Finite Element Method Magnetics: OctaveFEMM

Version 1.2

User's Manual

October 25, 2015

David Meeker
dmeeker@ieee.org
http://femm.foster-miller.com
©2006

1   Introduction

OctaveFEMM is a Matlab toolbox that allows for the operation of Finite Element Method Magnetics (FEMM) via a set of Matlab functions. The toolbox works with Octave, a Matlab clone.

When OctaveFEMM starts up a FEMM process, the usual FEMM user interface is displayed and is fully functional. The user then has the choice of accomplishing modeling and analysis tasks either exclusively through functions implemented by the toolbox, or by a combination of manual and programmatic operations – whichever is easiest for the task at hand.

The syntax of the OctaveFEMM toolbox closely mirrors that of FEMM's existing Lua scripting language interface associated with FEMM v4.2. However, there are some differences between the Lua functions and the analogous Octave/Matlab implementations:

- All strings are enclosed in single quotes, rather than double quotes as in Lua.

- Functions in Lua that have no arguments require a set of empty parentheses after the function name (*e.g.* `mi_analyze()`). In Octave or Matlab, no parentheses should be used needed (*e.g.* `mi_analyze` with the OctaveFEMM toolbox).

- Several commands have also been added to OctaveFEMM that have no analog in Lua. These commands streamline the drawing of new geometries with the OctaveFEMM toolbox, as well is the collection of data from solutions.

Perhaps the most remarkable difference between Lua and OctaveFEMM, however, is due to the matrix-oriented nature of Octave/Matlab. In just about any OctaveFEMM function in which it would be desiable to enter an array of points such that multiple copies of an operation are performed, OctaveFEMM will correctly interpret the input perform the requested operation on every element in the array. In addition, for any function in which the coordinates of a point are required, that point can be specified as an array with two elements instead of specifying each element separately. In functions that require the specification of multiple points, those points can be entered as an array of two-element arrays.

2   Installation and Startup

2.1   Installation for Matlab and Octave 3

The OctaveFEMM distribution is automatically installed with FEMM 4.2 in the `mfiles` subdirectory. The absolute directory is typically `c:\Program Files\femm42\mfiles`. To use Octave-FEMM with Octave or Matlab, this path needs to be added to the program's search path. To add this path to the search path, type the following lines at the Matlab or Octave 3.X.X command prompt:

```
addpath('c:\\femm42\\mfiles');
savepath;
```

Alternatively, in Matlab, the interactive `pathtool` command can be used to add the `mfiles` directory to the search path.

2.2 Installation for Octave 2.1.50 and Octave 2.1.73

It is recommended that you use a newer version of Octave that has support for the `actxserver` function. If this function exists, Octave will use ActiveX automation to communication with FEMM. However, OctaveFEMM can still be used with versions of Octave (*e.g.* Octave 2.1.50 and 2.1.73) that do not support ActiveX. If `actxserver` is not available, a much slower interprocess communication mechanism based on temporary files will be used.

Again, to use OctaveFEMM, the directory where the OctaveFEMM mfiles are located must be permanently added to the search path. This directory can be added to the search path by an appropriate modification of the `.octaverc` initialization file. For example, in the Octave 2.1.50 release, Octave looks for `.octaverc` in the directory:

`C:\Program Files\GNU Octave 2.1.50\octave_files`

To insert OctaveFEMM into the search path, one can create the `.octaverc file` and add the line:

`LOADPATH = [ '/cygdrive/c/progra~1/femm42/mfiles//', LOADPATH ];`

to add the directory containing the OctaveFEMM commands to the Octave search path.

2.3 Startup

To start an OctaveFEMM session, use the `openfemm` function. This function starts up a FEMM process to which OctaveFEMM calls are sent. When done with OctaveFEMM, the FEMM process can be shut down via the `closefemm` function.

A number of examples that use OctaveFEMM to analyze various problems are included in the directory `cd c:\Program Files\femm42\examples`

3 Common Command Set

There are a number of FEMM-specific Octave that are not associated with any particular problem type. These functions manipulate the appearance of the main window and other top-level components like the Lua console and Point Properties output window.

- `clearconsole` Clears the output window of the Lua console.

- `newdocument(doctype)` Creates a new preprocessor document and opens up a new preprocessor window. 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. Alternative syntax for this function is `create(doctype)`

- `hideconsole` Hides the floating Lua console window.

- `hidepointprops` Hides the floating FEMM Properties display window.

- `messagebox('message')` displays the `'message'` string to the screen in a pop-up message box.

- `opendocument('filename')` Opens a document specified by `filename`.

- `print(item1,item2,...)` This is standard Lua "print" command directed to the output of the Lua console window. Any number of comma-separated items can be printed at once via the print command.

- `prompt('message')` This function allows a FEMM script to prompt a user for input. When this command is used, a dialog box pops up with the `'message'` string on the title bar of the dialog box. The user can enter in a single line of input via the dialog box. Prompt returns the user's input to Octave and parses it using Octave's `eval` command.

- `showconsole` Displays the floating Lua console window.

- `showpointprops` Displays the floating FEMM Properties display window.

- `smartmesh(state)` Calling with a state of 0 turns off "smart mesh" functionality f or the present session; calling with a state of 1 turns "smarth meshing" on. The setting os not permanent–using the Preferences setting to permanently turn on or off.

- `main_minimize` minimizes the main FEMM window.

- `main_maximize` maximizes the main FEMM window.

- `main_restore` restores the main FEMM window from a minimized or maximized state.

- `main_resize(width,height)` resizes the main FEMM window client area to width $\times$ height.

4 Magnetics Preprocessor Command Set

A number of different commands are available in the preprocessor.

4.1 Object Add/Remove Commands

- `mi_addnode(x,y)` Add a new node at x,y

- `mi_addsegment(x1,y1,x2,y2)` Add a new line segment from node closest to (x1,y1) to node closest to (x2,y2)

- `mi_addblocklabel(x,y)` Add a new block label at (x,y)

- `mi_addarc(x1,y1,x2,y2,angle,maxseg)` Add a new arc segment from the nearest node to (x1,y1) to the nearest node to (x2,y2) with angle 'angle' divided into 'maxseg' segments.

- `mi_drawline(x1,y1,x2,y2)` Adds nodes at (x1,y1) and (x2,y2) and adds a line between the nodes.

- `mi_drawpolyline([x1,y1;x2,y2'...])` Adds nodes at each of the specified points and connects them with segments.

- `mi_drawpolygon([x1,y1;x2,y2'...])` Adds nodes at each of the specified points and connects them with segments to form a closed contour.

- `mi_drawarc(x1,y1,x2,y2,angle,maxseg)` Adds nodes at (x1,y1) and (x2,y2) and adds an arc of the specified angle and discretization connecting the nodes.

- `mi_drawrectangle(x1,y1,x2,y2)` Adds nodes at the corners of a rectangle defined by the points (x1,y1) and (x2,y2), then adds segments connecting the corners of the rectangle.

- `mi_deleteselected` Delete all selected objects.

- `mi_deleteselectednodes` Delete selected nodes.

- `mi_deleteselectedlabels` Delete selected block labels.

- `mi_deleteselectedsegments` Delete selected segments.

- `mi_deleteselectedarcsegments` Delete selects arcs.

4.2   Geometry Selection Commands

- `mi_clearselected` Clear all selected nodes, blocks, segments and arc segments.

- `mi_selectsegment(x,y)` Select the line segment closest to (x,y)

- `mi_selectnode(x,y)` Select the node closest to (x,y). Returns the coordinates of the selected node.

- `mi_selectlabel(x,y)` Select the label closet to (x,y). Returns the coordinates of the selected label.

- `mi_selectarcsegment(x,y)` Select the arc segment closest to (x,y)

- `mi_selectgroup(n)` Select the nth group of nodes, segments, arc segments and blocklabels. This function will clear all previously selected elements and leave the editmode in 4 (group)

- `mi_selectcircle(x,y,R,editmode)` selects objects within a circle of radius R centered at (x,y). If only x, y, and R paramters are given, the current edit mode is used. If the editmode parameter is used, 0 denotes nodes, 2 denotes block labels, 2 denotes segments, 3 denotes arcs, and 4 specifies that all entity types are to be selected.

- `mi_selectrectangle(x1,y1,x2,y2,editmode)` selects objects within a rectangle defined by points (x1,y1) and (x2,y2). If no editmode parameter is supplied, the current edit mode is used. If the editmode parameter is used, 0 denotes nodes, 2 denotes block labels, 2 denotes segments, 3 denotes arcs, and 4 specifies that all entity types are to be selected.

4.3   Object Labeling Commands

- `mi_setnodeprop('propname',groupno)` Set the selected nodes to have the nodal property `'propname'` and group number `groupno`.

- `mi_setblockprop('blockname', automesh, meshsize, 'incircuit', magdir, group, turns)` Set the selected block labels to have the properties:

  - Block property `'blockname'`.
  - `automesh`: 0 = mesher defers to mesh size constraint defined in `meshsize`, 1 = mesher automatically chooses the mesh density.
  - `meshsize`: size constraint on the mesh in the block marked by this label.
  - Block is a member of the circuit named `'incircuit'`
  - The magnetization is directed along an angle in measured in degrees denoted by the parameter `magdir`
  - A member of group number `group`
  - The number of turns associated with this label is denoted by `turns`.

- `mi_setsegmentprop('propname', elementsize, automesh, hide, group)` Set the selected segments to have:

  - Boundary property `'propname'`
  - Local element size along segment no greater than `elementsize`
  - `automesh`: 0 = mesher defers to the element constraint defined by `elementsize`, 1 = mesher automatically chooses mesh size along the selected segments
  - `hide`: 0 = not hidden in post-processor, 1 == hidden in post processor
  - A member of group number `group`

- `mi_setarcsegmentprop(maxsegdeg, 'propname', hide, group)` Set the selected arc segments to:

  - Meshed with elements that span at most `maxsegdeg` degrees per element
  - Boundary property `'propname'`
  - `hide`: 0 = not hidden in post-processor, 1 == hidden in post processor
  - A member of group number `group`

- `mi_setgroup(n)` Set the group associated of the selected items to n.

4.4 Problem Commands

- `mi_probdef(freq,units,type,precision,depth,minangle,(acsolver))` changes the problem definition. Set `freq` to the desired frequency in Hertz. The `units` parameter specifies the units used for measuring length in the problem domain. Valid `'units'` entries are `'inches'`, `'millimeters'`, `'centimeters'`, `'mils'`, `'meters'`, and `'micrometers'`. Set the parameter `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, entering `1E-8` requires the RMS of the residual to be less than 10^{-8}. A fifth parameter, representing the depth of the problem in the into-the-page direction for 2-D planar problems. Specify the depth to be zero for axisymmetric problems. The sixth parameter represents the minimum angle constraint sent to the mesh generator – 30 degrees is the usual choice for this parameter. The acsolver parameter specifies which solver is to be used for AC problems: 0 for successive approximation, 1 for Newton.

- `mi_analyze(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`.

- `mi_loadsolution` loads and displays the solution corresponding to the current geometry.

- `mi_setfocus('documentname')` Switches the magnetics input file upon which commands are to act. If more than one magnetics 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.

- `mi_saveas('filename')` saves the file with name `'filename'`.

4.5 Mesh Commands

- `mi_createmesh` runs triangle to create a mesh. Note that this is not a necessary precursor of performing an analysis, as `mi_analyze` will make sure the mesh is up to date before running an analysis. The number of elements in the mesh is pushed back onto the lua stack.

- `mi_showmesh` shows the mesh.

- `mi_purgemesh` clears the mesh out of both the screen and memory.

4.6 Editing Commands

- `mi_copyrotate(bx, by, angle, copies )`

    - `bx, by` – base point for rotation
    - `angle` – angle by which the selected objects are incrementally shifted to make each copy. `angle` is measured in degrees.
    - `copies` – number of copies to be produced from the selected objects.

7

- `mi_copyrotate2(bx, by, angle, copies, editaction )`

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

- `mi_copytranslate(dx, dy, copies)`

    - `dx,dy` – distance by which the selected objects are incrementally shifted.
    - `copies` – number of copies to be produced from the selected objects.

- `mi_copytranslate2(dx, dy, copies, editaction)`

    - `dx,dy` – distance by which the selected objects are incrementally shifted.
    - `copies` – number of copies to be produced from the selected objects.
    - `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4- group

- `mi_createradius(x,y,r)`turnsacornerlocatedat`(x,y)`intoacurveofradius`r`.

- `mi_moverotate(bx,by,shiftangle)`

    - `bx, by` – base point for rotation
    - `shiftangle` – angle in degrees by which the selected objects are rotated.

- `mi_moverotate2(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

- `mi_movetranslate(dx,dy)`

    - `dx,dy` – distance by which the selected objects are shifted.

- `mi_movetranslate2(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

- `mi_scale(bx,by,scalefactor)`

    - `bx, by` – base point for scaling
    - `scalefactor` – a multiplier that determines how much the selected objects are scaled

- `mi_scale2(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

- mi_mirror(x1,y1,x2,y2) mirror the selected objects about a line passing through the points (x1,y1) and (x2,y2).

- mi_mirror2(x1,y1,x2,y2,editaction) mirror the selected objects about a line passing through the points (x1,y1) and (x2,y2). Valid editaction entries are 0 for nodes, 1 for lines (segments), 2 for block labels, 3 for arc segments, and 4 for groups.

- mi_seteditmode(editmode) Sets the current editmode to:

  - 'nodes' - nodes
  - 'segments' - line segments
  - 'arcsegments' - arc segments
  - 'blocks' - block labels
  - 'group' - selected group

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

4.7   Zoom Commands

- mi_zoomnatural zooms to a "natural" view with sensible extents.

- mi_zoomout zooms out by a factor of 50%.

- mi_zoomin zoom in by a factor of 200%.

- mi_zoom(x1,y1,x2,y2) Set the display area to be from the bottom left corner specified by (x1,y1) to the top right corner specified by (x2,y2).

4.8   View Commands

- mi_showgrid Show the grid points.

- mi_hidegrid Hide the grid points points.

- mi_grid_snap('flag') Setting flag to 'on' turns on snap to grid, setting flag to 'off' turns off snap to grid.

- mi_setgrid(density,'type') Change the grid spacing. The density parameter specifies the space between grid points, and the type parameter is set to 'cart' for cartesian coordinates or 'polar' for polar coordinates.

- mi_refreshview Redraws the current view.

9

- `mi_minimize` minimizes the active magnetics input view.

- `mi_maximize` maximizes the active magnetics input view.

- `mi_restore` restores the active magnetics input view from a minimized or maximized state.

- `mi_resize(width,height)` resizes the active magnetics input window client area to width × height.

4.9 Object Properties

- `mi_getmaterial('materialname')` fetches the material specified by `materialname` from the materials library.

- `mi_addmaterial('matname', mu_x, mu_y, H_c, J, Cduct, Lam_d, Phi_hmax, lam_fill, LamType, Phi_hx, Phi_hy, nstr, dwire)` adds a new material with called `'matname'` with the material properties:

  - `mu_x` Relative permeability in the x- or r-direction.
  - `mu_y` Relative permeability in the y- or z-direction.
  - `H_c` Permanent magnet coercivity in Amps/Meter.
  - `J` Applied source current density in Amps/mm^2.
  - `Cduct` Electrical conductivity of the material in MS/m.
  - `Lam_d` Lamination thickness in millimeters.
  - `Phi_hmax` Hysteresis lag angle in degrees, used for nonlinear BH curves.
  - `Lam_fill` Fraction of the volume occupied per lamination that is actually filled with iron (Note that this parameter defaults to 1 in the `femm` preprocessor dialog box because, by default, iron completely fills the volume)
  - `Lamtype` Set to
    * 0 – Not laminated or laminated in plane
    * 1 – laminated x or r
    * 2 – laminated y or z
    * 3 – magnet wire
    * 4 – plain stranded wire
    * 5 – Litz wire
    * 6 – square wire
  - `Phi_hx` Hysteresis lag in degrees in the x-direction for linear problems.
  - `Phi_hy` Hysteresis lag in degrees in the y-direction for linear problems.
  - `nstr` Number of strands in the wire build. Should be 1 for Magnet or Square wire.
  - `dwire` Diameter of each of the wire's constituent strand in millimeters.

Note that not all properties need be defined – properties that aren't defined are assigned default values.

- `mi_addbhpoint('blockname',b,h)` Adds a B-H data point the the material specified by the string `'blockname'`. The point to be added has a flux density of `b` in units of Teslas and a field intensity of `h` in units of Amps/Meter.

- `mi_clearbhpoints('blockname')` Clears all B-H data points associatied with the material specified by `'blockname'`.

- `mi_addpointprop('pointpropname',a,j)` adds a new point property of name `'pointpropname'` with either a specified potential `a` in units Webers/Meter or a point current `j` in units of Amps. Set the unused parameter pairs to 0.

- `mi_addboundprop('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.

- `mi_addcircprop('circuitname', i, circuittype)`
  adds a new circuit property with name `'circuitname'` with a prescribed current. The `circuittype` parameter is 0 for a parallel-connected circuit and 1 for a series-connected circuit.

- `mi_deletematerial('materialname')` deletes the material named `'materialname'`.

- `mi_deleteboundprop('propname')` deletes the boundary property named `'propname'`.

- `mi_deletecircuit('circuitname')` deletes the circuit named `circuitname`.

- `mi_deletepointprop('pointpropname')` deletes the point property named `'pointpropname'`

- `mi_modifymaterial('BlockName',propnum,value)` This function allows for modification of a material's properties without redefining the entire material (*e.g.* so that current can be modified from run to run). The material to be modified is specified by `'BlockName'`.

11

The next parameter is the number of the property to be set. The last number is the value to be applied to the specified property. The various properties that can be modified are listed below:

propnum	Symbol	Description
0	BlockName	Name of the material
1	μ_x	x (or r) direction relative permeability
2	μ_y	y (or z) direction relative permeability
3	H_c	Coercivity, Amps/Meter
4	J	Source current density, MA/m^2
5	σ	Electrical conductivity, MS/m
6	d_{lam}	Lamination thickness, mm
7	ϕ_{hmax}	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	ϕ_{hx}	Hysteresis lag in x-direction for linear problems, degrees
11	ϕ_{hy}	Hysteresis lag in y-direction for linear problems, degrees

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

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

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

propnum	Symbol	Description
0	PointName	Name of the point property
1	*A*	Nodal potential, Weber/Meter
2	*J*	Nodal current, Amps

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

propnum	Symbol	Description
0	CircName	Name of the circuit property
1	*i*	Total current
2	CircType	0 = Parallel, 1 = Series

- `mi_setcurrent('CircName',i)` sets the current in the circuit specified by `'CircName'` to `i`.

4.10   Miscellaneous

- `mi_savebitmap('filename')` saves a bitmapped screenshot of the current view to the file specified by `'filename'`.

- `mi_savemetafile('filename')` saves a metafile screenshot of the current view to the file specified by `'filename'`.

- `mi_refreshview` Redraws the current view.

- `mi_close` Closes current magnetics preprocessor document and destroys magnetics preprocessor window.

- `mi_shownames(flag)` This function allow the user to display or hide the block label names on screen. To hide the block label names, `flag` should be 0. To display the names, the parameter should be set to 1.

- `mi_readdxf('filename')` This function imports a dxf file specified by `'filename'`.

- `mi_savedxf('filename')` This function saves geometry information in a dxf file specified by `'filename'`.

- `mi_defineouterspace(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.

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

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

- `mi_attachdefault()` marks the selected block label as the default block label. This block label is applied to any region that has not been explicitly labeled.

- `mi_detachdefault()` undefines the default attribute for the selected block labels.

- `mi_makeABC(n,R,x,y,bc)` creates a series of circular shells that emulate the impedance of an unbounded domain (i.e. an Inprovised Asymptotic Boundary Condition). The `n` parameter contains the number of shells to be used (should be between 1 and 10), `R` is the radius of the solution domain, and `(x,y)` denotes the center of the solution domain. The `bc` parameter should be specified as 0 for a Dirichlet outer edge or 1 for a Neumann outer edge. If the function is called without all the parameters, the function makes up reasonable values for the missing parameters.

- `mi_setprevious(filename)` defines the previous solution to be used as the basis for an AC incremental permeability solution. The filename should include the `.ans` extension, *e.g.* `mi_setprevious("mymodel.ans")`

5   Magnetics Post Processor Command Set

There are a number of scripting commands designed to operate in the postprocessing environment.

5.1   Data Extraction Commands

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

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

  Returns typically two values. The first value is the result of the integral calculation, and the second value is the average of the quantity of interest over the contour. The only exception is integral 3, which evaluates Maxwell's stress tensor. This integral can return up to four results. For force and torque results, the $2\times$ results are only relevant for problems where $\omega \neq 0$.

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

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

- `mo_getpointvalues(x,y)` Get the values associated with the point at (x,y). The function returns an array whose contents are, in order:

Symbol	Definition
A	Potential A or flux ϕ
B1	B_x if planar, B_r if axisymmetric
B2	B_y if planar, B_z if axisymmetric
Sig	conductivity σ
E	stored energy density
H1	H_x if planar, H_r if axisymmetric
H2	H_y if planar, H_z if axisymmetric
Je	eddy current density
Js	source current density
Mu1	μ_x if planar, μ_r if axisymmetric
Mu2	μ_y if planar, μ_z if axisymmetric
Pe	Power density dissipated through ohmic losses
Ph	Power density dissipated by hysteresis
ff	Winding fill factor

The following series of functions retrieves smaller subsets of these results.

- `mo_geta(x,y)` Get the potential associated with the point at (x,y). For planar problems, the reported potential is vector potential A. For axisymmetric problems, $2\pi r A$ is reported.

- `mo_getb(x,y)` Get the magnetic flux density associated with the point at (x,y). The return value is a list with two element representing B_x and B_y for planar problems and B_r and B_z for axisymmetric problems.

- `mo_getconductivity(x,y)` Gets the conductivity associated with the point at (x,y).

- `mo_getenergydensity(x,y)` Gets the magnetic field energy density associated with the point at (x,y).

- `mo_geth(x,y)` Get the magnetic field intensity associated with the point at (x,y). The return value is a list with two element representing H_x and H_y for planar problems and H_r and H_z for axisymmetric problems.

- `mo_getj(x,y)` Get the electric current density associated with the point at (x,y).

- `mo_getmu(x,y)` Get the relative magnetic permeability associated with the point at (x,y). The return value is a list with two element representing μ_x and μ_y for planar problems and μ_r and μ_z for axisymmetric problems.

- `mo_getpe(x,y)` Get the ohmic loss density associated with the point at (x,y).

- `mo_getph(x,y)` Get the hysteresis/laminated eddy current loss density associated with the point at (x,y).

- `mo_getfill(x,y)` Get the winding factor (*i.e.* the average fraction of the volume filled with conductor) associated with the point at (x,y).

- `mo_makeplot(PlotType,NumPoints,Filename,FileFormat)` Allows Octave 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:

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

16

Valid file formats are

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

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

```
mo_makeplot(2,200)
```

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

```
mo_makeplot(2,200,'c:\temp\myfile.emf')
```

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

```
mo_makeplot(2,200,'c:\temp\myfile.txt',0)
```

- mo_getprobleminfo Returns info on problem description. Returns four values:

Return value	Definition
1	problem type
2	frequency in Hz
3	problem depth in meters
4	length unit used to draw the problem, represented in meters

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

  - current Current carried by the circuit.

  - volts Voltage drop across the circuit in the circuit.

  - flux Circuit's flux linkage

5.2 Selection Commands

- mo_seteditmode(mode) Sets the mode of the postprocessor to point, contour, or area mode. Valid entries for mode are 'point', 'contour', and 'area'.

- mo_selectblock(x,y) Select the block that contains point (x,y).

- mo_groupselectblock(n) Selects all of the blocks that are labeled by block labels that are members of group n. If no number is specified (*i.e.* mo_groupselectblock), all blocks are selected.

- mo_addcontour(x,y) Adds a contour point at (x,y). If this is the first point then it starts a contour, if there are existing points the contour runs from the previous point to this point. The mo_addcontour command has the same functionality as a right-button-click contour point addition when the program is running in interactive mode.

17

- `mo_bendcontour(angle,anglestep)` Replaces the straight line formed by the last two points in the contour by an arc that spans `angle` degrees. The arc is actually composed of many straight lines, each of which is constrained to span no more than `anglestep` degrees. The `angle` parameter can take on values from -180 to 180 degrees. The `anglestep` parameter must be greater than zero. If there are less than two points defined in the contour, this command is ignored.

- `mo_selectpoint(x,y)` Adds a contour point at the closest input point to (x,y). If the selected point and a previous selected points lie at the ends of an arcsegment, a contour is added that traces along the arcsegment. The `mo_selectpoint` command has the same functionality as the left-button-click contour point selection when the program is running in interactive mode.

- `mo_clearcontour` Clear a prevously defined contour

- `mo_clearblock` Clear block selection

5.3   Zoom Commands

- `mo_zoomnatural` Zoom to the natural boundaries of the geometry.

- `mo_zoomin` Zoom in one level.

- `mo_zoomout` Zoom out one level.

- `mo_zoom(x1,y1,x2,y2)` Zoom to the window defined by lower left corner (x1,y1) and upper right corner (x2,y2).

5.4   View Commands

- `mo_showmesh` Show the mesh.

- `mo_hidemesh` Hide the mesh.

- `mo_showpoints` Show the node points from the input geometry.

- `mo_hidepoints` Hide the node points from the input geometry.

- `mo_smooth('flag')` This function controls whether or not smoothing is applied to the B and H fields, which are naturally piece-wise constant over each element. Setting `flag` equal to `'on'` turns on smoothing, and setting `flag` to `'off'` turns off smoothing.

- `mo_showgrid` Show the grid points.

- `mo_hidegrid` Hide the grid points points.

- `mo_grid_snap('flag')` Setting `flag` to 'on' turns on snap to grid, setting `flag` to `'off'` turns off snap to grid.

- `mo_setgrid(density,'type')` Change the grid spacing. The `density` parameter speci-fies the space between grid points, and the `type` parameter is set to `'cart'` for cartesian coordinates or `'polar'` for polar coordinates.

- `mo_hidedensityplot` hides the flux density plot.

- `mo_showdensityplot(legend,gscale,upper_B,lower_B,type)` Shows the flux density plot with options:

  - `legend` Set to `0` to hide the plot legend or `1` to show the plot legend.
  - `gscale` Set to `0` for a colour density plot or `1` for a grey scale density plot.
  - `upper_B` Sets the upper display limit for the density plot.
  - `lower_B` Sets the lower display limit for the density plot.
  - `type` Type of density plot to display. Valid entries are `'mag'`, `'real'`, and `'imag'` for magnitude, real component, and imaginary component of *B*, respectively. Alterna-tively, current density can be displayed by specifying `'jmag'`, `'jreal'`, and `'jimag'` for magnitude, real component, and imaginary component of *J*, respectively.

- `mo_hidecontourplot` Hides the contour plot.

- `mo_showcontourplot(numcontours,lower_A,upper_A,type)` shows the *A* contour plot with options:

  - `numcontours` Number of *A* equipotential lines to be plotted.
  - `upper_A` Upper limit for *A* contours.
  - `lower_A` Lower limit for *A* contours.
  - `type` Choice of `'real'`, `'imag'`, or `'both'` to show either the real, imaginary of both real and imaginary components of A.

- `mo_showvectorplot(type,scalefactor)` controls the display of vectors denoting the field strength and direction. The parameters taken are the `type` of plot, which should be set to 0 for no vector plot, 1 for the real part of flux density B; 2 for the real part of field intensity H; 3 for the imaginary part of B; 4 for the imaginary part of H; 5 for both the real and imaginary parts of B; and 6 for both the real and imaginary parts of H. The `scalefactor` determines the relative length of the vectors. If the scale is set to 1, the length of the vectors are chosen so that the highest flux density corresponds to a vector that is the same length as the current grid size setting.

- `mo_minimize` minimizes the active magnetics output view.

- `mo_maximize` maximizes the active magnetics output view.

- `mo_restore` restores the active magnetics output view from a minimized or maximized state.

- `mo_resize(width,height)` resizes the active magnetics output window client area to width × height.

5.5   Miscellaneous

- `mo_close` Closes the current post-processor instance.

- `mo_refreshview` Redraws the current view.

- `mo_reload` Reloads the solution from disk.

- `mo_savebitmap('filename')` saves a bitmapped screen shot of the current view to the file specified by `'filename'`.

- `mo_savemetafile('filename')` saves a metafile screenshot of the current view to the file specified by `'filename'`.

- `mo_shownames(flag)` This function allow the user to display or hide the block label names on screen. To hide the block label names, `flag` should be 0. To display the names, the parameter should be set to 1.

- `mo_numnodes` Returns the number of nodes in the in focus magnetics output mesh.

- `mo_numelements` Returns the number of elements in the in focus magnets output mesh.

- `mo_getnode(n)` Returns the (x,y) or (r,z) position of the nth mesh node.

- `mo_getelement(n)` MOGetElement[n] returns the following proprerties for the nth element:

  1. Index of first element node
  2. Index of second element node
  3. Index of third element node
  4. x (or r) coordinate of the element centroid
  5. y (or z) coordinate of the element centroid
  6. element area using the length unit defined for the problem
  7. group number associated with the element

6   Electrostatics Preprocessor Command Set

A number of different commands are available in the preprocessor. Two naming conventions can be used: one which separates words in the command names by underscores, and one that eliminates the underscores.

6.1   Object Add/Remove Commands

- `ei_addnode(x,y)` Add a new node at x,y

- `ei_addsegment(x1,y1,x2,y2)` Add a new line segment from node closest to (x1,y1) to node closest to (x2,y2)

- `ei_addblocklabel(x,y)` Add a new block label at (x,y)

- `ei_addarc(x1,y1,x2,y2,angle,maxseg)` Add a new arc segment from the nearest node to (x1,y1) to the nearest node to (x2,y2) with angle 'angle' divided into 'maxseg' segments.

- `ei_drawline(x1,y1,x2,y2)` Adds nodes at (x1,y1) and (x2,y2) and adds a line between the nodes.

- `ei_drawpolyline([x1,y1;x2,y2'...])` Adds nodes at each of the specified points and connects them with segments.

- `ei_drawpolygon([x1,y1;x2,y2'...])` Adds nodes at each of the specified points and connects them with segments to form a closed contour.

- `ei_drawarc(x1,y1,x2,y2,angle,maxseg)` Adds nodes at (x1,y1) and (x2,y2) and adds an arc of the specified angle and discretization connecting the nodes.

- `ei_drawrectangle(x1,y1,x2,y2)` Adds nodes at the corners of a rectangle defined by the points (x1,y1) and (x2,y2), then adds segments connecting the corners of the rectangle.

- `ei_deleteselected` Delete all selected objects.

- `ei_deleteselectednodes` Delete selected nodes.

- `ei_deleteselectedlabels` Delete selected block labels.

- `ei_deleteselectedsegments` Delete selected segments.

- `ei_deleteselectedarcsegments` Delete selects arcs.

6.2   Geometry Selection Commands

- `ei_clearselected` Clear all selected nodes, blocks, segments and arc segments.

- `ei_selectsegment(x,y)` Select the line segment closest to (x,y)

- `ei_selectnode(x,y)` Select the node closest to (x,y). Returns the coordinates of the selected node.

- `ei_selectlabel(x,y)` Select the label closet to (x,y). Returns the coordinates of the selected label.

- `ei_selectarcsegment(x,y)` Select the arc segment closest to (x,y)

- `ei_selectgroup(n)` Select 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)

- `ei_selectcircle(x,y,R,editmode)` selects objects within a circle of radius R centered at (x,y). If only x, y, and R paramters are given, the current edit mode is used. If the editmode parameter is used, 0 denotes nodes, 2 denotes block labels, 2 denotes segments, 3 denotes arcs, and 4 specifies that all entity types are to be selected.

- `ei_selectrectangle(x1,y1,x2,y2,editmode)` selects objects within a rectangle defined by points (x1,y1) and (x2,y2). If no editmode parameter is supplied, the current edit mode is used. If the editmode parameter is used, 0 denotes nodes, 2 denotes block labels, 2 denotes segments, 3 denotes arcs, and 4 specifies that all entity types are to be selected.

6.3 Object Labeling Commands

- `ei_setnodeprop('propname',groupno, 'inconductor')` Set 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>'`.

- `ei_setblockprop('blockname', automesh, meshsize, group)` Set the selected block labels to have the properties:

  Block property `'blockname'`.

  `automesh`: 0 = mesher defers to mesh size constraint defined in `meshsize`, 1 = mesher automatically chooses the mesh density.

  `meshsize`: size constraint on the mesh in the block marked by this label.

  A member of group number `group`

- `ei_setsegmentprop('name', elmsize, automesh, hide, group, 'inconductor')` Set the select segments to have:

  Boundary property `'name'`

  Local element size along segment no greater than `elmsize`

  `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`

  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>'`.

- `ei_setarcsegmentprop(maxsegdeg, 'propname', hide, group, 'inconductor')` Set the selected arc segments to:

  Meshed with elements that span at most `maxsegdeg` degrees per element

  Boundary property `'propname'`

  `hide`: 0 = not hidden in post-processor, 1 == hidden in post processor

  A member of group number `group`

  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>'`.

- `ei_setgroup(n)` Set the group associated of the selected items to n.

6.4   Problem Commands

- ei_probdef(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, entering 1.E-8 requires the RMS of the residual to be less than 10^{-8}. The fourth parameter represents the depth of the problem in the into-the-page direction for 2-D planar problems–just enter 0 as a placeholder for axisymmetric problems. The sixth parameter represents the minimum angle constraint sent to the mesh generator (usually 30 degrees).

- ei_analyze(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.

- ei_loadsolution loads and displays the solution corresponding to the current geometry.

- ei_setfocus('documentname') Switches the electrostatics input file upon which 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.

- ei_saveas('filename') saves the file with name 'filename'.

6.5   Mesh Commands

- ei_createmesh runs triangle to create a mesh. Note that this is not a necessary precursor of performing an analysis, as ei_analyze will make sure the mesh is up to date before running an analysis. The number of elements in the mesh is pushed back onto the lua stack.

- ei_showmesh toggles the flag that shows or hides the mesh.

- ei_purgemesh clears the mesh out of both the screen and memory.

6.6   Editing Commands

- ei_copyrotate(bx, by, angle, copies)

  - bx, by – base point for rotation
  - angle – angle by which the selected objects are incrementally shifted to make each copy. angle is measured in degrees.
  - copies – number of copies to be produced from the selected objects.

- ei_copyrotate2(bx, by, angle, copies, editaction)

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

- `ei_copytranslate(dx, dy, copies)`

  - – `dx,dy` – distance by which the selected objects are incrementally shifted.
  - – `copies` – number of copies to be produced from the selected objects.

- `ei_copytranslate2(dx, dy, copies, editaction)`

  - – `dx,dy` – distance by which the selected objects are incrementally shifted.
  - – `copies` – number of copies to be produced from the selected objects.
  - – `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4- group

- `ei_createradius(x,y,r)` turns a corner located at `(x,y)` into a curve of radius `r`.

- `ei_moverotate(bx,by,shiftangle)`

  - – `bx, by` – base point for rotation
  - – `shiftangle` – angle in degrees by which the selected objects are rotated.

- `ei_moverotate2(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

- `ei_movetranslate(dx,dy)`

  - – `dx,dy` – distance by which the selected objects are shifted.

- `ei_movetranslate2(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

- `ei_scale(bx,by,scalefactor)`

  - – `bx, by` – base point for scaling
  - – `scalefactor` – a multiplier that determines how much the selected objects are scaled

- `ei_scale2(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

- `ei_mirror(x1,y1,x2,y2)` mirror the selected objects about a line passing through the points `(x1,y1)` and `(x2,y2)`.

- `ei_mirror2(x1,y1,x2,y2,editaction)` mirror the selected objects about a line passing through the points `(x1,y1)` and `(x2,y2)`. Valid `editaction` entries are 0 for nodes, 1 for lines (segments), 2 for block labels, 3 for arc segments, and 4 for groups.

- `ei_seteditmode(editmode)` Sets the current editmode to:

  - – `'nodes'` - nodes
  - – `'segments'` - line segments
  - – `'arcsegments'` - arc segments
  - – `'blocks'` - block labels
  - – `'group'` - selected group

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

6.7   Zoom Commands

- `ei_zoomnatural` zooms to a "natural" view with sensible extents.

- `ei_zoomout` zooms out by a factor of 50%.

- `ei_zoomin` zoom in by a factor of 200%.

- `ei_zoom(x1,y1,x2,y2)` Set the display area to be from the bottom left corner specified by `(x1,y1)` to the top right corner specified by `(x2,y2)`.

6.8   View Commands

- `ei_showgrid` Show the grid points.

- `ei_hidegrid` Hide the grid points points.

- `ei_gridsnap('flag')` Setting flag to "on" turns on snap to grid, setting flag to 'off'turns off snap to grid.

- `ei_setgrid(density,'type')` Change the grid spacing. The density parameter specifies the space between grid points, and the type parameter is set to `'cart'` for Cartesian coordinates or `'polar'` for polar coordinates.

- `ei_refreshview` Redraws the current view.

- `ei_minimize` minimizes the active electrostatics input view.

- `ei_maximize` maximizes the active electrostatics input view.

- `ei_restore` restores the active electrostatics input view from a minimized or maximized state.

- `ei_resize(width,height)` resizes the active electrostatics input window client area to width $\times$ height.

6.9 Object Properties

- `ei_getmaterial('materialname')` fetches the material specified by `materialname` from the materials library.

- `ei_addmaterial('matname', ex, ey, qv)` adds a new material with called `'matname'` 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

- `ei_addpointprop('pointname',Vp,qp)` adds a new point property of name `'pointname'` with either a specified potential `Vp` a point charge density `qp` in units of C/m. .

- `ei_addboundprop('boundname', Vs, qs, c0, c1, BdryFormat)` adds a new boundary property with name `'boundname'`

  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 `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/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.

- `ei_addconductorprop('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.

- `ei_deletematerial('materialname')` deletes the material named `'materialname'`.

- `ei_deleteboundprop('boundname')` deletes the boundary property named `'boundname'`.

- ei_deleteconductor('conductorname') deletes the conductor named conductorname.

- ei_deletepointprop('pointname') deletes the point property named 'pointname'

- ei_modifymaterial('BlockName',propnum,value) This function allows for modification of a material's properties without redefining the entire material (e.g. so that current can be modified from run to run). The material to be modified is specified by 'BlockName'. The next parameter is the number of the property to be set. The last number is the value to be applied to the specified property. The various properties that can be modified are listed below:

propnum	Symbol	Description
0	BlockName	Name of the material
1	ex	x (or r) direction relative permittivity
2	ey	y (or z) direction relative permittivity
3	qs	Volume charge

- ei_modifyboundprop('BdryName',propnum,value) This function allows for modification of a boundary property. The BC to be modified is specified by 'BdryName'. The next parameter is the number of the property to be set. The last number is the value to be applied to the specified property. The various properties that can be modified are listed below:

propnum	Symbol	Description
0	BdryName	Name of boundary property
1	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

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

propnum	Symbol	Description
0	PointName	Name of the point property
1	Vp	Prescribed nodal voltage
2	qp	Point charge density in C/m

- ei_modifyconductorprop('ConductorName',propnum,value) This function 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:

propnum	Symbol	Description
0	ConductorName	Name of the conductor property
1	Vc	Conductor voltage
2	qc	Total conductor charge
3	ConductorType	0 = Prescribed charge, 1 = Prescribed voltage

6.10   Miscellaneous

- `ei_savebitmap('filename')` saves a bitmapped screenshot of the current view to the file specified by `'filename'`.

- `ei_savemetafile('filename')` saves a metafile screenshot of the current view to the file specified by `'filename'`.

- `ei_refreshview` Redraws the current view.

- `ei_close` closes the preprocessor window and destroys the current document.

- `ei_shownames(flag)` This function allow the user to display or hide the block label names on screen. To hide the block label names, `flag` should be 0. To display the names, the parameter should be set to 1.

- `ei_readdxf('filename')` This function imports a dxf file specified by `'filename'`.

- `ei_savedxf('filename')` This function saves geometry information in a dxf file specified by `'filename'`.

- `ei_defineouterspace(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.

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

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

- `ei_attachdefault()` marks the selected block label as the default block label. This block label is applied to any region that has not been explicitly labeled.

- `ei_detachdefault()` undefines the default attribute for the selected block labels.

- `ei_makeABC(n,R,x,y,bc)` creates a series of circular shells that emulate the impedance of an unbounded domain (i.e. an Inprovised Asymptotic Boundary Condition). The `n` parameter contains the number of shells to be used (should be between 1 and 10), `R` is the radius of the solution domain, and `(x,y)` denotes the center of the solution domain. The `bc` parameter should be specified as 0 for a Dirichlet outer edge or 1 for a Neumann outer edge. If the function is called without all the parameters, the function makes up reasonablevaluesforthemissingparameters.

7 Electrostatics Post Processor Command Set

There are a number of scripting commands designed to operate in the postprocessor. As with the preprocessor commands, these commands can be used with either the underscore naming or with the no-underscore naming convention.

7.1 Data Extraction Commands

- `eo_lineintegral(type)` Calculates the line integral for the defined contour

type	Integral
0	$E \cdot t$
1	$D \cdot n$
2	Contour length
3	Force from stress tensor
4	Torque from stress tensor

This integral a single result for field intensity, flux density, contour length, and torque. For force, an array with two elements is returned representing force in the x and y or r and z directions.

- `eo_blockintegral(type)` Calculates a block integral for the selected blocks

type	Integral
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

This integral a single result for energy, volume and torque. For force, D and E, the function returns an array with two elements is returned representing components in the x and y or r and z directions.

- `eo_getpointvalues(x,y)` Gets the values associated with the point at (x,y). The function returns an array of results whose elements represent:

Symbol	Definition
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

- `eo_getv(x,y)` Gets the voltage associated with the point at (x,y).

- `eo_getd(x,y)` Gets the electric flux density associated with the point at (x,y). The return value is a list with two element representing D_x and D_y for planar problems and D_r and D_z for axisymmetric problems.

- `eo_gete(x,y)` Gets the electric field intensity associated with the point at (x,y). The return value is a list with two element representing E_x and E_y for planar problems and E_r and E_z for axisymmetric problems.

- `eo_getenergydensity(x,y)` Gets the electric field energy density associated with the point at (x,y).

- `eo_getperm(x,y)` Gets the electric permittivity associated with the point at (x,y). The return value is a list with two element representing ε_x and ε_y for planar problems and ε_r and ε_z for axisymmetric problems.

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

PlotType	Definition
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)

Valid file formats are:

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

For example, if one wanted to plot *V* to the screen with 200 points evaluated to make the graph, the command would be:

`eo_makeplot(0,200)`

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

`eo_makeplot(0,200,'c:temp.emf')`

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

`eo_makeplot(0,200,'c:temp.txt',0)`

- `eo_getprobleminfo` Returns info on problem description. Returns three values: the Problem type (0 for planar and 1 for axisymmetric); the depth assumed for planar problems in units of meters; and the length unit used to draw the problem, represented in meters.

- `eo_getconductorproperties('conductor')` Properties are returned for the conductor property named 'conductor'. Two values are returned: The voltage of the specified conductor, and the charge carried on the specified conductor.

7.2 Selection Commands

- `eo_seteditmode(mode)` Sets the mode of the postprocessor to point, contour, or area mode. Valid entries for mode are `'point'`, `'contour'`, and `'area'`.

- `eo_selectblock(x,y)` Select the block that contains point `(x,y)`.

- `eo_groupselectblock(n)` Selects all of the blocks that are labeled by block labels that are members of group n. If no number is specified (*i.e.* `eo_groupselectblock`), all blocks are selected.

- `eo_selectconductor('name')` Selects all nodes, segments, and arc segments that are part of the conductor specified by the string (`'name'`). This command is used to select conductors for the purposes of the "weighted stress tensor" force and torque integrals, where the conductors are points or surfaces, rather than regions (*i.e.* can't be selected with `eo_selectblock`).

- `eo_addcontour(x,y)` Adds a contour point at `(x,y)`. If this is the first point then it starts a contour, if there are existing points the contour runs from the previous point to this point. The `eo_addcontour` command has the same functionality as a right-button-click contour point addition when the program is running in interactive mode.

- `eo_bendcontour(angle,anglestep)` Replaces the straight line formed by the last two points in the contour by an arc that spans angle degrees. The arc is actually composed of many straight lines, each of which is constrained to span no more than anglestep degrees. The `angle` parameter can take on values from -180 to 180 degrees. The `anglestep` parameter must be greater than zero. If there are less than two points defined in the contour, this command is ignored.

- `eo_selectpoint(x,y)` Adds a contour point at the closest input point to `(x,y)`. If the selected point and a previous selected points lie at the ends of an arcsegment, a contour is added that traces along the arcsegment. The `selectpoint` command has the same functionality as the left-button-click contour point selection when the program is running in interactive mode.

- `eo_clearcontour` Clear a prevously defined contour

- `eo_clearblock` Clear block selection

7.3 Zoom Commands

- `eo_zoomnatural` Zoom to the natural boundaries of the geometry.

- `eo_zoomin` Zoom in one level.

- `eo_zoomout` Zoom out one level.

- `eo_zoom(x1,y1,x2,y2)` Zoom to the window defined by lower left corner `(x1,y1)` and upper right corner `(x2,y2)`.

7.4 View Commands

- `eo_showmesh` Show the mesh.

- `eo_hidemesh` Hide the mesh.

- `eo_showpoints` Show the node points from the input geometry.

- `eo_hidepoints` Hide the node points from the input geometry.

- `eo_smooth('flag')` This function controls whether or not smoothing is applied to the D and E fields which are naturally piece-wise constant over each element. Setting flag equal to `'on'` turns on smoothing, and setting flag to `'off'` turns off smoothing.

- `eo_showgrid` Show the grid points.

- `eo_hidegrid` Hide the grid points points.

  `eo_gridsnap('flag')` Setting flag to "on" turns on snap to grid, setting flag to 'off' turns off snap to grid.

- `eo_setgrid(density,'type')` Change the grid spacing. The density parameter specifies the space between grid points, and the type parameter is set to `'cart'` for Cartesian coordinates or `'polar'` for polar coordinates.

- `eo_hidedensityplot` hides the flux density plot.

- `eo_showdensityplot(legend,gscale,type,upper_D,lower_D)` 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_D` Sets the upper display limit for the density plot.

  `lower_D` 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

- `eo_hidecontourplot` Hides the contour plot.

- `eo_showcontourplot(numcontours,lower_V,upper_V)` shows the V contour plot with options:

  `numcontours` Number of equipotential lines to be plotted.

  `upper_V` Upper limit for contours.

  `lower_V` Lower limit for contours.

- `eo_showvectorplot(type,scalefactor)` controls the display of vectors denoting the field strength and direction. The parameters taken are the `type` of plot, which should be set to 0 for no vector plot, 1 for flux density D, and 2 for field intensity E. The `scalefactor` determines the relative length of the vectors. If the scale is set to 1, the length of the vectors are chosen so that the highest flux density corresponds to a vector that is the same length as the current grid size setting.

- `eo_minimize` minimizes the active electrostatics output view.

- `eo_maximize` maximizes the active electrostatics output view.

- `eo_restore` restores the active electrostatics output view from a minimized or maximized state.

- `eo_resize(width,height)` resizes the active electrostatics output window client area to width $\times$ height.

7.5 Miscellaneous

- `eo_close` close the current postprocessor window.

- `eo_refreshview` Redraws the current view.

- `eo_reload` Reloads the solution from disk.

- `eo_savebitmap('filename')` saves a bitmapped screen shot of the current view to the file specified by `'filename'`.

- `eo_savemetafile('filename')` saves a metafile screenshot of the current view to the file specified by `'filename'`.

- `eo_shownames(flag)` This function allow the user to display or hide the block label names on screen. To hide the block label names, `flag` should be 0. To display the names, the parameter should be set to 1.

- `eo_numnodes` Returns the number of nodes in the in focus electrostatics output mesh.

- `eo_numelements` Returns the number of elements in the in focus electrostatics output mesh.

- `eo_getnode(n)` Returns the (x,y) or (r,z) position of the nth mesh node.

- `eo_getelement(n)` MOGetElement[n] returns the following proprerties for the nth element:

  1. Index of first element node
  2. Index of second element node
  3. Index of third element node
  4. x (or r) coordinate of the element centroid
  5. y (or z) coordinate of the element centroid
  6. element area using the length unit defined for the problem
  7. group number associated with the element

8   Heat Flow Preprocessor Lua Command Set

A number of different commands are available in the preprocessor. Two naming conventions can be used: one which separates words in the command names by underscores, and one that eliminates the underscores.

8.1   Object Add/Remove Commands

- `hi_addnode(x,y)` Add a new node at x,y

- `hi_addsegment(x1,y1,x2,y2)` Add a new line segment from node closest to (x1,y1) to node closest to (x2,y2)

- `hi_addblocklabel(x,y)` Add a new block label at (x,y)

- `hi_addarc(x1,y1,x2,y2,angle,maxseg)` Add a new arc segment from the nearest node to (x1,y1) to the nearest node to (x2,y2) with angle 'angle' divided into 'maxseg' segments.

- `hi_deleteselected` Delete all selected objects.

- `hi_deleteselectednodes` Delete selected nodes.

- `hi_deleteselectedlabels` Delete selected block labels.

- `hi_deleteselectedsegments` Delete selected segments.

- `hi_deleteselectedarcsegments` Delete selects arcs.

8.2   Geometry Selection Commands

- `hi_clearselected()` Clear all selected nodes, blocks, segments and arc segments.

- `hi_selectsegment(x,y)` Select the line segment closest to (x,y)

- `hi_selectnode(x,y)` Select the node closest to (x,y). Returns the coordinates of the selected node.

- `hi_selectlabel(x,y)` Select the label closet to (x,y). Returns the coordinates of the selected label.

- `hi_selectarcsegment(x,y)` Select the arc segment closest to (x,y)

- `hi_selectgroup(n)` Select 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)

- `hi_selectcircle(x,y,R,editmode)` selects objects within a circle of radius R centered at (x,y). If only x, y, and R paramters are given, the current edit mode is used. If the editmode parameter is used, 0 denotes nodes, 2 denotes block labels, 2 denotes segments, 3 denotes arcs, and 4 specifies that all entity types are to be selected.

- `hi_selectrectangle(x1,y1,x2,y2,editmode)` selects objects within a rectangle defined by points (x1,y1) and (x2,y2). If no editmode parameter is supplied, the current edit mode is used. If the editmode parameter is used, 0 denotes nodes, 2 denotes block labels, 2 denotes segments, 3 denotes arcs, and 4 specifies that all entity types are to be selected.

8.3   Object Labeling Commands

- `hi_setnodeprop("propname",groupno, "inconductor")` Set 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>"`.

- `hi_setblockprop("blockname", automesh, meshsize, group)` Set the selected block labels to have the properties:

  Block property `"blockname"`.

  `automesh`: 0 = mesher defers to mesh size constraint defined in `meshsize`, 1 = mesher automatically chooses the mesh density.

  `meshsize`: size constraint on the mesh in the block marked by this label.

  A member of group number `group`

- `hi_setsegmentprop("propname", elementsize, automesh, hide, group, "inconductor")`
  Set 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 `group`

  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>"`.

- `hi_setarcsegmentprop(maxsegdeg, "propname", hide, group, "inconductor")` Set the selected arc segments to:

  Meshed with elements that span at most `maxsegdeg` degrees per element

  Boundary property `"propname"`

  `hide`: 0 = not hidden in post-processor, 1 == hidden in post processor

  A member of group number `group`

  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>"`.

- `hi_setgroup(n)` Set the group associated of the selected items to n.

8.4   Problem Commands

- `hi_probdef(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, entering `1.E-8` requires the RMS of the residual to be less than 10^{-8}. A fourth parameter, representing the depth of the problem in the into-the-page direction for 2-D planar problems, can also be specified for planar problems. A sixth parameter represents the minimum angle constraint sent to the mesh generator.

- `hi_analyze(flag)` runs `hsolv` to solve the problem. The `flag` parameter controls whether the hsolve 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.

- `hi_loadsolution()` loads and displays the solution corresponding to the current geometry.

- `hi_setfocus(``documentname'')` Switches the heat flow input file upon which Lua commands are to act. If more than one heat flow input file is being edited at a time, this command

36

can be used to switch between files so that the mutiple files can be operated upon program-matically via Lua. `documentname` should contain the name of the desired document as it appears on the window's title bar.

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

8.5 Mesh Commands

- `hi_createmesh()` runs triangle to create a mesh. Note that this is not a necessary precursor of performing an analysis, as `hi_analyze()` will make sure the mesh is up to date before running an analysis. The number of elements in the mesh is pushed back onto the lua stack.

- `hi_showmesh()` toggles the flag that shows or hides the mesh.

- `hi_purgemesh()` clears the mesh out of both the screen and memory.

8.6 Editing Commands

- `hi_copyrotate(bx, by, angle, copies, (editaction) )`

  `bx, by` – base point for rotation

  `angle` – angle by which the selected objects are incrementally shifted to make each copy. `angle` is measured in degrees.

  `copies` – number of copies to be produced from the selected objects.

  `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4- group

- `hi_copytranslate(dx, dy, copies, (editaction))`

  `dx,dy` – distance by which the selected objects are incrementally shifted.

  `copies` – number of copies to be produced from the selected objects.

  `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4- group

- `mi_createradius(x,y,r)turnsacornerlocatedat(x,y)intoacurveofradiusr`.

- `hi_moverotate(bx,by,shiftangle (editaction))`

  `bx, by` – base point for rotation

  `shiftangle` – angle in degrees by which the selected objects are rotated.

  `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4- group

- `hi_movetranslate(dx,dy,(editaction))`

  `dx,dy` – distance by which the selected objects are shifted.

  `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4- group

- `hi_scale(bx,by,scalefactor,(editaction))`

  `bx, by` – base point for scaling

  `scalefactor` – a multiplier that determines how much the selected objects are scaled

  `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4- group

- `hi_mirror(x1,y1,x2,y2,(editaction))` mirror the selected objects about a line passing through the points `(x1,y1)` and `(x2,y2)`. Valid `editaction` entries are 0 for nodes, 1 for lines (segments), 2 for block labels, 3 for arc segments, and 4 for groups.

- `hi_seteditmode(editmode)` Sets the current editmode to:

  `"nodes"` – nodes

  `"segments"` - line segments

  `"arcsegments"` - arc segments

  `"blocks"` - block labels

  `"group"` - selected group

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

8.7 Zoom Commands

- `hi_zoomnatural()` zooms to a "natural" view with sensible extents.

- `hi_zoomout()` zooms out by a factor of 50%.

- `hi_zoomin()` zoom in by a factor of 200%.

- `hi_zoom(x1,y1,x2,y2)` Set the display area to be from the bottom left corner specified by `(x1,y1)` to the top right corner specified by `(x2,y2)`.

8.8 View Commands

- `hi_showgrid()` Show the grid points.

- `hi_hidegrid()` Hide the grid points points.

- `hi_gridsnap("flag")` Setting flag to "on" turns on snap to grid, setting flag to "off"turns off snap to grid.

- `hi_setgrid(density,"type")` Change the grid spacing. The density parameter specifies the space between grid points, and the type parameter is set to `"cart"` for Cartesian coordinates or `"polar"` for polar coordinates.

- `hi_refreshview()` Redraws the current view.

- `hi_minimize()` minimizes the active heat flow input view.

- `hi_maximize()` maximizes the active heat flow input view.

- `hi_restore()` restores the active heat flow input view from a minimized or maximized state.

- `hi_resize(width,height)` resizes the active heat flow input window client area to width × height.

8.9 Object Properties

- `hi_getmaterial('materialname')` fetches the material specified by `materialname` from the materials library.

- `hi_addmaterial("materialname", kx, ky, qv, kt)` 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 W/m^3

  kt Volumetric heat capacity in units of MJ/(m^3*K)

- `hi_addpointprop("pointpropname",Tp,qp)` adds a new point property of name `"pointpropname"` with either a specified temperature `Tp` or a point heat generation density `qp` in units of W/m. .

- `hi_addboundprop("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 emissivity 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.

- `hi_addconductorprop("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 heat flux and 1 for prescribed temperature.

- `hi_deletematerial("materialname")` deletes the material named `"materialname"`.

- `hi_deleteboundprop("boundpropname")` deletes the boundary property named `"boundpropname"`.

- `hi_deleteconductor("conductorname")` deletes the conductor named `conductorname`.

- `hi_deletepointprop("pointpropname")` deletes the point property named `"pointpropname"`

- `hi_modifymaterial("BlockName",propnum,value)` This function allows for modification of a material's properties without redefining the entire material (e.g. so that current can be modified from run to run). The material to be modified is specified by `"BlockName"`. The next parameter is the number of the property to be set. The last number is the value to be applied to the specified property. The various properties that can be modified are listed below:

propnum	Symbol	Description
0	BlockName	Name of the material
1	kx	x (or r) direction thermal conductivity
2	ky	y (or z) direction thermal conductivity
3	qs	Volume heat generation
4	kt	Volumetric heat capacity

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

propnum	Symbol	Description		
0	BdryName	Name of boundary property		
1	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	Emissivity		

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

propnum	Symbol	Description
0	PointName	Name of the point property
1	Tp	Prescribed nodal temperature
2	qp	Point heat generation in W/m

- hi_modifyconductorprop("ConductorName",propnum,value) This function 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:

propnum	Symbol	Description
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

- hi_addtkpoint("materialname",T,k) adds the point (T,k) to the thermal conductivity vs. temperature curve for the material specified by "materialname".

- hi_cleartkpoints("materialname") erases all of the thermal conductivity points that have been defined for the material named "materialname".

8.10 Miscellaneous

- hi_savebitmap("filename") saves a bitmapped screenshot of the current view to the file specified by "filename", subject to the printf-type formatting explained previously for the hi_saveas command.

- hi_savemetafile("filename") saves a metafile screenshot of the current view to the file specified by "filename", subject to the printf-type formatting explained previously for the hi_saveas command.

- hi_refreshview() Redraws the current view.

- hi_close() closes the preprocessor window and destroys the current document.

- hi_shownames(flag) This function allow the user to display or hide the block label names on screen. To hide the block label names, flag should be 0. To display the names, the parameter should be set to 1.

- hi_readdxf("filename") This function imports a dxf file specified by "filename".

- hi_savedxf("filename") This function saves geometry information in a dxf file specified by "filename".

- hi_defineouterspace(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.

- `hi_attachouterspace()` marks all selected block labels as members of the external region used for modeling unbounded axisymmetric problems via the Kelvin Transformation.

- `hi_detachouterspace()` undefines all selected block labels as members of the external region used for modeling unbounded axisymmetric problems via the Kelvin Transformation.

- `hi_attachdefault()` marks the selected block label as the default block label. This block label is applied to any region that has not been explicitly labeled.

- `hi_detachdefault()` undefines the default attribute for the selected block labels.

- `hi_makeABC(n,R,x,y,bc)` creates a series of circular shells that emulate the impedance of an unbounded domain (i.e. an Inprovised Asymptotic Boundary Condition). The `n` parameter contains the number of shells to be used (should be between 1 and 10), `R` is the radius of the solution domain, and `(x,y)` denotes the center of the solution domain. The `bc` parameter should be specified as 0 for a Dirichlet outer edge or 1 for a Neumann outer edge. If the function is called without all the parameters, the function makes up reasonablevaluesforthemissingparameters.

9 Heat Flow Post Processor Command Set

There are a number of Lua scripting commands designed to operate in the postprocessor. As with the preprocessor commands, these commands can be used with either the underscore naming or with the no-underscore naming convention.

9.1 Data Extraction Commands

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

type	Integral
0	Temperature difference ($G \cdot t$)
1	Heat flux through the contour ($F \cdot n$)
2	Contour length
3	Average Temperature

  This integral returns either 1 or 2 values, depending on the integral type, *e.g.*:

  `Ftot, Favg = ho_lineintegral(2)`

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

type	Integral
0	Average *T* over the block
1	Block Cross-section
2	Block Volume
3	Average *F* over the block
4	Average *G* over the block

Returns one or two floating point values as results, *e.g.*:

```
Gx, Gy = ho_blockintegral(4)
```

- `ho_getpointvalues(X,Y)` Get the values associated with the point at x,y The return values, in order, are:

Symbol	Definition
V	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

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

```
T,Fx,Fy,Gx,Gy,kx,ky= ho_getpointvalues(0.01,0)
```

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

PlotType	Definition
0	V (Temperature)
1	|D| (Magnitude of heat flux density)
2	D . n (Normal heat flux density)
3	D . t (Tangential heat flux density)
4	|E| (Magnitude of field intensity)
5	E . n (Normal field intensity)
6	E . t (Tangential field intensity)

Valid file formats are:

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

For example, if one wanted to plot *V* to the screen with 200 points evaluated to make the graph, the command would be:

`ho_makeplot(0,200)`

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

`ho_makeplot(0,200,"c:temp.emf")`

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

`ho_makeplot(0,200,"c:temp.txt",0)`

- `ho_getprobleminfo()` Returns info on problem description. Returns three values: the Problem type (0 for planar and 1 for axisymmetric); the depth assumed for planar problems in units of meters; and the length unit used to draw the problem, represented in meters.

- `ho_getconductorproperties("conductor")` Properties are returned for the conductor property named "conductor". Two values are returned: The temperature of the specified conductor, and the total heat flux through the specified conductor.

9.2 Selection Commands

- `ho_seteditmode(mode)` Sets the mode of the postprocessor to point, contour, or area mode. Valid entries for mode are `"point"`, `"contour"`, and `"area"`.

- `ho_selectblock(x,y)` Select the block that contains point `(x,y)`.

- `ho_groupselectblock(n)` Selects all of the blocks that are labeled by block labels that are members of group n. If no number is specified (*i.e.* `ho_groupselectblock()`), all blocks are selected.

- `ho_selectconductor("name")` Selects all nodes, segments, and arc segments that are part of the conductor specified by the string (`"name"`). This command is used to select conductors for the purposes of the "weighted stress tensor" force and torque integrals, where the conductors are points or surfaces, rather than regions (*i.e.* can't be selected with `ho_selectblock`).

- `ho_addcontour(x,y)` Adds a contour point at `(x,y)`. If this is the first point then it starts a contour, if there are existing points the contour runs from the previous point to this point. The `ho_addcontour` command has the same functionality as a right-button-click contour point addition when the program is running in interactive mode.

- `ho_bendcontour(angle,anglestep)` Replaces the straight line formed by the last two points in the contour by an arc that spans angle degrees. The arc is actually composed of many straight lines, each of which is constrained to span no more than anglestep degrees. The `angle` parameter can take on values from -180 to 180 degrees. The `anglestep` parameter must be greater than zero. If there are less than two points defined in the contour, this command is ignored.

- `ho_selectpoint(x,y)` Adds a contour point at the closest input point to `(x,y)`. If the selected point and a previous selected points lie at the ends of an arcsegment, a contour is added

that traces along the arcsegment. The `selectpoint` command has the same functionality as the left-button-click contour point selection when the program is running in interactive mode.

- `ho_clearcontour()` Clear a prevously defined contour

- `ho_clearblock()` Clear block selection

9.3   Zoom Commands

- `ho_zoomnatural()` Zoom to the natural boundaries of the geometry.

- `ho_zoomin()` Zoom in one level.

- `ho_zoomout()` Zoom out one level.

- `ho_zoom(x1,y1,x2,y2)` Zoom to the window defined by lower left corner `(x1,y1)` and upper right corner `(x2,y2)`.

9.4   View Commands

- `ho_showmesh()` Show the mesh.

- `ho_hidemesh()` Hide the mesh.

- `ho_showpoints()` Show the node points from the input geometry.

- `ho_hidepoints()` Hide the node points from the input geometry.

- `ho_smooth("flag")` This function controls whether or not smoothing is applied to the *F* and *G* fields which are naturally piece-wise constant over each element. Setting flag equal to `"on"` turns on smoothing, and setting flag to `"off"` turns off smoothing.

- `ho_showgrid()` Show the grid points.

- `ho_hidegrid()` Hide the grid points points.

  `ho_gridsnap("flag")` Setting flag to "on" turns on snap to grid, setting flag to "off" turns off snap to grid.

- `ho_setgrid(density,"type")` Change the grid spacing. The density parameter specifies the space between grid points, and the type parameter is set to `"cart"` for Cartesian coordinates or `"polar"` for polar coordinates.

- `ho_hidedensityplot()` hides the heat flux density plot.

- `ho_showdensityplot(legend,gscale,type,upper,lower)` Shows the heat 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. A value of 0 plots temperature, 1 plots the magnitude of F, and 2 plots the magnitude of G

- ho_hidecontourplot() Hides the contour plot.

- ho_showcontourplot(numcontours,lower_V,upper_V) shows the V contour plot with options:

  numcontours Number of equipotential lines to be plotted.

  upper_V Upper limit for contours.

  lower_V Lower limit for contours.

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

- ho_showvectorplot(type,scalefactor) controls the display of vectors denoting the field strength and direction. The parameters taken are the type of plot, which should be set to 0 for no vector plot, 1 for heat flux density F, and 2 for temperature gradient G. The scalefactor determines the relative length of the vectors. If the scale is set to 1, the length of the vectors are chosen so that the highest flux density corresponds to a vector that is the same length as the current grid size setting.

- ho_minimize() minimizes the active heat flow input view.

- ho_maximize() maximizes the active heat flow input view.

- ho_restore() restores the active heat flow input view from a minimized or maximized state.

- ho_resize(width,height) resizes the active heat flow input window client area to width $\times$ height.

9.5   Miscellaneous

- ho_close() close the current postprocessor window.

- ho_refreshview() Redraws the current view.

- ho_reload() Reloads the solution from disk.

- ho_savebitmap("filename") saves a bitmapped screen shot of the current view to the file specified by "filename". Note that if you use a path you must use two backslashes (e.g. "c:\\temp\\myfile.bmp"). If the file name contains a space (e.g. file names like c:\program files\stuff) you must enclose the file name in (extra) quotes by using a \" sequence. For example:

  ho_savebitmap("\"c:\\temp\\screenshot.bmp\"")

- `ho_savemetafile("filename")` saves a metafile screenshot of the current view to the file specified by `"filename"`, subject to the printf-type formatting explained previously for the `savebitmap` command.

- `ho_shownames(flag)` This function allow the user to display or hide the block label names on screen. To hide the block label names, `flag` should be 0. To display the names, the parameter should be set to 1.

- `ho_numnodes` Returns the number of nodes in the in focus heat flow output mesh.

- `ho_numelements` Returns the number of elements in the in focus heat flow output mesh.

- `ho_getnode(n)` Returns the (x,y) or (r,z) position of the nth mesh node.

- `ho_getelement(n)` MOGetElement[n] returns the following proprerties for the nth element:

  1. Index of first element node
  2. Index of second element node
  3. Index of third element node
  4. x (or r) coordinate of the element centroid
  5. y (or z) coordinate of the element centroid
  6. element area using the length unit defined for the problem
  7. group number associated with the element

10 Current Flow Preprocessor Lua Command Set

A number of different commands are available in the preprocessor. Two naming conventions can be used: one which separates words in the command names by underscores, and one that eliminates the underscores.

10.1 Object Add/Remove Commands

- `ci_addnode(x,y)` Add a new node at x,y

- `ci_addsegment(x1,y1,x2,y2)` Add a new line segment from node closest to (x1,y1) to node closest to (x2,y2)

- `ci_addblocklabel(x,y)` Add a new block label at (x,y)

- `ci_addarc(x1,y1,x2,y2,angle,maxseg)` Add a new arc segment from the nearest node to (x1,y1) to the nearest node to (x2,y2) with angle 'angle' divided into 'maxseg' segments.

- `ci_deleteselected` Delete all selected objects.

- `ci_deleteselectednodes` Delete selected nodes.

- `ci_deleteselectedlabels` Delete selected block labels.

- `ci_deleteselectedsegments` Delete selected segments.

- `ci_deleteselectedarcsegments` Delete selects arcs.

10.2   Geometry Selection Commands

- `ci_clearselected()` Clear all selected nodes, blocks, segments and arc segments.

- `ci_selectsegment(x,y)` Select the line segment closest to (x,y)

- `ci_selectnode(x,y)` Select the node closest to (x,y). Returns the coordinates of the selected node.

- `ci_selectlabel(x,y)` Select the label closet to (x,y). Returns the coordinates of the selected label.

- `ci_selectarcsegment(x,y)` Select the arc segment closest to (x,y)

- `ci_selectgroup(n)` Select 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)

- `ci_selectcircle(x,y,R,editmode)` selects objects within a circle of radius R centered at (x,y). If only x, y, and R paramters are given, the current edit mode is used. If the editmode parameter is used, 0 denotes nodes, 2 denotes block labels, 2 denotes segments, 3 denotes arcs, and 4 specifies that all entity types are to be selected.

- `ci_selectrectangle(x1,y1,x2,y2,editmode)` selects objects within a rectangle defined by points (x1,y1) and (x2,y2). If no editmode parameter is supplied, the current edit mode is used. If the editmode parameter is used, 0 denotes nodes, 2 denotes block labels, 2 denotes segments, 3 denotes arcs, and 4 specifies that all entity types are to be selected.

10.3   Object Labeling Commands

- `ci_setnodeprop("propname",groupno, "inconductor")` Set 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>"`.

- `ci_setblockprop("blockname", automesh, meshsize, group)` Set the selected block labels to have the properties:

  Block property `"blockname"`.

  `automesh`: 0 = mesher defers to mesh size constraint defined in `meshsize`, 1 = mesher automatically chooses the mesh density.

  `meshsize`: size constraint on the mesh in the block marked by this label.

  A member of group number `group`

- `ci_setsegmentprop("propname", elementsize, automesh, hide, group, "inconductor",)`
  Set 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 `group`

  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>"`.

- `ci_setarcsegmentprop(maxsegdeg, "propname", hide, group, "inconductor")` Set the selected arc segments to:

  Meshed with elements that span at most `maxsegdeg` degrees per element

  Boundary property `"propname"`

  `hide`: 0 = not hidden in post-processor, 1 == hidden in post processor

  A member of group number `group`

  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>"`.

- `ci_setgroup(n)` Set the group associated of the selected items to n.

10.4   Problem Commands

- `ci_probdef(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 specifies the frequency in Hz at which the analysis ois to be performed. The `precision` parameter dictates the precision required by the solver. For example, entering `1.E-8` requires the RMS of the residual to be less than 10^{-8}. A fourth parameter, representing the depth of the problem in the into-the-page direction for 2-D planar problems, can also be specified for planar problems. A sixth parameter represents the minimum angle constraint sent to the mesh generator.

- `ci_analyze(flag)` runs `belasolv` to solve the problem. The `flag` parameter controls whether the Belasolve 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.

- `ci_loadsolution()` loads and displays the solution corresponding to the current geometry.

- `ci_setfocus("documentname")` Switches the electrostatics input file upon which Lua 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 via Lua. `documentname` should contain the name of the desired document as it appears on the window's title bar.

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

10.5   Mesh Commands

- `ci_createmesh()` runs triangle to create a mesh. Note that this is not a necessary precursor of performing an analysis, as `ci_analyze()` will make sure the mesh is up to date before running an analysis. The number of elements in the mesh is pushed back onto the lua stack.

- `ci_showmesh()` toggles the flag that shows or hides the mesh.

- `ci_purgemesh()` clears the mesh out of both the screen and memory.

10.6   Editing Commands

- `ci_copyrotate(bx, by, angle, copies, (editaction) )`

  `bx, by` – base point for rotation

  `angle` – angle by which the selected objects are incrementally shifted to make each copy. `angle` is measured in degrees.

  `copies` – number of copies to be produced from the selected objects.

  `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4- group

- `ci_copytranslate(dx, dy, copies, (editaction))`

  `dx,dy` – distance by which the selected objects are incrementally shifted.

  `copies` – number of copies to be produced from the selected objects.

  `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4- group

- `mi_createradius(x,y,r)` turns a corner located at (`x,y`) into a curve of radius `r`.

- `ci_moverotate(bx,by,shiftangle (editaction))`

  `bx, by` – base point for rotation

  `shiftangle` – angle in degrees by which the selected objects are rotated.

  `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4- group

- `ci_movetranslate(dx,dy,(editaction))`

  `dx,dy` – distance by which the selected objects are shifted.

  `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4- group

- `ci_scale(bx,by,scalefactor,(editaction))`

  `bx, by` – base point for scaling

  `scalefactor` – a multiplier that determines how much the selected objects are scaled

  `editaction` 0 –nodes, 1 – lines (segments), 2 –block labels, 3 – arc segments, 4- group

- `ci_mirror(x1,y1,x2,y2,(editaction))` mirror the selected objects about a line passing through the points `(x1,y1)` and `(x2,y2)`. Valid `editaction` entries are 0 for nodes, 1 for lines (segments), 2 for block labels, 3 for arc segments, and 4 for groups.

- `ci_seteditmode(editmode)` Sets the current editmode to:

  `"nodes"` – nodes

  `"segments"` - line segments

  `"arcsegments"` - arc segments

  `"blocks"` - block labels

  `"group"` - selected group

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

10.7   Zoom Commands

- `ci_zoomnatural()` zooms to a "natural" view with sensible extents.

- `ci_zoomout()` zooms out by a factor of 50%.

- `ci_zoomin()` zoom in by a factor of 200%.

- `ci_zoom(x1,y1,x2,y2)` Set the display area to be from the bottom left corner specified by `(x1,y1)` to the top right corner specified by `(x2,y2)`.

10.8   View Commands

- `ci_showgrid()` Show the grid points.

- `ci_hidegrid()` Hide the grid points points.

- `ci_gridsnap("flag")` Setting flag to "on" turns on snap to grid, setting flag to "off"turns off snap to grid.

- `ci_setgrid(density,"type")` Change the grid spacing. The density parameter specifies the space between grid points, and the type parameter is set to `"cart"` for Cartesian coordinates or `"polar"` for polar coordinates.

- `ci_refreshview()` Redraws the current view.

- `ci_minimize()` minimizes the active magnetics input view.

- `ci_maximize()` maximizes the active magnetics input view.

- `ci_restore()` restores the active magnetics input view from a minimized or maximized state.

- `ci_resize(width,height)` resizes the active magnetics input window client area to width $\times$ height.

10.9 Object Properties

- `ci_getmaterial('materialname')` fetches the material specified by `materialname` from the materials library.

- `ci_addmaterial("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 in units of S/m.

  oy Electrical conductivity in the y- or z-direction in units of S/m.

  ex Relative permittivity in the x- or r-direction.

  ey Relative 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.

- `ci_addpointprop("pointpropname",Vp,jp)` adds a new point property of name `"pointpropname"` with either a specified potential Vp a point current density jp in units of A/m.

- `ci_addboundprop("boundpropname", Vs, js, 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 C1 and C0 as required and BdryFormat to 1. Set all other parameters to zero.

  To obtain a prescribes surface current density, set js 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.

- `ci_addconductorprop("conductorname", Vc, jc, 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 current and 1 for prescribed voltage.

- `ci_deletematerial("materialname")` deletes the material named `"materialname"`.

- `ci_deleteboundprop("boundpropname")` deletes the boundary property named `"boundpropname"`.

- `ci_deleteconductor("conductorname")` deletes the conductor named `conductorname`.

- `ci_deletepointprop("pointpropname")` deletes the point property named `"pointpropname"`

- `ci_modifymaterial("BlockName",propnum,value)` This function allows for modification of a material's properties without redefining the entire material (e.g. so that current can be modified from run to run). The material to be modified is specified by `"BlockName"`. The next parameter is the number of the property to be set. The last number is the value to be applied to the specified property. The various properties that can be modified are listed below:

propnum	Symbol	Description
0	BlockName	Name of the material
1	ox	x (or r) direction conductivity
2	oy	y (or z) direction conductivity
3	ex	x (or r) direction relative permittivity
4	ey	y (or z) direction relative permittivity
5	ltx	x (or r) direction dielectric loss tangent
6	lty	y (or z) direction dielectric loss tangent

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

propnum	Symbol	Description		
0	BdryName	Name of boundary property		
1	Vs	Fixed Voltage		
2	js	Prescribed current density		
3	c0	Mixed BC parameter		
4	c1	Mixed BC parameter		
5	BdryFormat	Type of boundary condition:		
		0	=	Prescribed V
		1	=	Mixed
		2	=	Surface current density
		3	=	Periodic
		4	=	Antiperiodic

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

propnum	Symbol	Description
0	PointName	Name of the point property
1	Vp	Prescribed nodal voltage
2	jp	Point current density in A/m

- `ci_modifyconductorprop("ConductorName",propnum,value)` This function 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:

propnum	Symbol	Description
0	ConductorName	Name of the conductor property
1	Vc	Conductor voltage
2	jc	Total conductor current
3	ConductorType	0 = Prescribed current, 1 = Prescribed voltage

10.10   Miscellaneous

- `ci_savebitmap("filename")` saves a bitmapped screenshot of the current view to the file specified by `"filename"`, subject to the `printf`-type formatting explained previously for the `ci_saveas` command.

- `ci_savemetafile("filename")` saves a metafile screenshot of the current view to the file specified by `"filename"`, subject to the `printf`-type formatting explained previously for the `ci_saveas` command.

- `ci_refreshview()` Redraws the current view.

- `ci_close()` closes the preprocessor window and destroys the current document.

- `ci_shownames(flag)` This function allow the user to display or hide the block label names on screen. To hide the block label names, `flag` should be 0. To display the names, the parameter should be set to 1.

- `ci_readdxf("filename")` This function imports a dxf file specified by `"filename"`.

- `ci_savedxf("filename")` This function saves geometry information in a dxf file specified by `"filename"`.

- `ci_defineouterspace(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.

- `ci_attachouterspace()` marks all selected block labels as members of the external region used for modeling unbounded axisymmetric problems via the Kelvin Transformation.

- `ci_detachouterspace()` undefines all selected block labels as members of the external region used for modeling unbounded axisymmetric problems via the Kelvin Transformation.

- `ci_attachdefault()` marks the selected block label as the default block label. This block label is applied to any region that has not been explicitly labeled.

- `ci_detachdefault()` undefines the default attribute for the selected block labels.

- `ci_makeABC(n,R,x,y,bc)` creates a series of circular shells that emulate the impedance of an unbounded domain (i.e. an Inprovised Asymptotic Boundary Condition). The `n` parameter contains the number of shells to be used (should be between 1 and 10), `R` is the radius of the solution domain, and `(x,y)` denotes the center of the solution domain. The `bc` parameter should be specified as 0 for a Dirichlet outer edge or 1 for a Neumann outer edge. If the function is called without all the parameters, the function makes up reasonablevaluesforthemissingparameters.

11 Current Flow Post Processor Command Set

There are a number of Lua scripting commands designed to operate in the postprocessor. As with the preprocessor commands, these commands can be used with either the underscore naming or with the no-underscore naming convention.

11.1 Data Extraction Commands

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

type	Integral
0	$E \cdot t$
1	$J \cdot n$
2	Contour length
3	Average voltage over contour
4	Force from stress tensor
5	Torque from stress tensor

  This integral returns either 1 or 2 values, depending on the integral type, *e.g.*:

  `Fx, Fy = co_lineintegral(4)`

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

type	Integral
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 a value that can be complex, as necessary.

- `co_getpointvalues(X,Y)` Get the values associated with the point at x,y The return values, in order, are:

Symbol	Definition
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

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

PlotType	Definition
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)

Valid file formats are:

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

For example, if one wanted to plot *V* to the screen with 200 points evaluated to make the graph, the command would be:

`co_makeplot(0,200)`

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

`co_makeplot(0,200,"c:temp.emf")`

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

`co_makeplot(0,200,"c:temp.txt",0)`

- `co_getprobleminfo()` Returns info on problem description. Returns four values:

Return value	Definition
1	problem type
2	frequency in Hz
3	depth assumed for planar problems in meters
4	length unit used to draw the problem, represented in meters.

- `co_getconductorproperties("conductor")` Properties are returned for the conductor property named "conductor". Two values are returned: The voltage of the specified conductor, and the current on the specified conductor.

11.2   Selection Commands

- `co_seteditmode(mode)` Sets the mode of the postprocessor to point, contour, or area mode. Valid entries for mode are `"point"`, `"contour"`, and `"area"`.

- `co_selectblock(x,y)` Select the block that contains point `(x,y)`.

- `co_groupselectblock(n)` Selects all of the blocks that are labeled by block labels that are members of group n. If no number is specified (*i.e.* `co_groupselectblock()`), all blocks are selected.

- `co_selectconductor("name")` Selects all nodes, segments, and arc segments that are part of the conductor specified by the string (`"name"`). This command is used to select conductors for the purposes of the "weighted stress tensor" force and torque integrals, where the conductors are points or surfaces, rather than regions (*i.e.* can't be selected with `co_selectblock`).

- `co_addcontour(x,y)` Adds a contour point at `(x,y)`. If this is the first point then it starts a contour, if there are existing points the contour runs from the previous point to this point. The `co_addcontour` command has the same functionality as a right-button-click contour point addition when the program is running in interactive mode.

- `co_bendcontour(angle,anglestep)` Replaces the straight line formed by the last two points in the contour by an arc that spans angle degrees. The arc is actually composed of many straight lines, each of which is constrained to span no more than anglestep degrees. The `angle` parameter can take on values from -180 to 180 degrees. The `anglestep` parameter must be greater than zero. If there are less than two points defined in the contour, this command is ignored.

- `co_selectpoint(x,y)` Adds a contour point at the closest input point to `(x,y)`. If the selected point and a previous selected points lie at the ends of an arcsegment, a contour is added that traces along the arcsegment. The `selectpoint` command has the same functionality as the left-button-click contour point selection when the program is running in interactive mode.

- `co_clearcontour()` Clear a prevously defined contour

- `co_clearblock()` Clear block selection

11.3 Zoom Commands

- `co_zoomnatural()` Zoom to the natural boundaries of the geometry.

- `co_zoomin()` Zoom in one level.

- `co_zoomout()` Zoom out one level.

- `co_zoom(x1,y1,x2,y2)` Zoom to the window defined by lower left corner `(x1,y1)` and upper right corner `(x2,y2)`.

11.4 View Commands

- `co_showmesh()` Show the mesh.

- `co_hidemesh()` Hide the mesh.

- `co_showpoints()` Show the node points from the input geometry.

- `co_hidepoints()` Hide the node points from the input geometry.

- `co_smooth("flag")` This function controls whether or not smoothing is applied to the D and E fields which are naturally piece-wise constant over each element. Setting flag equal to `"on"` turns on smoothing, and setting flag to `"off"` turns off smoothing.

- `co_showgrid()` Show the grid points.

- `co_hidegrid()` Hide the grid points points.

  `co_gridsnap("flag")` Setting flag to "on" turns on snap to grid, setting flag to "off" turns off snap to grid.

- `co_setgrid(density,"type")` Change the grid spacing. The density parameter specifies the space between grid points, and the type parameter is set to `"cart"` for Cartesian coordinates or `"polar"` for polar coordinates.

- `co_hidedensityplot()` hides the current density plot.

- `co_showdensityplot(legend,gscale,type,upper,lower)` Shows the current 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. Specific choices for the type of density plot include:

type	Description
0	$|V|$
1	$|\mathrm{Re}(V)|$
2	$|\mathrm{Im}(V)|$
3	$|J|$
4	$|\mathrm{Re}(J)|$
5	$|\mathrm{Im}(J)|$
6	$|E|$
7	$|\mathrm{Re}(E)|$
8	$|\mathrm{Im}(E)|$

- `co_hidecontourplot()` Hides the contour plot.

- `co_showcontourplot(numcontours,lower_V,upper_V),type` shows the V contour plot with options:

  `numcontours` Number of equipotential lines to be plotted;

  `upper_V` Upper limit for contours;

`lower_V` Lower limit for contours;

`type` the type of contour plot to be rendered.

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

The type can take on the values of `"real"`, `"imag"`, or `"both"`, denoting the real part of voltage, the imaginary part of voltage, or both components of voltage.

- `co_showvectorplot(type,scalefactor)` controls the display of vectors denoting the field strength and direction. The `type` parameter can take on the following values:

type	Description
0	No vector plot
1	Re(J)
2	Re(E)
3	Im(J)
4	Im(E)
5	Re(J) and $Im(J)$
6	Re(E) and $Im(E)$

The `scalefactor` determines the relative length of the vectors.

If the scale is set to 1, the length of the vectors are chosen so that the highest field magnitude corresponds to a vector that is the same length as the current grid size setting.

- `co_minimize()` minimizes the active magnetics input view.

- `co_maximize()` maximizes the active magnetics input view.

- `co_restore()` restores the active magnetics input view from a minimized or maximized state.

- `co_resize(width,height)` resizes the active magnetics input window client area to width $\times$ height.

11.5   Miscellaneous

- `co_close()` close the current postprocessor window.

- `co_refreshview()` Redraws the current view.

- `co_reload()` Reloads the solution from disk.

- `co_savebitmap("filename")` saves a bitmapped screen shot of the current view to the file specified by `"filename"`. Note that if you use a path you must use two backslashes (e.g. `"c:\\temp\\myfile.bmp"`). If the file name contains a space (e.g. file names like c:\program files\stuff) you must enclose the file name in (extra) quotes by using a \" sequence. For example:

  `co_savebitmap("\"c:\\temp\\screenshot.bmp\"")`

- `co_savemetafile("filename")` saves a metafile screenshot of the current view to the file specified by `"filename"`, subject to the printf-type formatting explained previously for the `savebitmap` command.

- `co_shownames(flag)` This function allow the user to display or hide the block label names on screen. To hide the block label names, `flag` should be 0. To display the names, the parameter should be set to 1.

- `co_numnodes` Returns the number of nodes in the in focus current flow output mesh.

- `co_numelements` Returns the number of elements in the in focus current flow output mesh.

- `co_getnode(n)` Returns the (x,y) or (r,z) position of the nth mesh node.

- `co_getelement(n)` MOGetElement[n] returns the following proprerties for the nth element:

  1. Index of first element node
  2. Index of second element node
  3. Index of third element node
  4. x (or r) coordinate of the element centroid
  5. y (or z) coordinate of the element centroid
  6. element area using the length unit defined for the problem
  7. group number associated with the element