EIDrawRectangle[x1,y1,x2,y2] adds nodes at {x1,y1}, {x1,y2}, {x2,
  y2} and  {x2,y1} and joins them with new segments.  Equivalent forms are:
 EIDrawRectangle[{x1,y1},{x2,y2}]
 EIDrawRectangle[{{x1,y1},{x2,y2}}]

EICreateRadius[x,y,z] turns a corner located at {x,y} into a curve of radius r.
 An equivalent form is: EICreateRadius[{x,y},z]
```

■ Object Selection and Manipulation

```
EISelectArcSegment[x,y] selects the arc segment closest to {x,y}.  An equivalent form is:
 EISelectArcSegment[{x,y}]

EISelectGroup[n] selects the nth group of nodes, segments, arc segments and block labels. This
  function will clear all previously selected elements and leave the edit mode in 4 (group)

EISelectLabel[x,y] selects the block label closest to {x,y}.  An equivalent form is:
 EISelectLabel[{x,y}]

EISelectNode[x,y] selects the node closest to {x,y}.  An equivalent form is:
 EISelectNode[{x,y}]

EISelectSegment[x,y] selects the segment closest to {x,y}.  An equivalent form is:
 EISelectSegment[{x,y}]
```

```
EISetArcSegmentProp[maxsegdeg, "propname", hide, groupno,"inconductor"]
  sets the properties associated with the selected arc segments
 maxsegdeg specifies that the arcs must be meshed with elements
  that span at most maxsegdeg degrees per element
 "propname" specifies the boundary property to be associated with the selected arcs
 hide: 0 = not hidden in post-processor, 1 == hidden in post processor
 groupno is an integer specifying the
  group number of which the selected arcs are to be members.
 "inconductor" specifies the name of the conductor property with which
  the selected arcs are to be associated. If the arcs is not to be
  part of a conductor, this parameter can be specified as "<None>".
```

```
EISetBlockProp["blockname",automesh,meshsize,
  groupno] sets 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
. groupno: make selected members of specified group number
```

```
EISetNodeProp["propname",groupno,"inconductor"] sets the selected
  nodes to have the nodal property "propname" and group number groupno.
  The "inconductor" string specifies which conductor the node belongs to. If the
  node doesn't belong to a named conductor, this parameter can be set to "<None>".
```

```
EISetSegmentProp["propname",elementsize,automesh,
  hide,groupno,"inconductor"] sets the select 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 groupno
 A member of the conductor specified by the string "inconductor". If the
  segment is not part of a conductor, this parameter can be specified as "<None>".
```

```
EIDeleteSelected[] deletes all selected objects.
```

```
EIDeleteSelectedArcSegments[] deletes all selected arc segments.
```

```
EIDeleteSelectedLabels[] deletes all selected block labels.
```

```
EIDeleteSelectedNodes[] deletes all selected nodes.
```

```
EIDeleteSelectedSegments[] deletes all selected segments.
```

```
EIClearSelected[] clear all selected nodes, blocks, segments and arc segments.
```

```
EIDefineOuterSpace[Zo,Ro,Ri] defines an axisymmetric external region to be used in
  conjuction with the Kelvin Transformation method of modeling unbounded problems.  The
  Zo parameter is the z-location of the origin of the outer region, the Ro parameter
  is the radius of the outer region, and the Ri parameter is the radius of the inner
  region (i.e. the region of interest). In the exterior region, the permeability
  varies as a function of distance from the origin of the external region.  These
  parameters are necessary to define the permeability variation in the external region.
```

```
EIAttachOuterSpace[] marks all selected block labels as members of the external region
  used for modeling unbounded axisymmetric problems via the Kelvin Transformation.
```

```
EIDetachOuterSpace[] undefines all selected block labels as members of the external region
  used for modeling unbounded axisymmetric problems via the Kelvin Transformation.
```

■ Move, Copy, Scale

```
EICopyRotate[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. This angle is measured in degrees.
 copies number of copies to be produced from the selected objects
 editaction 0-nodes, 1-segments, 2-block labels, 3-arcs, 4-group
 An equivalent form is:
 EICopyRotate[{bx,by},angle,copies,(editaction)]
```

```
EICopyTranslate[dx,dy,copies,(editaction)]
 {dx,dy} represents the distance by
  which the selected objects are to be incrementally shifted.
 copies specifies the number of copoies to be produced from the selected objects
 editaction 0-nodes, 1-segments, 2-block labels, 3-arcs, 4-group
 An equivalent form is:
 EICopyTranslate[{dx,dy},copies,(editaction)]
```

```
EIMirror[x1,y1,x2,y2,(editaction)] mirrors 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. Equivalent forms are:
 EIMirror[{x1,y1},{x2,y2},(editaction)]
 EIMirror[{{x1,y1},{x2,y2}},(editaction)]
```

```
EIMoveRotate[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
 An equivalent form is:
 EIMoveRotate[{bx,by},shiftangle,(editaction)]
```

```
EIMoveTranslate[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
 An equivalent form is:
 EIMoveTranslate[{dx,dy},(editaction)]
```

```
EIScale[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
 An equivalent for is: EIScale[{bx,by},scalefactor,(editaction)]
```

■ View Manipulation

```
EISetGrid[density,"type"] changes 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.
```

```
EIShowGrid[] displays the grid points
```

```
EIHideGrid[] hides the electrostatics input grid points
```

```
EIShowMesh[] displays the mesh
```

```
EIPurgeMesh[] clears the mesh out of both the screen and memory.
```

EIShowNames[] displays the material names associated with each block label

EIHideNames[] stops the names of the materials associated with each block label from being displayed

EISnapGridOn[] turns on snap-to-grid

EISnapGridOff[] turns off snap-to-grid

EIZoom[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}. Equivalent forms are:
EIZoom[{x1,y1},{x2,y2}]
EIZoom[{{x1,y1},{x2,y2}}]

EIZoomIn[] zooms out by a factor of 200%.

EIZoomNatural[] zooms to a "natural" view with sensible extents.

EIZoomOut[] zooms out by a factor of 50%.

EIGetView[] grabs the current electrostatics input view and returns a bitmapped graphics object. This object can subsequently be displayed using the Show[] command

■ Problem Commands

EIAnalyze[(flag)] runs the electrostatics solver. The flag parameter controls whether the solver 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. An equivalent form is:
EIAnalyse[(flag)]

EIClose[] closes the preprocessor window and destroys the current document.

EICreateMesh[] runs triangle to create a mesh. Note that this is not a necessary precursor of performing an analysis, as EIAnalyze[] will make sure the mesh is up to date before running an analysis.

EILoadSolution[] loads and displays the solution corresponding to the current geometry.

EIProbDef[units,type,precision,depth,minangle] changes the problem definition. 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 problemtype 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, specifying 1.E-8 requires the RMS of the residual to be less than 10^(-8). The depth parameter, represents the depth of the problem in the into-the-page direction for 2-D planar problems. The minangle parameter is a constraint for the mesh generator. It specifies the smallest permissible angle in triangles that compose the finite element mesh. A good value to choose is 30 degrees, but smaller values may be needed for "tough" geometries that contain small angles.

EIReadDXF["filename"] reads in geometry information a DXF file specified by "filename"

EIRefreshView[] Redraws the current view.

EISaveAs["filename"] saves the file with name "filename". Note if you use a path you must use two backslashes e.g. "c:\\temp\\myfemmfile.fem"

EISaveBitmap["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 EISaveAs command.

```
EISaveMetafile["filename"] saves an extended metafile screenshot
  of the current view to the file specified by "filename", subject to the
  printf-type formatting explained previously for the EISaveAs command.
```

```
EISetEditMode["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.
```

```
EISetFocus["documentname"] switches the electrostatics input file upon which
  scripting commands are to act. If more than one electrostatics input file is
  being edited at a time, this command can be used to switch between files so that
  the mutiple files can be operated upon programmatically . "documentname" should
  contain the name of the desired document as it appears on the window's title bar.
```

Heat Flow Input Commands

Define, Modify, or Delete Properties

```
HIAddBoundProp["boundpropname", BdryFormat, Tset, qs,
  Tinf, h, beta] adds a new boundary property with name "boundpropname"
 For a "Fixed Temperature" type boundary condition, set the Tset
  parameter to the desired temperature and all other parameters to zero.
 To obtain a "Heat Flux" type boundary condition, set qs to be the heat
  flux density and BdryFormat to 1. Set all other parameters to zero.
 To obtain a convection boundary condition, set h to the desired heat transfer
  coefficient and Tinf to the desired external temperature and set BdryFormat to 2.
 For a Radiation boundary condition, set beta equal to the desired emmissivity
  and Tinf to the  desired external temperature and set BdryFormat to 3.
 For a "Periodic" boundary condition, set BdryFormat to 4 and set all other parameters to zero.
 For an "Anti-Perodic" boundary condition, set BdryFormat to 5 set all other parameters to zero.
```

```
HIAddConductorProp["conductorname",Tc,qc,conductortype] adds a new conductor
  property with name "conductorname" with either a prescribed temperature or a
  prescribed total heat flux. Set the unused property to zero. The conductortype
  parameter is 0 for prescribed charge and 1 for prescribed temperature.
```

```
HIAddMaterial["materialname",Kx,Ky,qv] adds a new
  material with called "materialname" with the material properties:
 Kx Thermal conductivity in the x- or r-direction.
 Ky Thermal conductivity in the y- or z-direction.
 qv Volume heat generation density in units of J/m^3
```

```
HIAddPointProp["pointpropname",Tp,qp] adds a new point property of name "pointpropname" with
  either a specified temperature Tp or point heat generation qp in units of J/m.
```

HIModifyBoundProp["BdryName",propnum,value] 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:
0 – "BdryName" – Name of boundary property
1 – BdryFormat – Type of boundary condition (0 = Prescribed Temperature; 1 = Heat Flux; 2 = Convection; 3 = Radiation; 4 = Periodic; 5 = Antiperiodic) 2 – Tset – Fixed Temperature
3 – qs – Prescribed heat flux density
4 – Tinf – External temperature
5 – h – heat transfer coefficient
6 – beta – emmissivity

HIModifyConductorProp["ConductorName",propnum,value] allows for modification of a conductor property. The conductor property to be modified is specified by "ConductorName". 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:
0 – "ConductorName" – Name of the conductor property
1 – Tc – Conductor Temperature
2 – qc – Total conductor heat flux
3 – ConductorType – 0 = Prescribed heat flow, 1 = Prescribed temperature

HIModifyMaterial["BlockName",propnum,value] 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:
0 – "BlockName" – Name of the material 1 – Kx – x (or r) direction thermal conductivity
2 – Ky – y (or z) direction thermal conductivity 3 – qv – Volume heat generation

HIModifyPointProp["PointName",propnum,value] 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:
0 – PointName – Name of the point property
1 – Tp – Prescribed nodal temperature
2 – qp – Point heat generation in C/m

HIDeleteBoundProp["boundpropname"] deletes the boundary property named "boundpropname".

HIDeleteConductor["conductorname"] deletes the conductor property named "conductorname".

HIDeleteMaterial["materialname"] deletes the material property named "materialname".

HIDeletePointProp["pointpropname"] deletes the point property named "pointpropname".

■ Object Drawing Commands

HIAddArc[x1,y1,x2,y2,angle,maxseg] adds an arc to the electrostatics input geometry from the point nearst to {x1,y1} to the point nearest to {x2,y2}. The arc spans a number of degrees specified by angle. Since FEMM approximates arcs by many line segments, the parameter maxseg specifies the maximum number of degrees that is allowed to be spanned by any one segment. Equivalent forms are:
HIAddArc[{x1,y1},{x2,y2},angle,maxseg]
HIAddArc[{{x1,y1},{x2,y2}},angle,maxseg]

HIAddBlockLabel[x,y] adds a block label at the point {x,y}. An equivalent form is:
HIAddBlockLabel[{x,y}]

HIAddNode[x,y] adds a new node at {x,y}. An equivalent form is:
HIAddNode[{x,y}]

```
HIAddSegment[x1,y1,x2,y2] add a new line segment from node
  closest to {x1,y1} to node closest to {x2,y2}.  Equivalent forms are:
 HIAddSegment[{x1,y1},{x2,y2}]
 HIAddSegment[{{x1,y1},{x2,y2}}]
```

```
HIDrawArc[x1,y1,x2,y2,angle,maxseg] adds an arc to the heat flow input geometry by
  drawing points at {x1,y1} and {x2,y2} and then connecting them with an arc segment.
  The arc spans a number of degrees specified by angle. Since FEMM approximates
  arcs by many line segments, the parameter maxseg specifies the maximum number of
  degrees that is allowed to be spanned by any one segment. Equivalent forms are:
 HIDrawArc[{x1,y1},{x2,y2},angle,maxseg]
 HIDrawArc[{{x1,y1},{x2,y2}},angle,maxseg]
```

```
HIDrawLine[x1,y1,x2,y2] adds points at {x1,y1} and {x2,y2} and
  then adds a segment connecting these two points.  Equivalent forms are:
 HIDrawLine[{x1,y1},{x2,y2}]
 HIDrawLine[{{x1,y1},{x2,y2}}]
```

```
HIDrawPolygon[{{x1,y2},...,{xn,yn}}] adds new node points at
  every listed point and then draws a closed figure that connects the points
```

```
HIDrawPolyLine[{{x1,y2},...,{xn,yn}}] draws a multi-segment line by adding each
  of the points in the list and then adding segments that join the listed points.
```

```
HIDrawRectangle[x1,y1,x2,y2] adds nodes at {x1,y1}, {x1,y2}, {x2,
  y2} and  {x2,y1} and joins them with new segments.  Equivalent forms are:
 HIDrawRectangle[{x1,y1},{x2,y2}]
 HIDrawRectangle[{{x1,y1},{x2,y2}}]
```

```
EICreateRadius[x,y,z] turns a corner located at {x,y} into a curve of radius r.
 An equivalent form is: EICreateRadius[{x,y},z]
```

■ Object Selection and Manipulation

```
HISelectArcSegment[x,y] selects the arc segment closest to {x,y}.  An equivalent form is:
 HISelectArcSegment[{x,y}]
```

```
HISelectGroup[n] selects the nth group of nodes, segments, arc segments and block labels. This
  function will clear all previously selected elements and leave the edit mode in 4 (group)
```

```
HISelectLabel[x,y] selects the block label closest to {x,y}.  An equivalent form is:
 HISelectLabel[{x,y}]
```

```
HISelectNode[x,y] selects the node closest to {x,y}.  An equivalent form is:
 HISelectNode[{x,y}]
```

```
HISelectSegment[x,y] selects the segment closest to {x,y}.  An equivalent form is:
 HISelectSegment[{x,y}]
```

```
HISetArcSegmentProp[maxsegdeg, "propname", hide, groupno,"inconductor"]
  sets the properties associated with the selected arc segments
 maxsegdeg specifies that the arcs must be meshed with elements
  that span at most maxsegdeg degrees per element
 "propname" specifies the boundary property to be associated with the selected arcs
 hide: 0 = not hidden in post-processor, 1 == hidden in post processor
 groupno is an integer specifying the
  group number of which the selected arcs are to be members.
 "inconductor" specifies the name of the conductor property with which
  the selected arcs are to be associated. If the arcs is not to be
  part of a conductor, this parameter can be specified as "<None>".
```

```
HISetBlockProp["blockname",automesh,meshsize,
  groupno] sets 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
. groupno: make selected members of specified group number
```

```
HISetNodeProp["propname",groupno,"inconductor"] sets the selected
  nodes to have the nodal property "propname" and group number groupno.
  The "inconductor" string specifies which conductor the node belongs to. If the
  node doesn't belong to a named conductor, this parameter can be set to "<None>".
```

```
HISetSegmentProp["propname",elementsize,automesh,
  hide,groupno,"inconductor"] sets the select 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 groupno
 A member of the conductor specified by the string "inconductor". If the
  segment is not part of a conductor, this parameter can be specified as "<None>".
```

```
HIDeleteSelected[] deletes all selected objects.
```

```
HIDeleteSelectedArcSegments[] deletes all selected arc segments.
```

```
HIDeleteSelectedLabels[] deletes all selected block labels.
```

```
HIDeleteSelectedNodes[] deletes all selected nodes.
```

```
HIDeleteSelectedSegments[] deletes all selected segments.
```

```
HIClearSelected[] clear all selected nodes, blocks, segments and arc segments.
```

```
HIDefineOuterSpace[Zo,Ro,Ri] defines an axisymmetric external region to be used in
  conjuction with the Kelvin Transformation method of modeling unbounded problems.  The
  Zo parameter is the z-location of the origin of the outer region, the Ro parameter
  is the radius of the outer region, and the Ri parameter is the radius of the inner
  region (i.e. the region of interest). In the exterior region, the permeability
  varies as a function of distance from the origin of the external region.  These
  parameters are necessary to define the permeability variation in the external region.
```

```
HIAttachOuterSpace[] marks all selected block labels as members of the external region
  used for modeling unbounded axisymmetric problems via the Kelvin Transformation.
```

```
HIDetachOuterSpace[] undefines all selected block labels as members of the external region
  used for modeling unbounded axisymmetric problems via the Kelvin Transformation.
```

■ Move, Copy, Scale

```
HICopyRotate[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. This angle is measured in degrees.
 copies number of copies to be produced from the selected objects
 editaction 0-nodes, 1-segments, 2-block labels, 3-arcs, 4-group
 An equivalent form is:
 HICopyRotate[{bx,by},angle,copies,(editaction)]
```

```
HICopyTranslate[dx,dy,copies,(editaction)]
 {dx,dy} represents the distance by
  which the selected objects are to be incrementally shifted.
 copies specifies the number of copoies to be produced from the selected objects
 editaction 0-nodes, 1-segments, 2-block labels, 3-arcs, 4-group
 An equivalent form is:
 HICopyTranslate[{dx,dy},copies,(editaction)]
```

```
HIMirror[x1,y1,x2,y2,(editaction)] mirrors 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. Equivalent forms are:
 HIMirror[{x1,y1},{x2,y2},(editaction)]
 HIMirror[{{x1,y1},{x2,y2}},(editaction)]
```

```
HIMoveRotate[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
 An equivalent form is:
 HIMoveRotate[{bx,by},shiftangle,(editaction)]
```

```
HIMoveTranslate[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
 An equivalent form is:
 HIMoveTranslate[{dx,dy},(editaction)]
```

```
HIScale[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
 An equivalent for is: HIScale[{bx,by},scalefactor,(editaction)]
```

■ View Manipulation

```
HISetGrid[density,"type"] changes 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.
```

```
HIShowGrid[] displays the grid points
```

```
HIHideGrid[] hides the heat flow input grid points
```

```
HIShowMesh[] displays the mesh
```

```
HIPurgeMesh[] clears the mesh out of both the screen and memory.
```

```
HIShowNames[] displays the material names associated with each block label
```

```
HIHideNames[] stops the names of the
  materials associated with each block label from being displayed
```

```
HISnapGridOn[] turns on snap-to-grid
```

```
HISnapGridOff[] turns off snap-to-grid
```

```
HIZoom[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}. Equivalent forms are:
 HIZoom[{x1,y1},{x2,y2}]
 HIZoom[{{x1,y1},{x2,y2}}]
```

HIZoomIn[] zooms out by a factor of 200%.

HIZoomNatural[] zooms to a "natural" view with sensible extents.

HIZoomOut[] zooms out by a factor of 50%.

HIGetView[] grabs the current heat flow input view and returns a bitmapped graphics object. This object can subsequently be displayed using the Show[] command

■ Problem Commands

HIAnalyze[(flag)] runs the heat flow solver. The flag parameter controls whether the solver 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. An equivalent form is: HIAnalyse[(flag)]

HIClose[] closes the preprocessor window and destroys the current document.

HICreateMesh[] runs triangle to create a mesh. Note that this is not a necessary precursor of performing an analysis, as HIAnalyze[] will make sure the mesh is up to date before running an analysis.

HILoadSolution[] loads and displays the solution corresponding to the current geometry.

HIProbDef[units,type,precision,depth,minangle] changes the problem definition. 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 problemtype 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, specifying 1.E-8 requires the RMS of the residual to be less than 10^(-8). The depth parameter, represents the depth of the problem in the into-the-page direction for 2-D planar problems. The minangle parameter is a constraint for the mesh generator. It specifies the smallest permissible angle in triangles that compose the finite element mesh. A good value to choose is 30 degrees, but smaller values may be needed for "tough" geometries that contain small angles.

HIReadDXF["filename"] reads in geometry information a DXF file specified by "filename"

HIRefreshView[] Redraws the current view.

HISaveAs["filename"] saves the file with name "filename". Note if you use a path you must use two backslashes e.g. "c:\\temp\\myfemmfile.fem"

HISaveBitmap["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 HISaveAs command.

HISaveMetafile["filename"] saves an extended metafile screenshot of the current view to the file specified by "filename", subject to the printf-type formatting explained previously for the HISaveAs command.

HISetEditMode["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.

```
HISetFocus["documentname"] switches the heat flow input file upon which
  scripting commands are to act. If more than one heat flow input file is being
  edited at a time, this command can be used to switch between files so that the
  mutiple files can be operated upon programmatically . "documentname" should
  contain the name of the desired document as it appears on the window's title bar.
```

■ Electrostatics Input Commands

■ Define, Modify, or Delete Properties

```
CIAddBoundProp["boundpropname", Vs, qs, c0, c1,
  bdryformat] adds a new boundary property with name "boundpropname"
 For a "Fixed Voltage" type boundary condition, set the Vs parameter
  to the desired voltage and all other parameters to zero.
 To obtain a "Mixed"  type boundary condition, set     exttt{C1} and c0
  as required and bdryformat to 1. Set all other parameters to zero.
 To obtain a prescribes surface current density, set qs to the
  desired current density in A/m^2 and set bdryformat to 2.
 For a "Periodic" boundary condition, set bdryformat to 3 and set all other parameters to zero.
 For an "Anti-Perodic" boundary condition, set bdryformat to 4 set all other parameters to zero.
```

```
CIAddConductorProp["conductorname",vc,qc,conductortype] adds a
  new conductor property with name "conductorname" with either a prescribed
  voltage or a prescribed total current. Set the unused property to zero. The
  conductortype parameter is 0 for prescribed charge and 1 for prescribed voltage.
```

```
CIAddMaterial["materialname",ox,oy,ex,ey,ltx,lty] adds a
  new material with called "materialname" with the material properties:
 ox Electrical conductivity in the x- or r-direction.
 oy Electrical conductivity in the y- or z-direction.
 ex Relative electrical permittivity in the x- or r-direction.
 ey Relative electrical permittivity in the y- or z-direction.
 ltx Dielectric loss tangent in the x- or r-direction.
 lty Dielectric loss tangent in the y- or z-direction.
```

```
CIAddPointProp["pointpropname",vp,qp] adds a new point property of name "pointpropname" with
  either a specified voltage vp or point heat generation qp in units of J/m.
```

```
CIModifyBoundProp["BdryName",propnum,value] 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:
 0 - "BdryName" - Name of boundary property
 1 - Vs - Fixed Voltage
 2 - js - Prescribed current density
 3 - c0 - External voltage
 4 - c1 - heat transfer coefficient
 5 - BdryFormat -  Type of boundary condition (0 = Prescribed
  Voltage; 1 = Mixed; 2 = Surface Current Density; 3 = Periodic; 4 = Antiperiodic)
```

CIModifyConductorProp["ConductorName",propnum,value] allows for modification of a conductor property. The conductor property to be modified is specified by "ConductorName". 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:
0 – "ConductorName" – Name of the conductor property
1 – Vc – Conductor Voltage
2 – jc – Total conductor current
3 – ConductorType – 0 = Prescribed current, 1 = Prescribed voltage

CIModifyMaterial["BlockName",propnum,value] 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:
0 – "BlockName" – Name of the material
1 – ox – x (or r) direction electrical conductivity
2 – oy – y (or z) direction electrical conductivity
1 – ox – x (or r) direction relative electrical permittivity
2 – oy – y (or z) direction relative electrical permittivity
1 – ox – x (or r) direction dielectric loss tangent
2 – oy – y (or z) direction dielectric loss tangent

CIModifyPointProp["PointName",propnum,value] 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:
0 – PointName – Name of the point property
1 – Vp – Prescribed nodal voltage
2 – jp – Point current in A/m

CIDeleteBoundProp["boundpropname"] deletes the boundary property named "boundpropname".

CIDeleteConductor["conductorname"] deletes the conductor property named "conductorname".

CIDeleteMaterial["materialname"] deletes the material property named "materialname".

CIDeletePointProp["pointpropname"] deletes the point property named "pointpropname".

■ Object Drawing Commands

CIAddArc[x1,y1,x2,y2,angle,maxseg] adds an arc to the electrostatics input geometry from the point nearst to {x1,y1} to the point nearest to {x2,y2}. The arc spans a number of degrees specified by angle. Since FEMM approximates arcs by many line segments, the parameter maxseg specifies the maximum number of degrees that is allowed to be spanned by any one segment. Equivalent forms are:
CIAddArc[{x1,y1},{x2,y2},angle,maxseg]
CIAddArc[{{x1,y1},{x2,y2}},angle,maxseg]

CIAddBlockLabel[x,y] adds a block label at the point {x,y}. An equivalent form is:
CIAddBlockLabel[{x,y}]

CIAddNode[x,y] adds a new node at {x,y}. An equivalent form is:
CIAddNode[{x,y}]

CIAddSegment[x1,y1,x2,y2] add a new line segment from node closest to {x1,y1} to node closest to {x2,y2}. Equivalent forms are:
CIAddSegment[{x1,y1},{x2,y2}]
CIAddSegment[{{x1,y1},{x2,y2}}]

```
CIDrawArc[x1,y1,x2,y2,angle,maxseg] adds an arc to the current input geometry by
  drawing points at {x1,y1} and {x2,y2} and then connecting them with an arc segment.
  The arc spans a number of degrees specified by angle. Since FEMM approximates
  arcs by many line segments, the parameter maxseg specifies the maximum number of
  degrees that is allowed to be spanned by any one segment. Equivalent forms are:
 CIDrawArc[{x1,y1},{x2,y2},angle,maxseg]
 CIDrawArc[{{x1,y1},{x2,y2}},angle,maxseg]
```

```
CIDrawLine[x1,y1,x2,y2] adds points at {x1,y1} and {x2,y2} and
  then adds a segment connecting these two points.  Equivalent forms are:
 CIDrawLine[{x1,y1},{x2,y2}]
 CIDrawLine[{{x1,y1},{x2,y2}}]
```

```
CIDrawPolygon[{{x1,y2},...,{xn,yn}}] adds new node points at
  every listed point and then draws a closed figure that connects the points
```

```
CIDrawPolyLine[{{x1,y2},...,{xn,yn}}] draws a multi-segment line by adding each
  of the points in the list and then adding segments that join the listed points.
```

```
CIDrawRectangle[x1,y1,x2,y2] adds nodes at {x1,y1}, {x1,y2}, {x2,
  y2} and  {x2,y1} and joins them with new segments.  Equivalent forms are:
 CIDrawRectangle[{x1,y1},{x2,y2}]
 CIDrawRectangle[{{x1,y1},{x2,y2}}]
```

```
CICreateRadius[x,y,z] turns a corner located at {x,y} into a curve of radius r.
 An equivalent form is: CICreateRadius[{x,y},z]
```

■ Object Selection and Manipulation

```
CISelectArcSegment[x,y] selects the arc segment closest to {x,y}.  An equivalent form is:
 CISelectArcSegment[{x,y}]
```

```
CISelectGroup[n] selects the nth group of nodes, segments, arc segments and block labels. This
  function will clear all previously selected elements and leave the edit mode in 4 (group)
```

```
CISelectLabel[x,y] selects the block label closest to {x,y}.  An equivalent form is:
 CISelectLabel[{x,y}]
```

```
CISelectNode[x,y] selects the node closest to {x,y}.  An equivalent form is:
 CISelectNode[{x,y}]
```

```
CISelectSegment[x,y] selects the segment closest to {x,y}.  An equivalent form is:
 CISelectSegment[{x,y}]
```

```
CISetArcSegmentProp[maxsegdeg, "propname", hide, groupno,"inconductor"]
  sets the properties associated with the selected arc segments
 maxsegdeg specifies that the arcs must be meshed with elements
  that span at most maxsegdeg degrees per element
 "propname" specifies the boundary property to be associated with the selected arcs
 hide: 0 = not hidden in post-processor, 1 == hidden in post processor
 groupno is an integer specifying the
  group number of which the selected arcs are to be members.
 "inconductor" specifies the name of the conductor property with which
  the selected arcs are to be associated. If the arcs is not to be
  part of a conductor, this parameter can be specified as "<None>".
```

CISetBlockProp["blockname",automesh,meshsize,
  groupno] sets 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
. groupno: make selected members of specified group number

CISetNodeProp["propname",groupno,"inconductor"] sets the selected
  nodes to have the nodal property "propname" and group number groupno.
  The "inconductor" string specifies which conductor the node belongs to. If the
  node doesn't belong to a named conductor, this parameter can be set to "<None>".

CISetSegmentProp["propname",elementsize,automesh,
  hide,groupno,"inconductor"] sets the select 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 groupno
 A member of the conductor specified by the string "inconductor". If the
  segment is not part of a conductor, this parameter can be specified as "<None>".

CIDeleteSelected[] deletes all selected objects.

CIDeleteSelectedArcSegments[] deletes all selected arc segments.

CIDeleteSelectedLabels[] deletes all selected block labels.

CIDeleteSelectedNodes[] deletes all selected nodes.

CIDeleteSelectedSegments[] deletes all selected segments.

CIClearSelected[] clear all selected nodes, blocks, segments and arc segments.

CIDefineOuterSpace[Zo,Ro,Ri] defines an axisymmetric external region to be used in
  conjuction with the Kelvin Transformation method of modeling unbounded problems. The
  Zo parameter is the z-location of the origin of the outer region, the Ro parameter
  is the radius of the outer region, and the Ri parameter is the radius of the inner
  region (i.e. the region of interest). In the exterior region, the permeability
  varies as a function of distance from the origin of the external region. These
  parameters are necessary to define the permeability variation in the external region.

CIAttachOuterSpace[] marks all selected block labels as members of the external region
  used for modeling unbounded axisymmetric problems via the Kelvin Transformation.

CIDetachOuterSpace[] undefines all selected block labels as members of the external region
  used for modeling unbounded axisymmetric problems via the Kelvin Transformation.

■ Move, Copy, Scale

CICopyRotate[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. This angle is measured in degrees.
 copies number of copies to be produced from the selected objects
 editaction 0-nodes, 1-segments, 2-block labels, 3-arcs, 4-group
 An equivalent form is:
 CICopyRotate[{bx,by},angle,copies,(editaction)]

CICopyTranslate[dx,dy,copies,(editaction)]
{dx,dy} represents the distance by
which the selected objects are to be incrementally shifted.
copies specifies the number of copoies to be produced from the selected objects
editaction 0-nodes, 1-segments, 2-block labels, 3-arcs, 4-group
An equivalent form is:
CICopyTranslate[{dx,dy},copies,(editaction)]

CIMirror[x1,y1,x2,y2,(editaction)] mirrors 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. Equivalent forms are:
CIMirror[{x1,y1},{x2,y2},(editaction)]
CIMirror[{{x1,y1},{x2,y2}},(editaction)]

CIMoveRotate[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
An equivalent form is:
CIMoveRotate[{bx,by},shiftangle,(editaction)]

CIMoveTranslate[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
An equivalent form is:
CIMoveTranslate[{dx,dy},(editaction)]

CIScale[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
An equivalent for is: CIScale[{bx,by},scalefactor,(editaction)]

■ View Manipulation

CISetGrid[density,"type"] changes 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.

CIShowGrid[] displays the grid points

CIHideGrid[] hides the current input grid points

CIShowMesh[] displays the mesh

CIPurgeMesh[] clears the mesh out of both the screen and memory.

CIShowNames[] displays the material names associated with each block label

CIHideNames[] stops the names of the materials associated with each block label from being displayed

CISnapGridOn[] turns on snap-to-grid

CISnapGridOff[] turns off snap-to-grid

CIZoom[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}. Equivalent forms are:
CIZoom[{x1,y1},{x2,y2}]
CIZoom[{{x1,y1},{x2,y2}}]

CIZoomIn[] zooms out by a factor of 200%.

CIZoomNatural[] zooms to a "natural" view with sensible extents.

CIZoomOut[] zooms out by a factor of 50%.

CIGetView[] grabs the current current input view and returns a bitmapped graphics object. This object can subsequently be displayed using the Show[] command

■ Problem Commands

CIAnalyze[(flag)] runs the current solver. The flag parameter controls whether the solver 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. An equivalent form is: CIAnalyse[(flag)]

CIClose[] closes the preprocessor window and destroys the current document.

CICreateMesh[] runs triangle to create a mesh. Note that this is not a necessary precursor of performing an analysis, as CIAnalyze[] will make sure the mesh is up to date before running an analysis.

CILoadSolution[] loads and displays the solution corresponding to the current geometry.

CIProbDef[units,type,frequency,precision,depth,minangle] changes the problem definition. 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 problemtype to "planar" for a 2-D planar problem, or to "axi" for an axisymmetric problem. The frequency parameter denotes the frequency of the analysis in Hz. The precision parameter dictates the precision required by the solver. For example, specifying 1.E-8 requires the RMS of the residual to be less than 10^(-8). The depth parameter, represents the depth of the problem in the into-the-page direction for 2-D planar problems. The minangle parameter is a constraint for the mesh generator. It specifies the smallest permissible angle in triangles that compose the finite element mesh. A good value to choose is 30 degrees, but smaller values may be needed for "tough" geometries that contain small angles.

CIReadDXF["filename"] reads in geometry information a DXF file specified by "filename"

CIRefreshView[] Redraws the current view.

CISaveAs["filename"] saves the file with name "filename". Note if you use a path you must use two backslashes e.g. "c:\\temp\\myfemmfile.fem"

CISaveBitmap["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 CISaveAs command.

CISaveMetafile["filename"] saves an extended metafile screenshot of the current view to the file specified by "filename", subject to the printf-type formatting explained previously for the CISaveAs command.

CISetEditMode["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.

CISetFocus["documentname"] switches the current input file upon which scripting commands are to act. If more than one current input file is being edited at a time, this command can be used to switch between files so that the mutiple files can be operated upon programmatically . "documentname" should contain the name of the desired document as it appears on the window's title bar.

■ Magnetics Input Commands

■ Define, Modify, or Delete Properties

MIAddBoundProp["propname",A0,A1,A2,Phi,Mu,Sig,c0, c1,BdryFormat] adds a new boundary property with name "propname"
- 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-Perodic" boundary condition, set BdryFormat to 5 set all other parameters to zero.

MIAddCircProp["circuitname",i,circuittype] adds a new circuit property with name "circuitname" with a prescribed current i. It is OK for the current to be complex-valued. The circuittype parameter is 0 for a parallel-connected circuit and 1 for a series-connected circuit.

MIAddMaterial["materialname",mux,muy,Hc,J, Cduct,Lamd,Phihmax,lamfill,LamType,Phihx,Phihy,NStrands,WireD] adds a new material with called "materialname" with the material properties:
- mux - Relative permeability in the x- or r-direction (for linear materials)
- muy - Relative permeability in the y- or z-direction (for linear materials)
- Hc - Permanent magnet coercivity in Amps/Meter.
- J - Applied source current density in Amps/mm2. It is OK for J to be complex-valued.
- Cduct - Electrical conductivity of the material in MS/m.
- Lamd - Lamination thickness in millimeters.
- Phihmax - Hysteresis lag angle in degrees, used for nonlinear BH curves.
- Lamfill - Fraction of the volume occupied per lamination that is actually filled with iron (Note that this parameter defaults to 1 the FEMM preprocessor dialog box because, by default, iron completely fills the volume)
- Lamtype - Set to 0 for "Not laminated or laminated in plane"; 1 for "Laminated x or r"; 2 for "Laminated y or z"
- Phihx - Hysteresis lag in degrees in the x-direction for linear problems.
- Phihy - Hysteresis lag in degrees in the y-direction for linear problems.
- NStrands - number of strands per wire if the material is a wire.
- WireD - diameter of each strand in mm if the material is a wire.

MIAddBHPoint["materialname",b,h] adds the point {b,h} to the BH curve for the material specified by "materialname". An equivalent form is: MIAddBHPoint["materialname",{b,h}]

```
MIAddBHPoints["materialname",list] adds all of the points
  in list to the BH curve for the material specified by "materialname"

MIClearBHPoint["materialname"] erases all of the BH
  points that have been defined for the material named "materialname"

MIAddPointProp["pointpropname",A,J] adds a new point property of name "pointpropname" with
  either a specified potential A in units of Webers/Meter or a point current J in units
  of Amps. It is OK for either A or J to be complex-valued. Set the unused parameter to 0.

MIModifyBoundProp["BdryName",propnum,value] 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:
 0 - "BdryName" - Name of boundary property
 1 A0 - Prescribed A parameter
 2 A1 - Prescribed A parameter
 3 A2 - Prescribed A parameter
 4 phi - Prescribed A phase
 5 mu -  Small skin depth relative permeability
 6 Cduct -  Small skin depth conductivity, MS/m
 7 c0 - Mixed BC parameter
 8 c1 - Mixed BC parameter
 9 - BdryFormat.  Valid choices for boundary format include:
     0 = Prescribed A
     1 = Small skin depth
     2 = Mixed
     3 = Strategic Dual Image
     4 = Periodic
     5 = Antiperiodic

MIModifyCircProp["circname",propnum,value] 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:
 0 - CircName - Name of the circuit property
 1 - i - Total current. It is OK for i to be complex-valued
 2 - CircType -  0 = Parallel, 1 = Series

MISetCurrent["circname",i] sets the current of the circuit named "circname" to
  the value specified by i.  It is OK for the current to be complex-valued.

MIModifyMaterial["BlockName",propnum,value] allows for modification of
  a material's properties without redefining the entire material. 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:
 0 - "BlockName" - Name of the material
 1 - mux - x (or r) direction relative permeability
 2 - muy - y (or z) direction relative permeability
 3 - Hc - Coercivity, Amps/Meter
 4 - J - Current density, MA/m2.  OK if J is complex-valued.
 5 - cduct - Electrical conductivity, MS/m
 6 - dlam - Lamination thickness, mm
 7 - phihmax - Hysteresis lag angle for nonlinear problems, degrees
 8 - LamFill - Iron fill fraction
 9 - LamType - 0 = None/In plane, 1 = parallel to x, 2=parallel to y
 10 - phihx - Hysteresis lag in x-direction for linear problems, degrees
 11 - phihy - Hysteresis lag in y-direction for linear problems, degrees
```

```
MIModifyPointProp["PointName",propnum,value] 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:
 0 - PointName - Name of the point property
 1 - A - Nodal potential, Weber/Meter
 2 - J - Nodal current, Amps
 It is OK if either A or J are complex-valued.
```

```
MIDeleteBoundProp["boundpropname"] deletes the boundary property named "boundpropname".
```

```
MIDeleteCircuit["circuitname"] deletes the circuit property named "circuitname".
```

```
MIDeleteMaterial["materialname"] deletes the material property named "materialname".
```

```
MIDeletePointProp["pointpropname"] deletes the point property named "pointpropname".
```

■ Object Drawing Commands

```
MIAddArc[x1,y1,x2,y2,angle,maxseg] adds an arc to the electrostatics input
  geometry from the point nearst to {x1,y1} to the point nearest to {x2,y2}.  The
  arc spans a number of degrees specified by angle. Since FEMM approximates arcs
  by many line segments, the parameter maxseg specifies the maximum number of
  degrees that is allowed to be spanned by any one segment. Equivalent forms are:
 MIAddArc[{x1,y1},{x2,y2},angle,maxseg]
 MIAddArc[{{x1,y1},{x2,y2}},angle,maxseg]
```

```
MIAddBlockLabel[x,y] adds a block label at the point {x,y}.  An equivalent form is:
 MIAddBlockLabel[{x,y}]
```

```
MIAddNode[x,y] adds a new node at {x,y}. An equivalent form is:
 MIAddNode[{x,y}]
```

```
MIAddSegment[x1,y1,x2,y2] add a new line segment from node
  closest to {x1,y1} to node closest to {x2,y2}.  Equivalent forms are:
 MIAddSegment[{x1,y1},{x2,y2}]
 MIAddSegment[{{x1,y1},{x2,y2}}]
```

```
MIDrawArc[x1,y1,x2,y2,angle,maxseg] adds an arc to the electrostatics input geometry by
  drawing points at {x1,y1} and {x2,y2} and then connecting them with an arc segment.
  The arc spans a number of degrees specified by angle. Since FEMM approximates
  arcs by many line segments, the parameter maxseg specifies the maximum number of
  degrees that is allowed to be spanned by any one segment. Equivalent forms are:
 MIDrawArc[{x1,y1},{x2,y2},angle,maxseg]
 MIDrawArc[{{x1,y1},{x2,y2}},angle,maxseg]
```

```
MIDrawLine[x1,y1,x2,y2] adds points at {x1,y1} and {x2,y2} and
  then adds a segment connecting these two points.  Equivalent forms are:
 MIDrawLine[{x1,y1},{x2,y2}]
 MIDrawLine[{{x1,y1},{x2,y2}}]
```

```
MIDrawPolygon[{{x1,y2},...,{xn,yn}}] adds new node points at
  every listed point and then draws a closed figure that connects the points
```

```
MIDrawPolyLine[{{x1,y2},...,{xn,yn}}] draws a multi-segment line by adding each
  of the points in the list and then adding segments that join the listed points.
```

```
MIDrawRectangle[x1,y1,x2,y2] adds nodes at {x1,y1}, {x1,y2}, {x2,
  y2} and  {x2,y1} and joins them with new segments.  Equivalent forms are:
 MIDrawRectangle[{x1,y1},{x2,y2}]
 MIDrawRectangle[{{x1,y1},{x2,y2}}]
```

MICreateRadius[x,y,z] turns a corner located at {x,y} into a curve of radius r.
An equivalent form is: MICreateRadius[{x,y},z]

■ Object Selection and Manipulation

MISelectArcSegment[x,y] selects the arc segment closest to {x,y}. An equivalent form is:
MISelectArcSegment[{x,y}]

MISelectGroup[n] selects the nth group of nodes, segments, arc segments and block labels. This function will clear all previously selected elements and leave the edit mode in 4 (group)

MISelectLabel[x,y] selects the block label closest to {x,y}. An equivalent form is:
MISelectLabel[{x,y}]

MISelectNode[x,y] selects the node closest to {x,y}. An equivalent form is:
MISelectNode[{x,y}]

MISelectSegment[x,y] selects the segment closest to {x,y}. An equivalent form is:
MISelectSegment[{x,y}]

MISetArcSegmentProp[maxsegdeg,"propname",hide,groupno] sets 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 groupno

MISetBlockProp["blockname",automesh,meshsize,"incircuit", magdirection,group,turns]
sets 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"
– The magnetization is directed along an
angle in measured in degrees denoted by the parameter magdirection
– A member of group number group
– The number of turns associated with this label is denoted by turns.

MISetNodeProp["propname",groupno] set the selected
nodes to have the nodal property "propname" and group number groupno.

MISetSegmentProp["propname",elementsize,automesh,hide,group] sets 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

MIDeleteSelected[] deletes all selected objects.

MIDeleteSelectedArcSegments[] deletes all selected arc segments.

MIDeleteSelectedLabels[] deletes all selected block labels.

MIDeleteSelectedNodes[] deletes all selected nodes.

MIDeleteSelectedSegments[] deletes all selected segments.

MIClearSelected[] clear all selected nodes, blocks, segments and arc segments.

MIDefineOuterSpace[Zo,Ro,Ri] defines an axisymmetric external region to be used in conjuction with the Kelvin Transformation method of modeling unbounded problems. The Zo parameter is the z-location of the origin of the outer region, the Ro parameter is the radius of the outer region, and the Ri parameter is the radius of the inner region (i.e. the region of interest). In the exterior region, the permeability varies as a function of distance from the origin of the external region. These parameters are necessary to define the permeability variation in the external region.

MIAttachOuterSpace[] marks all selected block labels as members of the external region used for modeling unbounded axisymmetric problems via the Kelvin Transformation.

MIDetachOuterSpace[] undefines all selected block labels as members of the external region used for modeling unbounded axisymmetric problems via the Kelvin Transformation.

■ Move/Copy/Scale

MICopyRotate[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. This angle is measured in degrees.
copies number of copies to be produced from the selected objects
editaction 0-nodes, 1-segments, 2-block labels, 3-arcs, 4-group
An equivalent form is:
MICopyRotate[{bx,by},angle,copies,(editaction)]

MICopyTranslate[dx,dy,copies,(editaction)]
{dx,dy} represents the distance by
which the selected objects are to be incrementally shifted.
copies specifies the number of copoies to be produced from the selected objects
editaction 0-nodes, 1-segments, 2-block labels, 3-arcs, 4-group
An equivalent form is:
MICopyTranslate[{dx,dy},copies,(editaction)]

MIMirror[x1,y1,x2,y2,(editaction)] mirrors 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. Equivalent forms are:
MIMirror[{x1,y1},{x2,y2},(editaction)]
MIMirror[{{x1,y1},{x2,y2}},(editaction)]

MIMoveRotate[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
An equivalent form is:
MIMoveRotate[{bx,by},shiftangle,(editaction)]

MIMoveTranslate[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
An equivalent form is:
MIMoveTranslate[{dx,dy},(editaction)]

MIScale[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
An equivalent for is: MIScale[{bx,by},scalefactor,(editaction)]

■ View Manipulation

```
MISetGrid[density,"type"] changes 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.
```

```
MIShowGrid[] displays the grid points
```

```
MIHideGrid[] hides the electrostatics input grid points
```

```
MIShowMesh[] displays the mesh
```

```
MIPurgeMesh[] clears the mesh out of both the screen and memory.
```

```
MIShowNames[] displays the material names associated with each block label
```

```
MIHideNames[] stops the names of the
  materials associated with each block label from being displayed
```

```
MISnapGridOn[] turns on snap-to-grid
```

```
MISnapGridOff[] turns off snap-to-grid
```

```
MIZoom[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}. Equivalent forms are:
 MIZoom[{x1,y1},{x2,y2}]
 MIZoom[{{x1,y1},{x2,y2}}]
```

```
MIZoomIn[] zooms out by a factor of 200%.
```

```
MIZoomNatural[] zooms to a "natural" view with sensible extents.
```

```
MIZoomOut[] zooms out by a factor of 50%.
```

```
MIGetView[] grabs the current electrostatics input view and returns a bitmapped
  graphics object.  This object can subsequently be displayed using the Show[] command
```

■ Problem Commands

```
MIAnalyze[(flag)] runs the magnetics solver. The flag parameter controls whether the solver
  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. An equivalent form is:
 MIAnalyse[(flag)]
```

```
MIClose[] closes the preprocessor window and destroys the current document.
```

```
MICreateMesh[] runs triangle to create a mesh. Note that this
  is not a necessary precursor of performing an analysis, as MIAnalyze[]
  will make sure the mesh is up to date before running an analysis.
```

```
MILoadSolution[] loads and displays the solution corresponding to the current geometry.
```

MIProbDef[freq,units,type,precision,depth,minangle] changes the problem definition. The freq parameter specifies the frequency at which the analysis is performed in Hz. 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 problemtype 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, specifying 1.E-8 requires the RMS of the residual to be less than 10^(-8). The depth parameter, represents the depth of the problem in the into-the-page direction for 2-D planar problems. The minangle parameter is a constraint for the mesh generator. It specifies the smallest permissible angle in triangles that compose the finite element mesh. A good value to choose is 30 degrees, but smaller values may be needed for "tough" geometries that contain small angles.

MIReadDXF["filename"] reads in geometry information a DXF file specified by "filename"

MIRefreshView[] Redraws the current view.

MISaveAs["filename"] saves the file with name "filename". Note if you use a path you must use two backslashes e.g. "c:\\temp\\myfemmfile.fem"

MISaveBitmap["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 MISaveAs command.

MISaveMetafile["filename"] saves an extended metafile screenshot of the current view to the file specified by "filename", subject to the printf-type formatting explained previously for the MISaveAs command.

MISetEditMode["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.

MISetFocus["documentname"] switches the electrostatics input file upon which scripting commands are to act. If more than one electrostatics input file is being edited at a time, this command can be used to switch between files so that the mutiple files can be operated upon programmatically . "documentname" should contain the name of the desired document as it appears on the window's title bar.

■ Electrostatics Output Commands

■ Contours, Regions, and Integrals

EOAddContour[x,y] adds the point {x,y} to the contour that is used for plotting values along lines and for computing line integrals. An equivalent form is:
EOAddContour[{x,y}]

EOBendContour[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.

```
EOClearContour[] clears the current contour selection

EOSelectBlock[x,y] select the block that contains point {x,y}.  An equivalent form is:
 EOSelectBlock[{x,y}]

EOSelectPoint[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. An equivalent form is:
 EOSelectPoint[{x,y}]

EOGroupSelectBlock[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. EOGroupSelectBlock[]), all blocks are selected.

EOClearBlock[] clears the current block selection

EOBlockIntegral[type] calculate a block integral for the
  selected blocks. The type parameter can take on the following values:
 0 - Stored Energy
 1 - Block Cross-section
 2 - Block Volume
 3 - Average D over the block
 4 - Average E over the block
 5 - Weighted Stress Tensor Force
 6 - Weighted Stress Tensor Torque
 returns either a single value or a list of two values, depending on the specified type

EOLineIntegral[type] calculate the line
  integral for the defined contour. Valid type selections include:
 0 - E ·t
 1 - D·n
 2 - Contour length
 3 - Force from stress tensor
 4 - Torque from stress tensor
 This integral returns either one value
  or a list of two values, depending on the type of integral

EOMakePlot[PlotType,NumPoints,Filename,FileFormat]
 This function allows programmatic access to FEMM's 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 specfied file name,
  rather than display it to make a graphical plot. Valid entries for PlotType are:
 0 - V (Voltage)
 1 - |D| (Magnitude of flux density)
 2 - D . n (Normal flux density)
 3 - D . t (Tangential flux density)
 4 - |E| (Magnitude of field intensity)
 5 - E . n (Normal field intensity)
 6 - E . t (Tangential field intensity)

EOGetConductorProperties["conductor"] returns properties for the conductor
  property named "conductor". A list with two values is returned: The voltage
  of the specified conductor, and the charge carried on the specified conductor.
```

■ Field Values

```
EOGetPointValues[x,y] returns the various field values
  associated with the point at {x,y}. The return values, in order, are:
 V – Voltage
 Dx – x– or r– direction component of displacement
 Dy – y– or z– direction component of displacement
 Ex – x– or r– direction component of electric field intensity
 Ey – y– or z– direction component of electric field intensity
 ex – x– or r– direction component of permittivity
 ey – y– or z– direction component of permittivity
 nrg – electric field energy density
 An equivalent form is:
 EOGetPointValues[x,y]
```

```
EOGetV[x,y] returns the voltage at point {x,y}. An equivalent form is:
 EOGetV[{x,y}]
```

```
EOGetD[x,y] returns a list with two elements containing
  the electric flux density at point {x,y}.  An equivalent form is:
 EOGetD[{x,y}]
```

```
EOGetE[x,y] returns a list with two elements containing
  the electric field intensity at point {x,y}.  An equivalent form is:
 EOGetE[{x,y}]
```

```
EOGetEnergyDensity[x,y] returns the energy
  density, (E.D)/2, at point {x,y}.  An equivalent form is:
 EOGetEnergyDensity[{x,y}]
```

```
EOGetPerm[x,y] returns a list with two elements
  containing the permittivity at point {x,y}. An equivalent form is:
 EOGetPerm[{x,y}]
```

■ View Manipulation

```
EOShowContourPlot[numcontours,lowerV,upperV] shows the V contour plot with options:
 numcontours – Number of equipotential lines to be plotted.
 upperV – Upper limit for contours.
 lowerV – Lower limit for contours.
 If numcontours is –1 all parameters are ignored and default values are used
```

```
EOHideContourPlot[] hides the currently displayed contour plot
```

```
EOShowDensityPlot[legend,gscale,type,upperD,lowerD]
 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.
 upperD – Sets the upper display limit for the density plot.
 lowerD – Sets the lower display limit for the density plot.
 type – Sets the type of density plot. A value of 0
  plots voltage, 1 plots the magnitude of D, and 2 plots the magnitude of E
```

```
EOHideDensityPlot[] hides the currently displayed density plot
```

```
EOSetGrid[density,"type"] changes 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.
```

EOShowGrid[] displays the grid points

EOHideGrid[] hides the grid points from view in the electrostatics output window

EOShowMesh[] displays the mesh

EOHideMesh[] hides the finite element mesh from in the electrostatics output view

EOShowNames[] displays the material names associated with each block label

EOHideNames[] hides material names associated
with block labels from view in the electrostatics output window

EOShowPoints[] displays the input node point as part of the output geometry

EOHidePoints[] inhibits the display of input node points in the electrostatics output window

EOSmoothOn[] turns on smoothing of the D and E fields. The D and E fields
are then displayed using a linear interpolation of the field from nodal values

EOSmoothOff[] turns off smoothing of the D and E fields. The D and E fields
are then displayed using values that are piecewise constant over each element.

EOSnapGridOn[] turns on snap-to-grid

EOSnapGridOff[] turns off snap-to-grid

EOZoom[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}. Equivalent forms are:
EOZoom[{x1,y1},{x2,y2}]
EOZoom[{{x1,y1},{x2,y2}}]

EOZoomIn[] zooms out by a factor of 200%.

EOZoomNatural[] zooms to a "natural" view with sensible extents.

EOZoomOut[] zooms out by a factor of 50%.

EORefreshView[] Redraws the current view.

EOGetView[] grabs the current electrostatics output view and returns a bitmapped
graphics object. This object can subsequently be displayed using the Show[] command

■ Miscellaneous

EOClose[] close the current postprocessor document and window

EOGetProblemInfo[] returns a list with two values: the Problem Type (0 for planar and
1 for axisymmetric) and the depth assumed for planar problems in units of meters.

EOReload[] reloads the current electrostatics output file.

EOSaveBitmap["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 EISaveAs command.

EOSaveMetafile["filename"] saves an extended metafile screenshot
of the current view to the file specified by "filename", subject to the
printf-type formatting explained previously for the EISaveAs command.

EOSetEditMode["mode"] sets the mode of the postprocessor to point,
contour, or area mode. Valid entries for mode are "point", "contour", and "area".

■ Heat Flow Output Commands

■ Contours, Regions, and Integrals

HOAddContour[x,y] adds the point {x,y} to the contour that is used for plotting values along lines and for computing line integrals. An equivalent form is: HOAddContour[{x,y}]

HOBendContour[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.

HOClearContour[] clears the current contour selection

HOSelectBlock[x,y] select the block that contains point {x,y}. An equivalent form is: HOSelectBlock[{x,y}]

HOSelectPoint[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. An equivalent form is: HOSelectPoint[{x,y}]

HOGroupSelectBlock[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. HOGroupSelectBlock[]), all blocks are selected.

HOClearBlock[] clears the current block selection

HOBlockIntegral[type] calculate a block integral for the selected blocks. The type parameter can take on the following values:
0 - Average Temperature
1 - Block Cross-section
2 - Block Volume
3 - Average F over the block
4 - Average G over the block
returns either a single value or a list of two values, depending on the specified type

HOLineIntegral[type] calculate the line integral for the defined contour. Valid type selections include:
0 - Temperature Difference (G·t)
1 - Heat Flux (F·n)
2 - Contour length
3 - Average temperature
This integral returns either one value or a list of two values, depending on the type of integral

```
HOMakePlot[PlotType,NumPoints,Filename,FileFormat]
 This function allows programmatic access to FEMM's 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 specfied file name,
  rather than display it to make a graphical plot. Valid entries for PlotType are:
 0 - T (temperature)
 1 - |F| (Magnitude of heat flux density)
 2 - F . n (Normal heat flux density)
 3 - F . t (Tangential heat flux density)
 4 - |G| (Magnitude of field intensity)
 5 - G . n (Normal temperature gradient)
 6 - G . t (Tangential temperature gradient)
```

```
HOGetConductorProperties["conductor"] returns properties for the conductor
  property named "conductor". A list with two values is returned: The temperature
  of the specified conductor, and the total heat flux from the specified conductor.
```

■ Field Values

```
HOGetPointValues[x,y] returns the various field values
  associated with the point at {x,y}. The return values, in order, are:
 T - temperature
 Fx - x- or r- direction component of heat flux density
 Fy - y- or z- direction component of heat flux density
 Gx - x- or r- direction component of temperature gradient
 Gy - y- or z- direction component of temperature gradient
 Kx - x- or r- direction component of thermal conductivity
 Ky - y- or z- direction component of thermal conductivity
 An equivalent form is:
 HOGetPointValues[x,y]
```

```
HOGetT[x,y] returns the temperature at point {x,y}. An equivalent form is:
 HOGetT[{x,y}]
```

```
HOGetK[x,y] returns a list with two elements containing
  the thermal conductivity at point {x,y}. An equivalent form is:
 HOGetK[{x,y}]
```

```
HOGetD[x,y] returns a list with two elements
  containing the heat flux density at point {x,y}.  An equivalent form is:
 HOGetF[{x,y}]
```

```
HOGetE[x,y] returns a list with two elements containing
  the temperature gradient at point {x,y}.  An equivalent form is:
 HOGetG[{x,y}]
```

■ View Manipulation

```
HOShowContourPlot[numcontours,lowerV,upperV] shows the V contour plot with options:
 numcontours - Number of equipotential lines to be plotted.
 upperV - Upper limit for contours.
 lowerV - Lower limit for contours.
 If numcontours is -1 all parameters are ignored and default values are used
```

```
HOHideContourPlot[] hides the currently displayed contour plot
```

HOShowDensityPlot[legend,gscale,type,upperD,lowerD]
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.
upperD – Sets the upper display limit for the density plot.
lowerD – Sets the lower display limit for the density plot.
type – Sets the type of density plot. A value of 0 plots temperature, 1 plots the magnitude of D, and 2 plots the magnitude of E

HOHideDensityPlot[] hides the currently displayed density plot

HOSetGrid[density,"type"] changes 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.

HOShowGrid[] displays the grid points

HOHideGrid[] hides the grid points from view in the heat flow output window

HOShowMesh[] displays the mesh

HOHideMesh[] hides the finite element mesh from in the heat flow output view

HOShowNames[] displays the material names associated with each block label

HOHideNames[] hides material names associated with block labels from view in the heat flow output window

HOShowPoints[] displays the input node point as part of the output geometry

HOHidePoints[] inhibits the display of input node points in the heat flow output window

HOSmoothOn[] turns on smoothing of the D and E fields. The D and E fields are then displayed using a linear interpolation of the field from nodal values

HOSmoothOff[] turns off smoothing of the D and E fields. The D and E fields are then displayed using values that are piecewise constant over each element.

HOSnapGridOn[] turns on snap-to-grid

HOSnapGridOff[] turns off snap-to-grid

HOZoom[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}. Equivalent forms are:
HOZoom[{x1,y1},{x2,y2}]
HOZoom[{{x1,y1},{x2,y2}}]

HOZoomIn[] zooms out by a factor of 200%.

HOZoomNatural[] zooms to a "natural" view with sensible extents.

HOZoomOut[] zooms out by a factor of 50%.

HORefreshView[] Redraws the current view.

HOGetView[] grabs the current heat flow output view and returns a bitmapped graphics object. This object can subsequently be displayed using the Show[] command

■ Miscellaneous

HOClose[] close the current postprocessor document and window

HOGetProblemInfo[] returns a list with two values: the Problem Type (0 for planar and 1 for axisymmetric) and the depth assumed for planar problems in units of meters.

HOReload[] reloads the current heat flow output file.

HOSaveBitmap["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 HISaveAs command.

HOSaveMetafile["filename"] saves an extended metafile screenshot of the current view to the file specified by "filename", subject to the printf-type formatting explained previously for the HISaveAs command.

HOSetEditMode["mode"] sets the mode of the postprocessor to point, contour, or area mode. Valid entries for mode are "point", "contour", and "area".

Current Flow Output Commands

Contours, Regions, and Integrals

COAddContour[x,y] adds the point {x,y} to the contour that is used for plotting values along lines and for computing line integrals. An equivalent form is: COAddContour[{x,y}]

COBendContour[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.

COClearContour[] clears the current contour selection

COSelectBlock[x,y] select the block that contains point {x,y}. An equivalent form is: COSelectBlock[{x,y}]

COSelectPoint[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. An equivalent form is: COSelectPoint[{x,y}]

COGroupSelectBlock[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. COGroupSelectBlock[]), all blocks are selected.

COClearBlock[] clears the current block selection

```
COBlockIntegral[type] calculate a block integral for the
  selected blocks. The type parameter can take on the following values:
      0   Real Power
      1   Reactive Power
      2   Apparent Power
      3   Time-Average Stored Energy
      4   Block cross-section area
      5   Block volume
      6   x (or r) direction Weighted Stress Tensor force, DC component
      7   y (or z) direction Weighted Stress Tensor force, DC component
      8   x (or r) direction Weighted Stress Tensor force, 2x frequency component
      9   y (or z) direction Weighted Stress Tensor force, 2x frequency component
      10  Weighted Stress Tensor torque, DC component
      11  Weighted Stress Tensor torque, 2x frequency component
 Returns either a single value or a list of two values, depending on the specified type
```

```
COLineIntegral[type] calculate the line
  integral for the defined contour. Valid type selections include:
  0  E.t
  1  J.n
  2  Contour length
  3  Average voltage over contour
  4  Force from stress tensor
  5  Torque from stress tensor
 This integral returns either one value
  or a list of two values, depending on the type of integral
```

```
COMakePlot[PlotType,NumPoints,Filename,FileFormat]
 This function allows programmatic access to FEMM's 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 specfied file name,
  rather than display it to make a graphical plot. Valid entries for PlotType are:
  0   V (Voltage)
  1   |J| (Magnitude of current density)
  2   J.n (Normal current density)
  3   J.t (Tangential current density)
  4   |E (Magnitude of field intensity)
  5   E.n (Normal field intensity)
  6   E.t (Tangential field intensity)
  7   |Jc| (Magnitude of conduction current density)
  8   Jc.n (Normal conduction current density)
  9   Jc.t (Tangential conduction current density)}
  10  |Jd| (Magnitude of displacement current density)
  11  Jd.n (Normal displacement current density)
  12  Jd.t (Tangential displacement current density)
```

```
COGetConductorProperties["conductor"] returns properties for the conductor
  property named "conductor". A list with two values is returned: The voltage of
  the specified conductor, and the total current from the specified conductor.
```

■ Field Values

```
COGetPointValues[x,y] returns the various field values
  associated with the point at {x,y}. The return values, in order, are:
 V   & Voltage
 Jx  & x- or r- direction component of current density
 Jy  & y- or z- direction component of current density
 Kx  & x- or r- direction component of AC conductivity
 Ky  & y- or z- direction component of AC conductivity
 Ex  & x- or r- direction component of electric field intensity
 Ey  & y- or z- direction component of electric field intensity
 ex  & x- or r- direction component of permittivity
 ey  & y- or z- direction component of permittivity
 Jdx & x- or r- direction component of displacement current density
 Jdy & y- or z- direction component of displacement current density
 ox  & x- or r- direction component of permittivity
 oy  & y- or z- direction component of permittivity
 Jcx & x- or r- direction component of conduction current density
 Jcy & y- or z- direction component of conduction current density
 An equivalent form is:
 COGetPointValues[{x,y}]
```

```
COGetV[x,y] returns the voltage at point {x,y}. An equivalent form is:
 COGetV[{x,y}]
```

```
COGetJ[x,y] returns a list with two elements
  containing the current density at point {x,y}.  An equivalent form is:
 COGetJ[{x,y}]
```

```
COGetK[x,y] returns a list with two elements
  containing the ac conductivity at point {x,y}.  An equivalent form is:
 COGetK[{x,y}]
```

```
COGetE[x,y] returns a list with two elements containing
  the  electric field intensity at point {x,y}. An equivalent form is:
 COGetE[{x,y}]
```

■ View Manipulation

```
COShowContourPlot[numcontours,lowerV,upperV] shows the V contour plot with options:
 numcontours - Number of equipotential lines to be plotted.
 upperV - Upper limit for contours.
 lowerV - Lower limit for contours.
 If numcontours is -1 all parameters are ignored and default values are used
```

```
COHideContourPlot[] hides the currently displayed contour plot
```

```
COShowDensityPlot[legend,gscale,type,upper,lower]
 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 - Sets the upper display limit for the density plot.
 lower - Sets the lower display limit for the density plot.
 type - Sets the type of density plot. Plot types include:
 0  |V|
 1  |Re(V)|
 2  |Im(V)|
 3  |J|
 4  |Re(J)|
 5  |Im(J)|
 6  |E|
 7  |Re(E)|
 8  |Im(E)|

COHideDensityPlot[] hides the currently displayed density plot

COSetGrid[density,"type"] changes 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.

COShowGrid[] displays the grid points

COHideGrid[] hides the grid points from view in the current output window

COShowMesh[] displays the mesh

COHideMesh[] hides the finite element mesh from in the current output view

COShowNames[] displays the material names associated with each block label

COHideNames[] hides material names associated
  with block labels from view in the current output window

COShowPoints[] displays the input node point as part of the output geometry

COHidePoints[] inhibits the display of input node points in the current output window

COSmoothOn[] turns on smoothing of the D and E fields.  The D and E fields
  are then displayed using a linear interpolation of the field from nodal values

COSmoothOff[] turns off smoothing of the D and E fields.  The D and E fields
  are then displayed using values that are piecewise constant over each element.

COSnapGridOn[] turns on snap-to-grid

COSnapGridOff[] turns off snap-to-grid

COZoom[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}. Equivalent forms are:
 COZoom[{x1,y1},{x2,y2}]
 COZoom[{{x1,y1},{x2,y2}}]

COZoomIn[] zooms out by a factor of 200%.

COZoomNatural[] zooms to a "natural" view with sensible extents.

COZoomOut[] zooms out by a factor of 50%.

CORefreshView[] Redraws the current view.
```

COGetView[] grabs the current output view and returns a bitmapped graphics object. This object can subsequently be displayed using the Show[] command

■ Miscellaneous

COClose[] close the current postprocessor document and window

COGetProblemInfo[] returns a list with two values: the Problem Type (0 for planar and 1 for axisymmetric) and the depth assumed for planar problems in units of meters.

COReload[] reloads the current output file.

COSaveBitmap["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 CISaveAs command.

COSaveMetafile["filename"] saves an extended metafile screenshot of the current view to the file specified by "filename", subject to the printf-type formatting explained previously for the CISaveAs command.

COSetEditMode["mode"] sets the mode of the postprocessor to point, contour, or area mode. Valid entries for mode are "point", "contour", and "area".

■ Magnetics Output Commands

■ Contours, Regions, and Integrals

MOAddContour[x,y] adds the point {x,y} to the contour that is used for plotting values along lines and for computinlocatedg line integrals. An equivalent form is: MOAddContour[{x,y}]

MOBendContour[angle,anglestep] replaces the straight line formed by the last two points in curvethe 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.

MOClearContour[] clears the current contour selection

MOSelectBlock[x,y] select the block that contains point {x,y}. An equivalent form is: MOSelectBlock[{x,y}]

MOSelectPoint[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. An equivalent form is: MOSelectPoint[{x,y}]

MOGroupSelectBlock[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. MOGroupSelectBlock[]), all blocks are selected.

MOClearBlock[] clears the current block selection

```
MOBlockIntegral[type] calculates a block integral over the selected blocks.
  There is a single (possibly complex) return value. Valid type specifications are:
 0 – A·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 Bx (or Br) over block
 9 – Integral of By (or Bz) 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 2X Lorentz force
 14 – y (or z) part of 2X Lorentz force
 15 – Steady–state Lorentz torque
 16 – 2X 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 2X weighted stress tensor force
 21 – y (or z) part of 2X weighted stress tensor force
 22 – Steady–state weighted stress tensor torque
 23 – 2X component of weighted stress tensor torque
 24 – R^2 (i.e. moment of inertia / density

MOLineIntegral[type] calculates the line integral for the defined contour.
  The following types of line integral and the return values are as follows:
 0 – total B.n, average B.n over the contour
 1 – total H.t, average H.t over the contour
 2 – Contour length, surface area
 3 – Stress Tensor Force: DC r/x force, DC y/z force, 2X r/x force, 2X y/z force
 4 – Stress Tensor Torque: DC torque, 2X torque
 5 – total (B.n)^2, average (B.n)^2
 Typically returns two (possibly complex) values as
  results. However, the Stress Tensor Force integral returns four values.

MOMakePlot[PlotType,NumPoints,Filename,FileFormat]
 This function allows programmatic access to FEMM's 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 specfied file name,
  rather than display it to make a graphical plot. Valid entries for PlotType are:
 0 – Potential
 1 – |B|
 2 B·n
 3 – B·t
 4 – |H|
 5 – H ·n
 6 – H ·t
 7 – Jeddy
 8 – Jsource+Jeddy
```

MOGetCircuitProperties["circuit"] is used primarily to obtain impedance information associated with circuit properties. Properties are returned for the circuit property named "circuit". A list of three values is returned by the function. In order, the elements of this list are:
current – the current carried by the circuit.
volts – the voltage drop across the circuit in the circuit.
flux – the circuit's flux linkage
Any of these entries could possibly be complex-valued.

■ Field Values

MOGetPointValues[x,y] get the field values at {x,y}. The function returns a list of values which represent:
A – vector potential A, flux=2*Pi*r*A if axisymmetric
B1 – Bx if planar, Br if axisymmetric
B2 – By if planar, Bz if axisymmetric
Sig – conductivity
E – stored energy density
H1 – Hx if planar, Hr if axisymmetric
H2 – Hy if planar, Hz if axisymmetric
Je – Eddy current contribution to current density
Js – Source current contribution to current density
Mu1 – mux if planar, mur if axisymmetric
Mu2 – muy if planar, muz if axisymmetric
Pe – Power density dissipated through ohmic losses
Ph – Power density dissipated by hysteresis
ff – Winding fill factor

MOGetA[x,y] returns the vector potential, A, for 2D planar problems, and it returns flux, 2*Pi*r*A, for axisymmetric problems. The return value is possbly complex-valued. An equivalent form is:
MOGetA[{x,y}]

MOGetB[x,y] returns a list with two elements containing the magnetic flux density at point {x,y}. An equivalent form is:
MOGetB[{x,y}]

MOGetH[x,y] returns a list with two elements containing the magnetic field intensity at point {x,y}. An equivalent form is:
MOGetB[{x,y}]

MOGetJ[x,y] returns the electric current density at point {x,y}. An equivalent form is:
MOGetJ[{x,y}]

MOGetMu[x,y] returns a list with two elements containing the magnetic permeability at point {x,y}. An equivalent form is:
MOGetMu[{x,y}]

MOGetPe[x,y] returns the electrical (ohmic) loss density at point {x,y}. An equivalent form is:
MOGetMu[{x,y}]

MOGetPh[x,y] returns the hysteresis and/or laminated eddy current loss density at point {x,y}. An equivalent form is:
MOGetPh[{x,y}]

MOGetConductivity[x,y] returns the electrical conductivity at point {x,y}. An equivalent form is:
MOGetConductivity[{x,y}]

```
MOGetEnergyDensity[x,y] returns the energy
  density in the magnetic field at point {x,y}. An equivalent form is:
 MOGetEnergyDensity[{x,y}]
```

■ View Manipulation

```
MOShowContourPlot[numcontours,lowerA,upperA,type] shows the A contour plot with options:
 numcontours – Number of A equipotential lines to be plotted.
 upperA – Upper limit for A contours.
 lowerA – Lower limit for A contours.
 type – Choice of "real", "imag", or "both" to show
  either the real, imaginary of both real and imaginary components of A.
 If numcontours is -1, all parameters are ignored and default values are used

MOHideContourPlot[] hides the currently displayed contour plot

MOShowDensityPlot[legend,gscale,upperB,lowerB,type]
 shows the flux density plot with the 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 scaledensity plot.
 upperB – Sets the upper display limit for the density plot.
 lowerB – 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

MOHideDensityPlot[] hides the currently displayed density plot

MOSetGrid[density,"type"] changes 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.

MOShowGrid[] displays the grid points

MOHideGrid[] hides the grid points from view in the electrostatics output window

MOShowMesh[] displays the mesh

MOHideMesh[] hides the finite element mesh from in the electrostatics output view

MOShowNames[] displays the material names associated with each block label

MOHideNames[] hides material names associated
  with block labels from view in the electrostatics output window

MOShowPoints[] displays the input node point as part of the output geometry

MOHidePoints[] inhibits the display of input node points in the electrostatics output window

MOSmoothOn[] turns on smoothing of the D and E fields.  The D and E fields
  are then displayed using a linear interpolation of the field from nodal values

MOSmoothOff[] turns off smoothing of the D and E fields.  The D and E fields
  are then displayed using values that are piecewise constant over each element.

MOSnapGridOn[] turns on snap-to-grid

MOSnapGridOff[] turns off snap-to-grid
```

MOZoom[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}. Equivalent forms are:
MOZoom[{x1,y1},{x2,y2}]
MOZoom[{{x1,y1},{x2,y2}}]

MOZoomIn[] zooms out by a factor of 200%.

MOZoomNatural[] zooms to a "natural" view with sensible extents.

MOZoomOut[] zooms out by a factor of 50%.

MORefreshView[] Redraws the current view.

MOGetView[] grabs the current electrostatics output view and returns a bitmapped graphics object. This object can subsequently be displayed using the Show[] command

■ Miscellaneous

MOClose[] close the current postprocessor document and window

MOGetProblemInfo[] returns a list of two values: the problem type (0 for planar, 1 for axisymmetric) and the analysis frequency in Hz.

MOReload[] reloads the current electrostatics output file.

MOSaveBitmap["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 EISaveAs command.

MOSaveMetafile["filename"] saves an extended metafile screenshot of the current view to the file specified by "filename", subject to the printf-type formatting explained previously for the EISaveAs command.

MOSetEditMode["mode"] sets the mode of the postprocessor to point, contour, or area mode. Valid entries for mode are "point", "contour", and "area".