Introduction to the use of FEMM

Attilio Milanese

CAS course on Normal- and Superconducting Magnets
19 Nov. – 2 Dec. 2023
St. Pölten, Austria

Thanks in particular to Herbert De Gersem, Thomas Zickler and Susana Izquierdo Bermudez

Installation

HomePage:Finite Element Meth

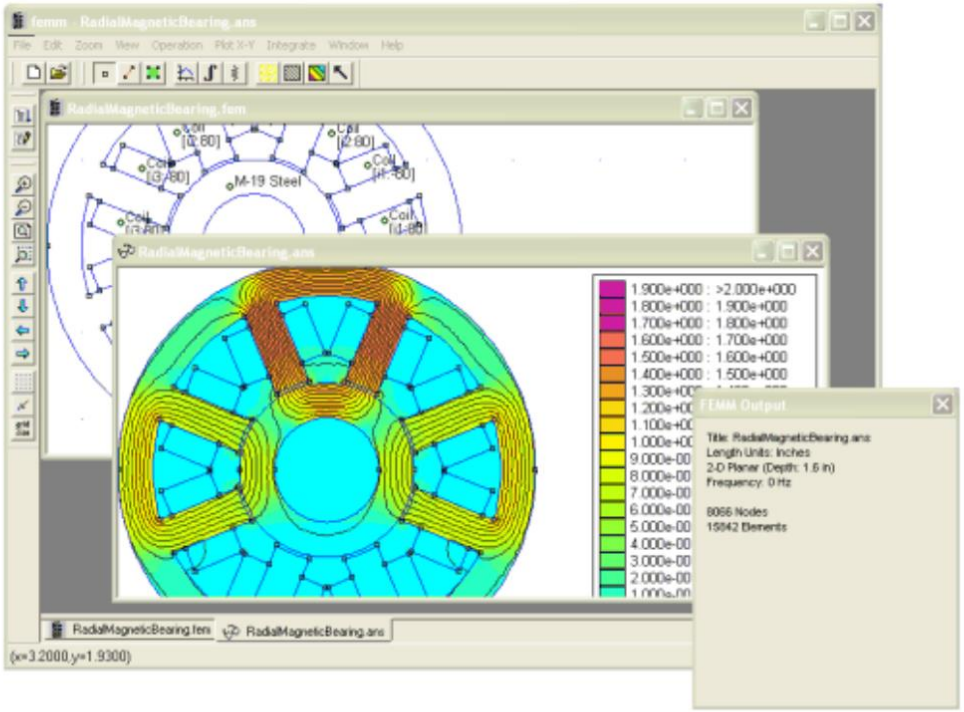
https://www.femm.info/wiki/HomePage 130% Search

Finite Element Method Magnetics : HomePage

Finite Element Method Magnetics

Magnetics, Electrostatics, Heat Flow, and Current Flow

- Download
- Documentation
- FAQ
- Linux Support
- Examples
- User Contributions
- Miscellaneous
- Related Links
- Author



1 Introduction

1.1 Overall Purpose

These lecture notes deal with electromagnetic field solvers. The main purpose is to explain what is behind a software for electromagnetic-field solving such that calculations for particle-accelerator components can be carried out with confidence.

1.2 Used Software

An introductory class as this one may benefit from a few hands-on sessions using generally available software tools. For exercising, I suggest

1. FEMM
2. CST Student Edition

1.2.1 Using FEMM on Windows

Install FEMM itself

Download FEMM from http://www.femm.info/wiki/HomePage and follow the installation instructions.

Scripting FEMM from MATLAB® and GNU OCTAVE

Search for the directory Add the m-files to your MATLAB® or GNU OCTAVE installation by typing

```
>addpath("~/.wine/drive_c/femm42/mfiles");
>savepath;
```

on the GNU OCTAVE or MATLAB® prompt. Now, you should be able test your installation by typing

```
>openfemm
```

1.2.2 Using FEMM on Linux or Mac

Install FEMM itself

FEMM is available as a WINDOWS binary, thus the installation on a recent WINDOWS version is straight forward. To install FEMM on a MAC or LINUX, we suggest to install WINE first. WINE is a free software that is available via several package managers on both MAC and LINUX. For example, the installation via the command line looks like

```
apt-get install wine          % for Ubuntu and Debian Linux
brew install wine          % for a Mac using homebrew
                        % (see www.brew.sh)
```

Alternatively you can buy a commercial WINE license called CROSSOVER from CODEWEAVERS (www.codeweavers.com) which is particularly easy to use. After having installed WINE, you can run the WINDOWS installer on your MAC or LINUX machine from the command line by

```
# wine femm42bin_win32.exe
```


assuming that the WINE executable is in your path. After the installation with standard options the FEMM installation is located on your hard disk in the directory ~/.wine/drive_c/femm42/. You can execute FEMM from the command line by

```
# wine ~/.wine/drive_c/femm42/bin/femm.exe
```

Scripting FEMM from MATLAB® and GNU OCTAVE (automatic)

The scripting environment requires some additional steps. The easiest approach is to use the modified files from us (see our website) openfemm.m and callfemm.m and replace the ones in the folder ~/.wine/drive_c/femm42/mfiles/. These m-files will look automatically in a few standard locations used by FEMM and WINE. Add the m-files to your MATLAB® or GNU OCTAVE installation by typing

```
>addpath("~/.wine/drive_c/femm42/mfiles");
>savepath;
```

on the GNU OCTAVE or MATLAB® prompt. Now, you should be able test your installation by typing

```
>openfemm
```

Scripting FEMM from Octave (manual installation)

There is a detailed description on the FEMM website on how to do the steps above manually (http://www.femm.info/wiki/LinuxSupport). In several m-files the hard coded information must be changed, e.g., the installation path of FEMM. However, depending on the GNU OCTAVE version that you use, there might be a problem with the line

```
system(['wine "',rootdir,'femm.exe" -filelink'],0,'async');
```

in the file located at ~/.wine/drive_c/femm42/ mfiles/openfemm.m. This line should be replaced by

```
system(['/wine ',rootdir,'/femm.exe -filelink &']);
```

For the room temperature and superconducting magnets hands-on sessions we will use FEMM and its embedded scripting language Lua.

For Windows users: just install FEMM
For Linux or Mac users: see here

General comments

Electromagnetic field solvers

TECHNISCHE UNIVERSITÄT DARMSTADT

- commercial
  or academic
  or freeware
- low-frequency
  and/or high-frequency
- circuit
  and/or 2D fields
  and/or 3D fields
- problem specific
  or multi-purpose
- electromagnetic fields
  and/or multi-physics
- cheap or expensive

non-exhaustive list !!

CEDRAT

CST

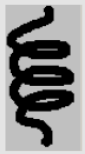
JMAG

infolytica corporation

opera simulation software

ANSYS

FEMM



FEKO Comprehensive Electromagnetic Solutions

COMSOL MULTIPHYSICS

https://roxie.docs.cern.ch

Roxie Webpage

CERN

Poisson (Los Alamos)
Radia (ESRF)
Mermaid (Novosibirsk)

1. Finite Element Method Magnetics

   www.femm.info

2. H. De Gersem, lectures at the CAS on Numerical Methods for Analysis, Design and Modeling of Particle Accelerators, https://indico.cern.ch/event/759124/

   Field Solver lectures

   Magnetostatic Simulation of an Accelerator Magnet exercise

   (make your own finite-element solver)

   https://arxiv.org/pdf/2006.10463.pdf

   https://arxiv.org/pdf/2006.10353.pdf

3. T. Zickler, Numerical design of a normal-conducting, iron-dominated electro-magnet using FEMM 4.2, JUAS

   Tutorial attached on indico

FEMM can be used in different ways

Through the GUI (Graphical User Interface)

Through scripting, either with the embedded Lua, or with MATLAB®, GNU Octave, Python, etc.

Through a mix of GUI and scripting

For details, see the excellent FEMM manual

In all cases, please check that the results make sense (flux lines, analytical estimates), exploit symmetries, run the model with different mesh sizes, check also the impact of the background dimensions

https://www.lua.org

Lua (programming language)

Article Talk

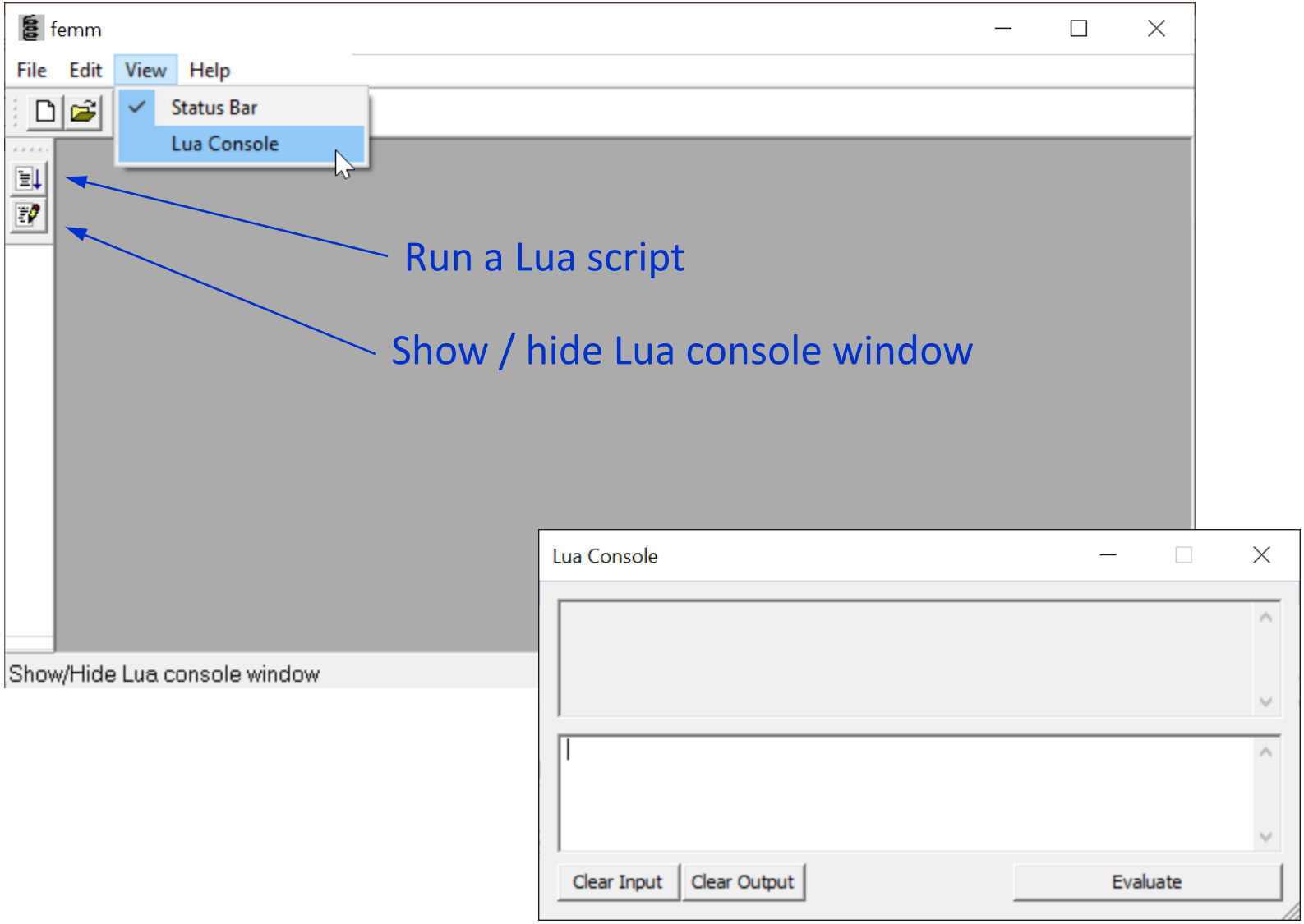
From Wikipedia, the free encyclopedia

Lua (/ˈluːə/ *LOO-ə*; from Portuguese: *lua* [ˈlu(w)ɐ] meaning *moon*)

in 2023

To script in FEMM with Lua, see the dedicated chapter
“Lua scripting” in the excellent FEMM manual

No separate or additional installation for Lua shall be needed


femm
File
Edit
View
Help
Status Bar
Lua Console
Run a Lua script
Show / hide Lua console window
Show/Hide Lua console window
Lua Console
Clear Input
Clear Output
Evaluate

2.2.1 Preprocessor Drawing Modes

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

One can switch between drawing modes by clicking the appropriate button on the Drawing Mode potion of the toolbar. This section of the toolbar is pictured in Figure 2.1. The buttons

Figure 2.1: Drawing Mode toolbar buttons.

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

[from the FEMM manual]

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

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

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

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

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

Table 2.1: Magnetics Preprocessor hot keys

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

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

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

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

Table 2.2: Magnetics Preprocessor Mouse button actions

[from the FEMM manual]

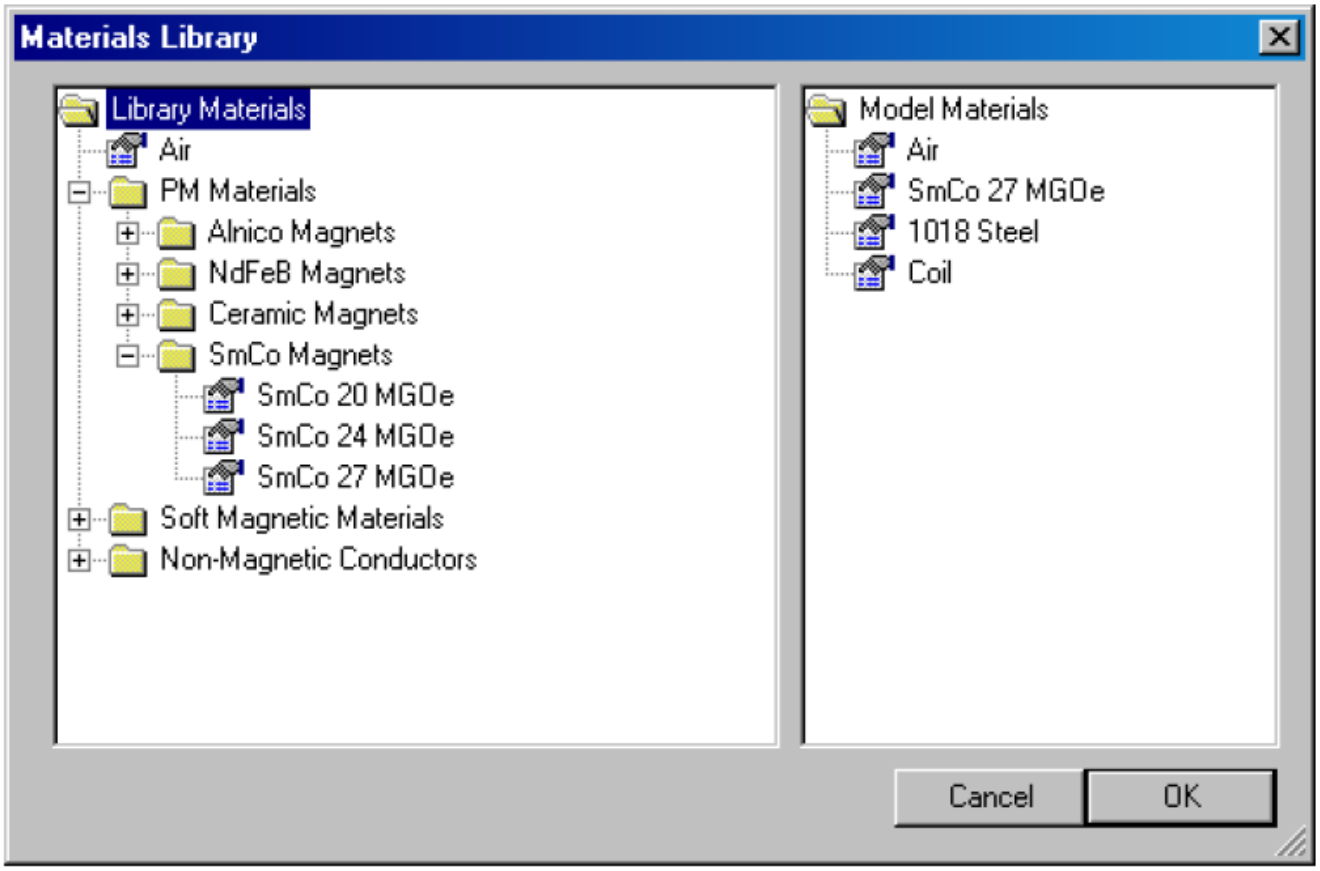
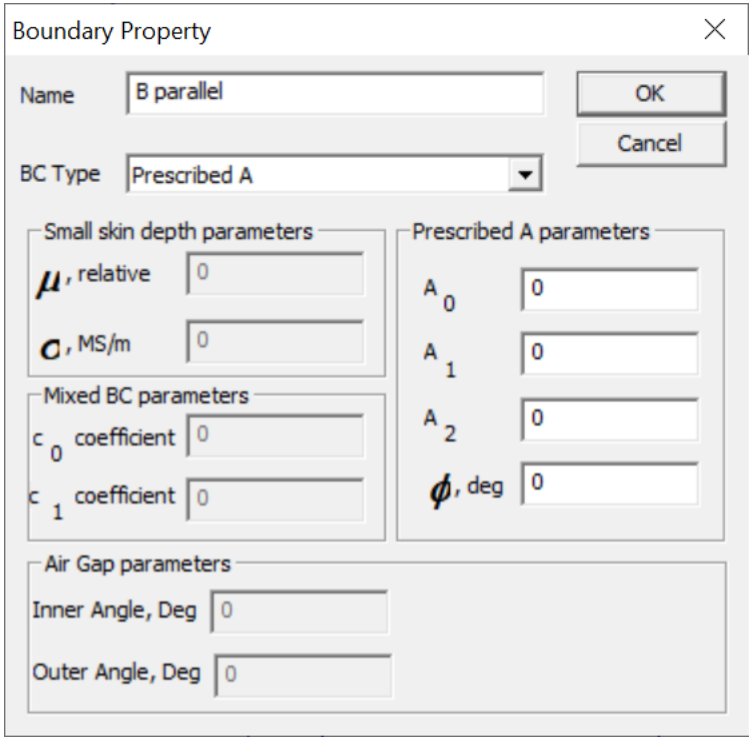


Figure 2.13: Materials Library dialog.

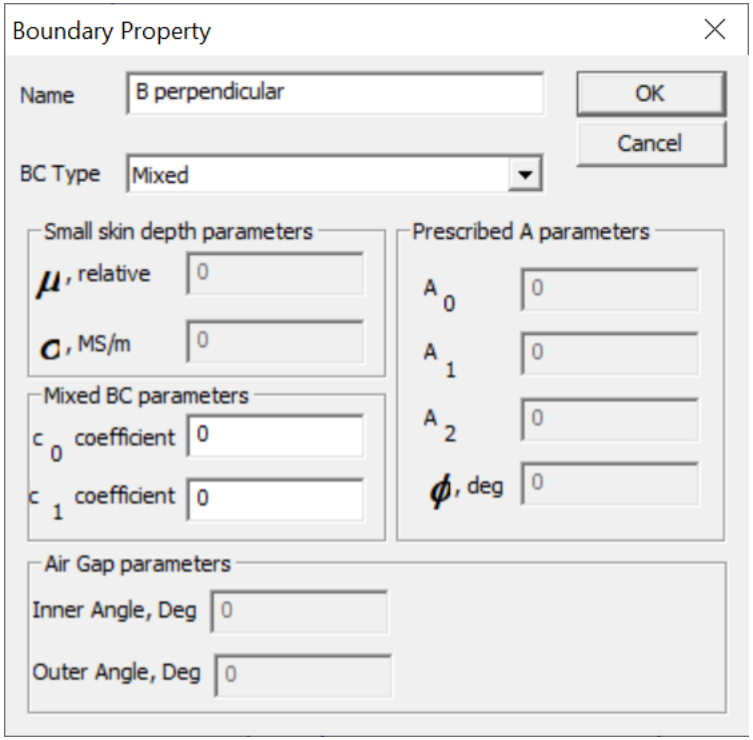
[from the FEMM manual]

Typical boundary conditions (on lines)

B parallel



B perpendicular



(this is sort of implicit when using linear triangles, see FEMM documentation)

Fancy boundary condition (extra for sc magnet hands-on)

2.2.8 Exterior Region

One often desires to solve problems on an unbounded domain. Appendix A.3.3 describes an easy-to-implement conformal mapping method for representing an unbounded domain in a 2D planar finite element analysis. Essentially, one models two disks–one represents the solution region of interest and contains all of the items of interest, around which one desires to determine the magnetic field. The second disk represents the region exterior to the first disk. If periodic boundary conditions are employed to link the edges of the two disks, it can be shown (see Appendix A.3.3) that the result is exactly equivalent to solving for the fields in an unbounded domain.

A.3.3 Kelvin Transformation

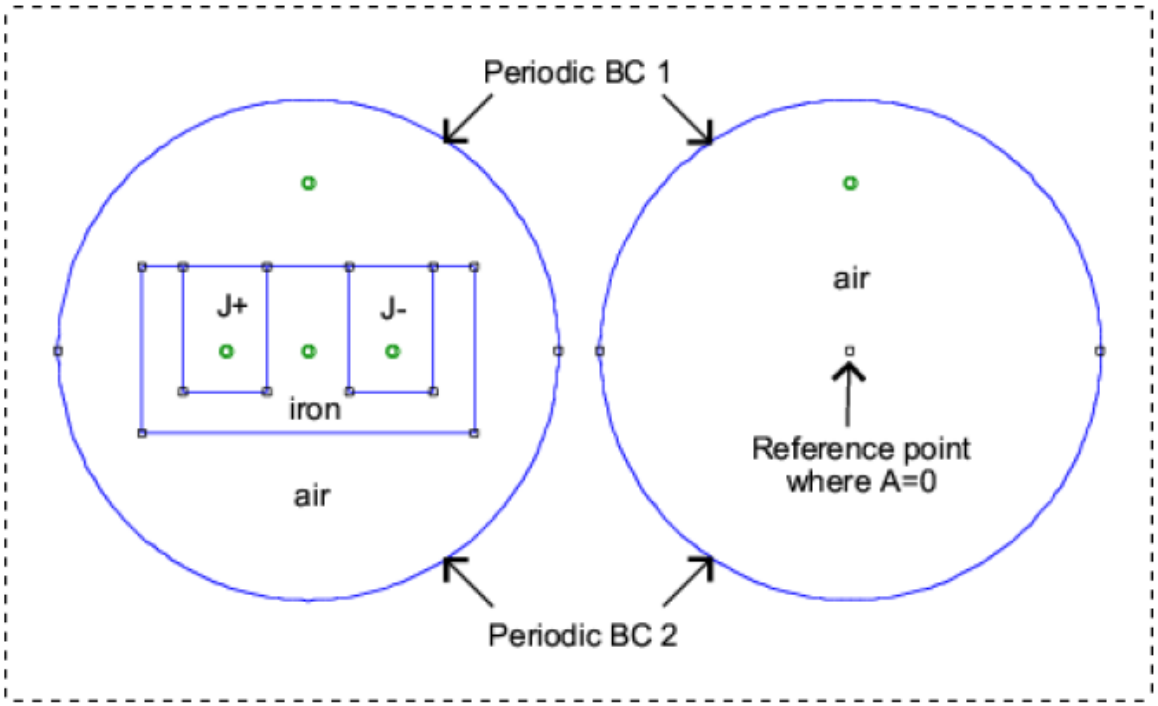


Figure A.8: Example input geometry.

[from the FEMM manual]

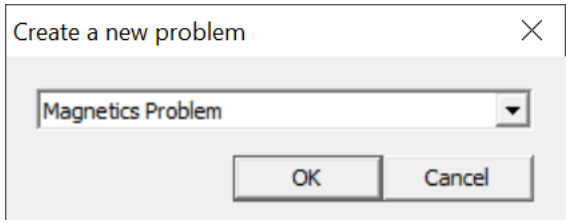
Walk-through a FEMM example

A decalogue (in 8 points)

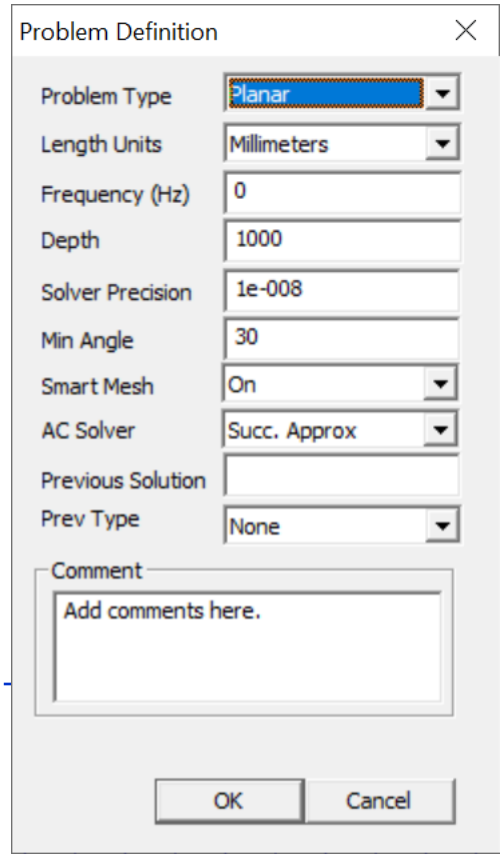
1. Create a new file and set main problem parameters
2. Declare a few variables (for parametric analyses)
3. Load or prepare material properties, boundary conditions and circuit elements
4. Define the geometry
5. Save and mesh
6. Solve
7. Post-process
8. Check flux lines, compare against analytical estimates, solve with a different mesh size, check with different background dimensions (in particular for “open” magnets), etc.

1. Create a new file and set main problem parameters

---> File ---> New



---> Problem

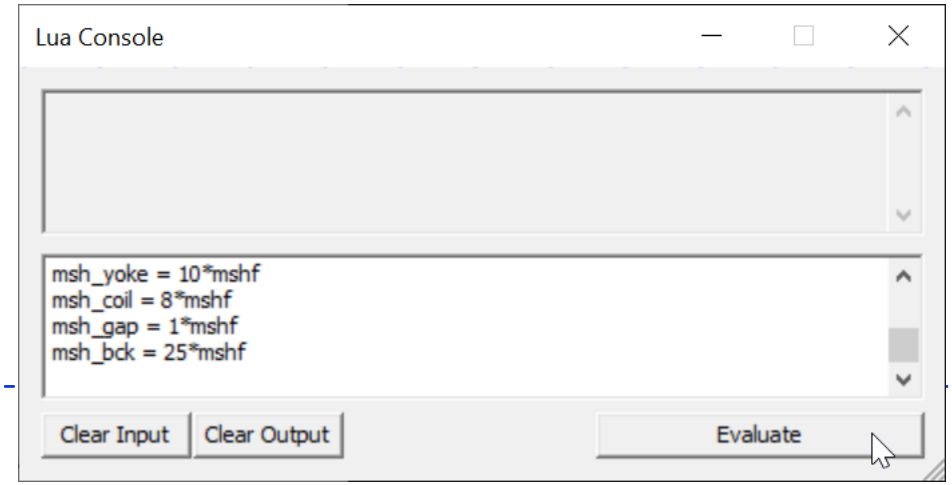


1 m depth, so results (energy, inductance, force, …) will be per m length

```
-- Creates a new preprocessor document (magnetics problem)
newdocument(0)

-- Main problem parameters
--    0 frequency
--    mm units
--    planar problem
--    solver precision
--    depth, set to 1 m so to have results per m length
mi_probdef(0, "millimeters", "planar", 1e-8, 1000)
```

2. Declare a few variables (for parametric analyses)

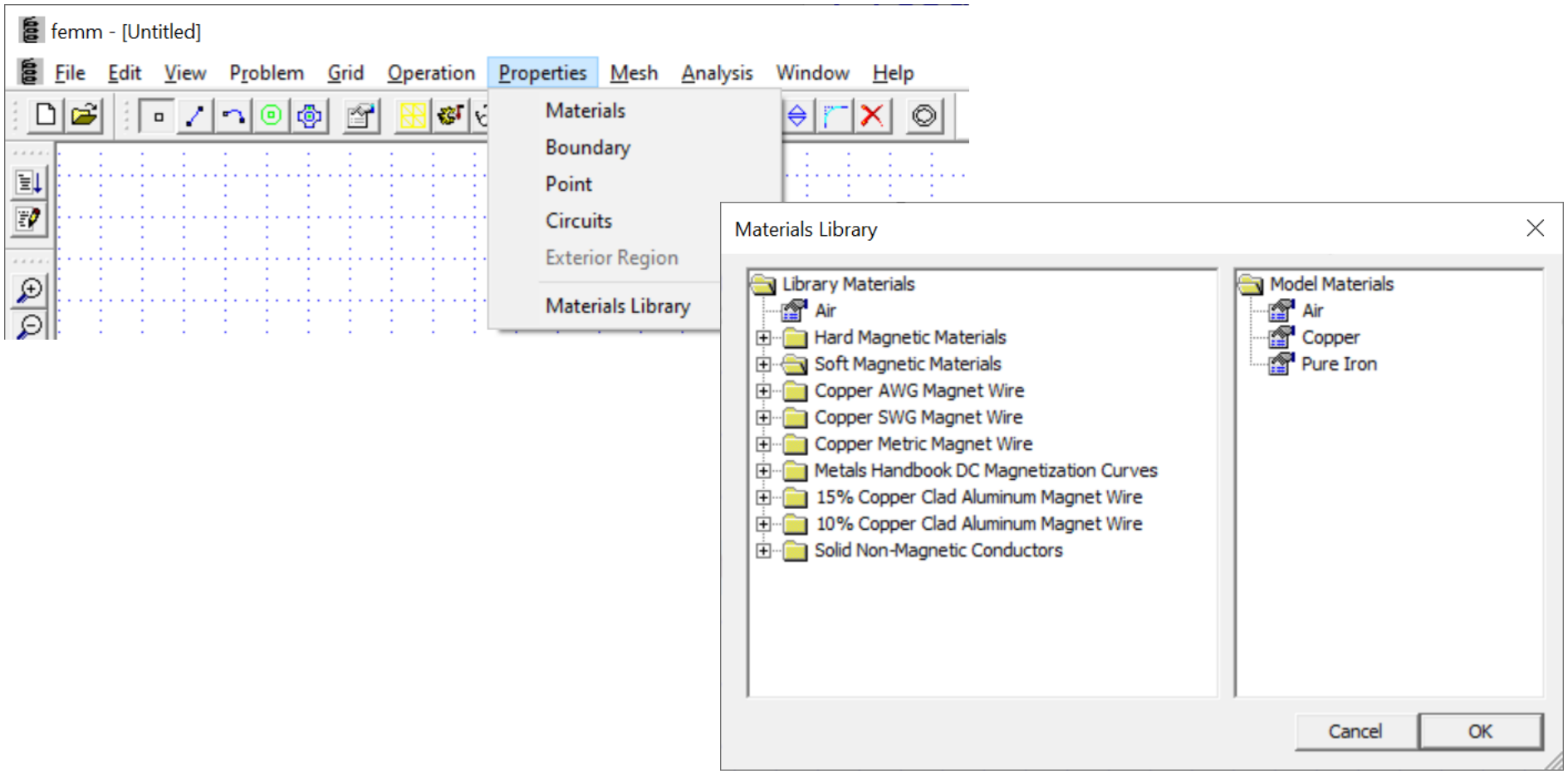
Write (or copy & paste) in Lua console, then click Evaluate




```
-- A few variables
w_coil = 99
h_coil = 22
x_bck = 800
y_bck = 400
workfolder = "C:\\temp\\"
filename = "dipole"

-- Current and number of turns per coil block
current = 450
turns = 18

-- Mesh parameters
mshf = 1
msh_yoke = 10*mshf
msh_coil = 8*mshf
msh_gap = 1*mshf
msh_bck = 25*mshf
```

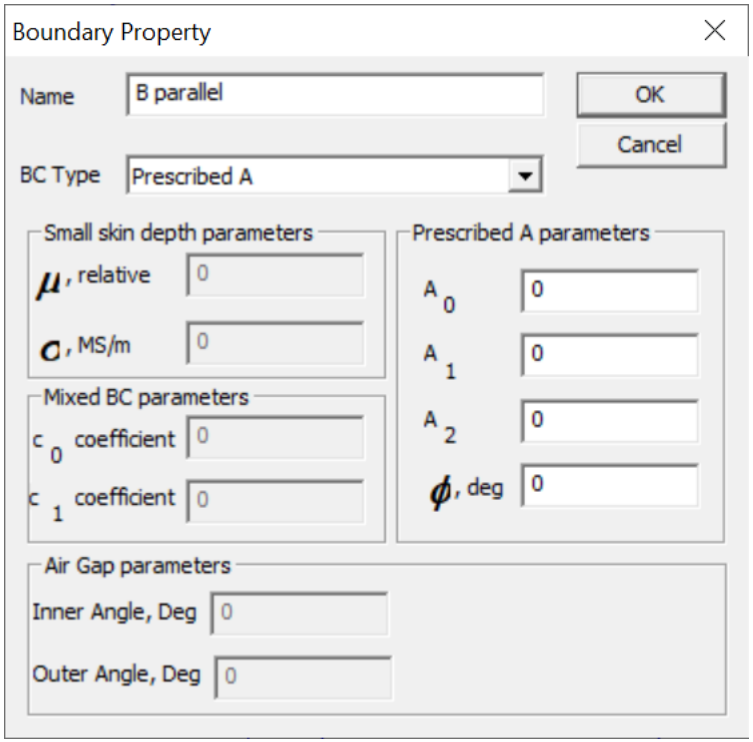
mesh size define via a scaling factor

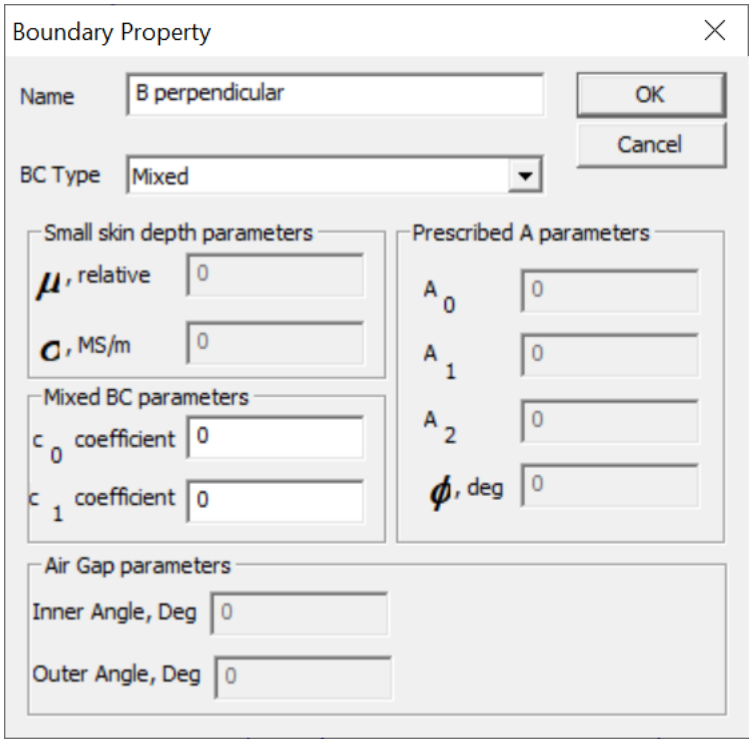
3. Load or prepare material properties (from the available library), boundary conditions and circuit elements




```
-- Material properties, from the available library
mi_getmaterial("Air")
mi_getmaterial("Pure Iron")
mi_getmaterial("Copper")
```

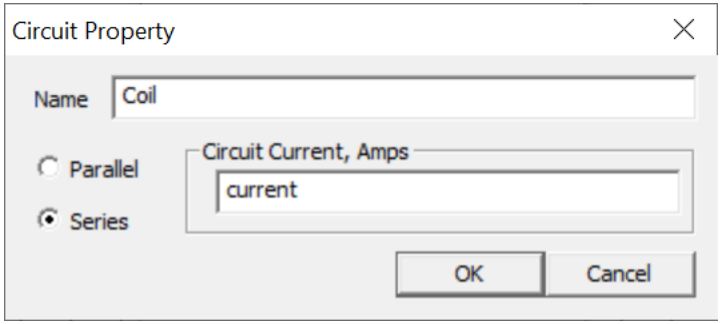
3. Load or prepare material properties (from the available library), boundary conditions and circuit elements






```
-- Boundary conditions
mi_addboundprop("B parallel", 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
mi_addboundprop("B perpendicular", 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0)
```

3. Load or prepare material properties (from the available library), boundary conditions and circuit elements

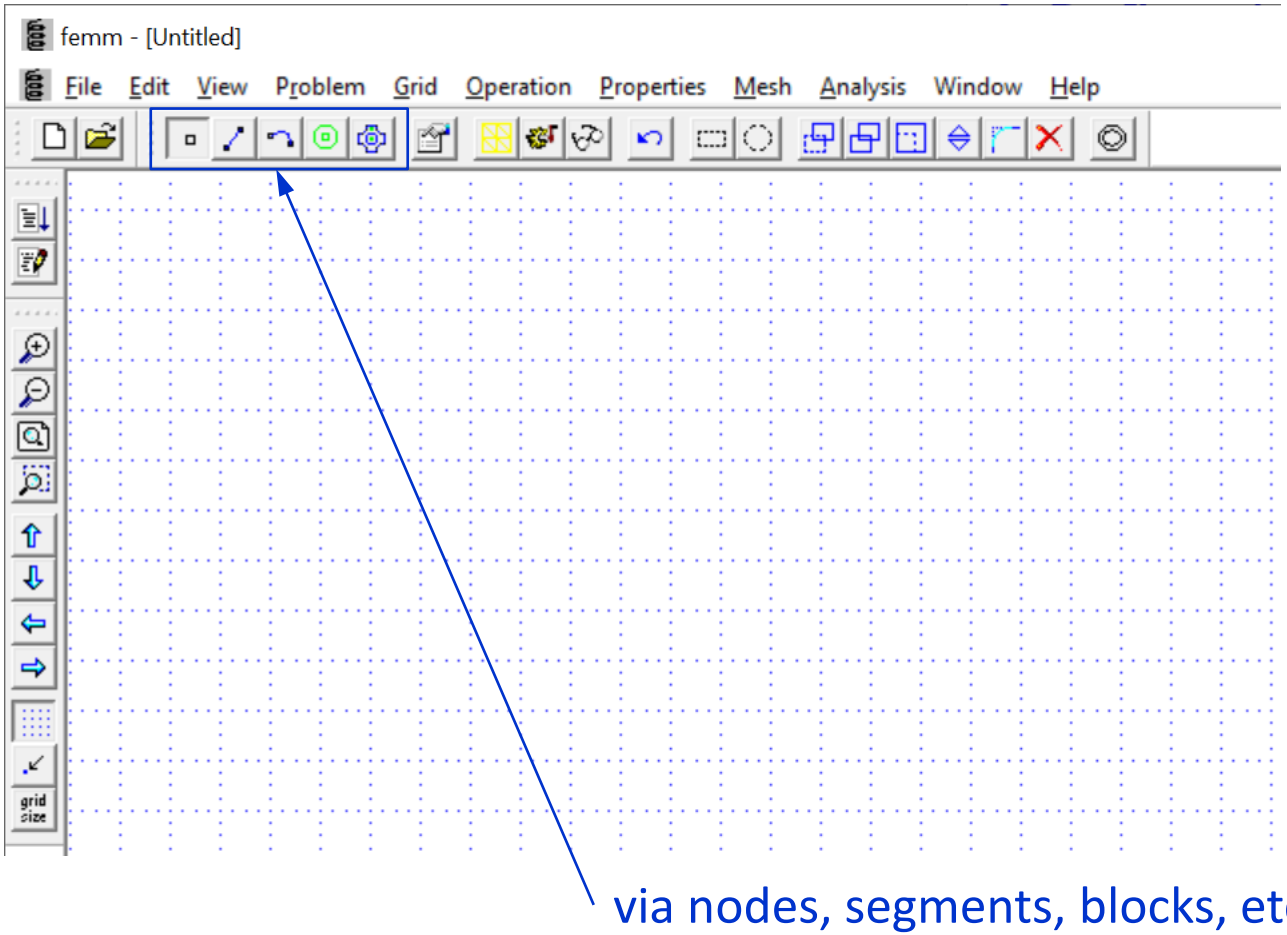


"current" is a previously defined variable, alternatively you can enter a number

---

```
-- A circuit, multiple ones are possible
mi_addcircprop("Coil", current, 1)
```

4. Define the geometry (iron, coil, air, background)



via nodes, segments, blocks, etc…

Hot keys are particularly useful, also the grid can be handy

Previously defined variables can be used to describe the geometry

Copy and paste in the Lua console is also a possibility

Another approach is to import a DXF

4. Define the geometry (iron, coil, air, background)

```
-- Yoke (array of points, for convenience)
x_yoke, y_yoke = {}, {}
x_yoke[1], y_yoke[1] = 0, 25
x_yoke[2], y_yoke[2] = 71, 25
x_yoke[3], y_yoke[3] = 71, 24.2
x_yoke[4], y_yoke[4] = 90, 24.2
x_yoke[5], y_yoke[5] = 105, 60
x_yoke[6], y_yoke[6] = 105, 295
x_yoke[7], y_yoke[7] = 55, 345
x_yoke[8], y_yoke[8] = -409, 345
x_yoke[9], y_yoke[9] = -459, 295
x_yoke[10], y_yoke[10] = -459, 0
x_yoke[11], y_yoke[11] = -249, 0
x_yoke[12], y_yoke[12] = -249, 127
x_yoke[13], y_yoke[13] = -105, 127
x_yoke[14], y_yoke[14] = -105, 60
x_yoke[15], y_yoke[15] = -90, 24.2
x_yoke[16], y_yoke[16] = -71, 24.2
x_yoke[17], y_yoke[17] = -71, 25
np_yoke = getn(x_yoke)
for ip_yoke = 1, np_yoke do
   mi_addnode(x_yoke[ip_yoke], y_yoke[ip_yoke])
end
for ip_yoke = 1, np_yoke-1 do
   mi_addsegment(x_yoke[ip_yoke], y_yoke[ip_yoke], x_yoke[ip_yoke+1], y_yoke[ip_yoke+1])
end
mi_addsegment(x_yoke[np_yoke], y_yoke[np_yoke], x_yoke[1], y_yoke[1])
--
mi_addblocklabel(0, 150)
mi_selectlabel(0, 150)
mi_setblockprop("Pure Iron", 0, msh_yoke)
mi_clearselected()
```

definition of coordinates of points via an array, for convenience

cycles to create node and then segments

block label, assign material properties and mesh size

4. Define the geometry (iron, coil, air, background)

```
-- Coil
mi_addnode(127, 100)
mi_addnode(127+w_coil, 100)
mi_addnode(127+w_coil, 100+h_coil)
mi_addnode(127, 100+h_coil)
mi_addsegment(127, 100, 127+w_coil, 100)
mi_addsegment(127+w_coil, 100, 127+w_coil, 100+h_coil)
mi_addsegment(127+w_coil, 100+h_coil, 127, 100+h_coil)
mi_addsegment(127, 100+h_coil, 127, 100)
--
mi_addblocklabel(127+w_coil/2, 100+h_coil/2)
mi_selectlabel(127+w_coil/2, 100+h_coil/2)
mi_setblockprop("Copper", 0, msh_coil, "Coil", 0, 0, turns)
-- copies
mi_selectrectangle(127, 100, 127+w_coil, 100+h_coil, 4)
mi_copytranslate(0, -(h_coil+5), 2, 4)
mi_selectrectangle(127, 100-2*(h_coil+5), 127+w_coil, 100+h_coil, 4)
mi_copytranslate(-336, 0, 1, 4)
-- change sign of current on one side
mi_selectrectangle(127, 100-2*(h_coil+5), 127+w_coil, 100+h_coil, 2)
mi_setblockprop("Copper", 0, msh_coil, "Coil", 0, 0, -turns)
mi_clearselected()
```

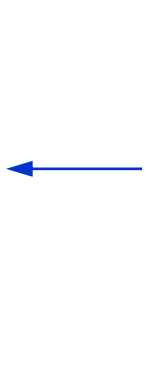
for the current carrying region, assign the relevant circuit element and number of turns

4. Define the geometry (iron, coil, air, background)

```
-- Air region (background and gap)
mi_addnode(0, 0)
mi_addnode(130, 0)
mi_addnode(-130, 0)
mi_addsegment(130, 0, x_yoke[4], y_yoke[4])
mi_addsegment(-130, 0, x_yoke[15], y_yoke[15])
--
mi_addnode(-500, 0)
mi_addnode(x_bck, 0)
mi_addnode(x_bck, y_bck)
mi_addnode(-500, y_bck)
mi_addsegment(-500, 0, x_bck, 0)
mi_addsegment(x_bck, 0, x_bck, y_bck)
mi_addsegment(x_bck, y_bck, -500, y_bck)
mi_addsegment(-500, y_bck, -500, 0)
--
mi_addblocklabel(0, 10)
mi_selectlabel(0, 10)
mi_setblockprop("Air", 0, msh_gap)
mi_clearselected()
--
mi_addblocklabel(150, 150)
mi_selectlabel(150, 150)
mi_setblockprop("Air", 0, msh_bck)
mi_clearselected()
--
mi_addblocklabel(-150, 20)
mi_selectlabel(-150, 20)
mi_setblockprop("Air", 0, 6*msh_gap)
mi_clearselected()
```

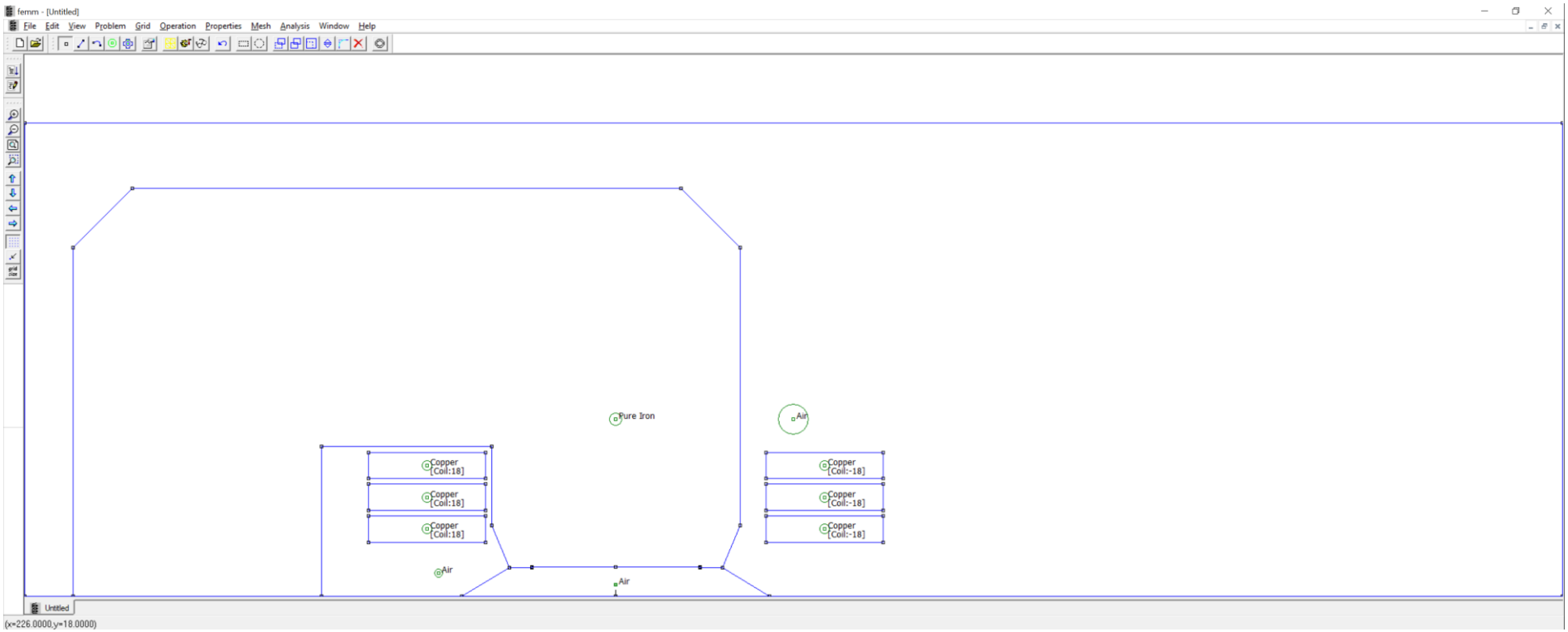
4. Define the geometry (iron, coil, air, background)

```
-- hide lines in post-processor
mi_selectsegment((130+x_yoke[4])/2, y_yoke[4]/2)
mi_selectsegment((-130+x_yoke[15])/2, y_yoke[15]/2)
mi_setsegmentprop("", 0, 1, 1)
mi_clearselected()

-- Boundary conditions on segments
mi_selectrectangle(-500, 0, x_bck, 0, 1)
mi_setsegmentprop("B perpendicular")
mi_clearselected()
mi_selectsegment(x_bck, y_bck/2)
mi_selectsegment((x_bck-500)/2, y_bck)
mi_selectsegment(-500, y_bck/2)
mi_setsegmentprop("B parallel")
mi_clearselected()

-- Zoom out
mi_zoomnatural()
```



assign boundary conditions

We should be here: a good moment for a break?



By the way, no need to model separate conductors or even coil blocks, for such designs

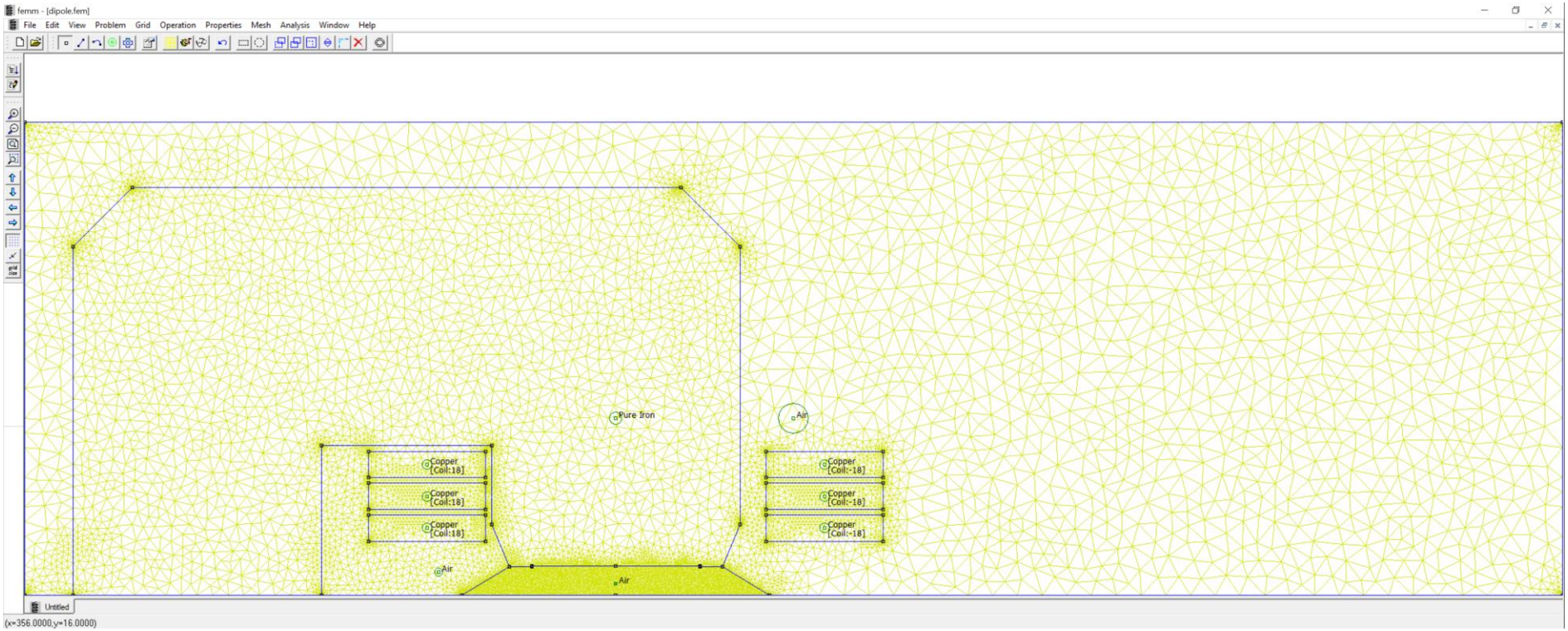
5. Save and mesh

Probably best to save before, in any case you need to save before you mesh

---> Mesh ---> Create Mesh

```
-- Save
mi_saveas(workfolder .. filename .. ".fem")

-- Mesh
mi_createmesh()
```

Now we should be here: another good moment for a break?



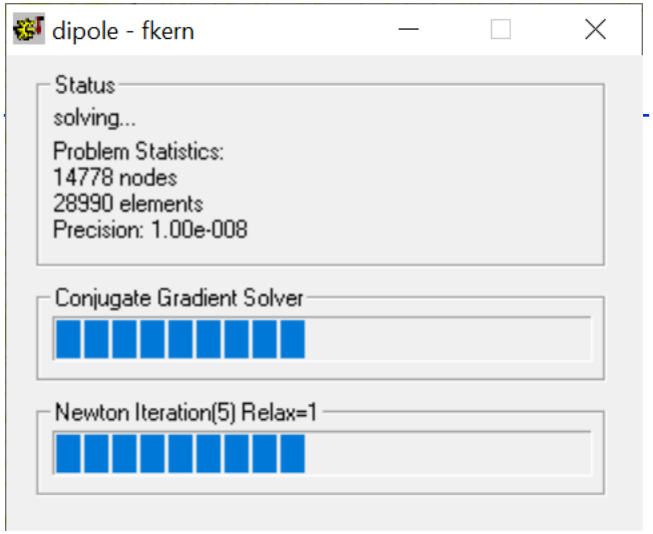
Notice the finer mesh in the gap; sometimes a separate region can be created for the pole tip, to allow for a finer mesh. The background region can have a coarser mesh – the size of the element is inversely proportional to the field gradient.

Other meshing options are available in FEMM, see under segment and region properties

[If you save via the script, the tab on the bottom left might still display “Untitled”]

6. Solve

---> Analysis ---> Analyze




```
-- Solve
mi_analyze()
```

7. Post-processing: typical quantities of interest

Flux lines

Flux density, with or without flux lines

Field in the center

Polarity

Field plots along a line (absolute or in relative w.r.t. the central field)

Allowed harmonics

Energy

Inductance

Lorentz forces on coil

Fringe field

Magnetic forces on yoke

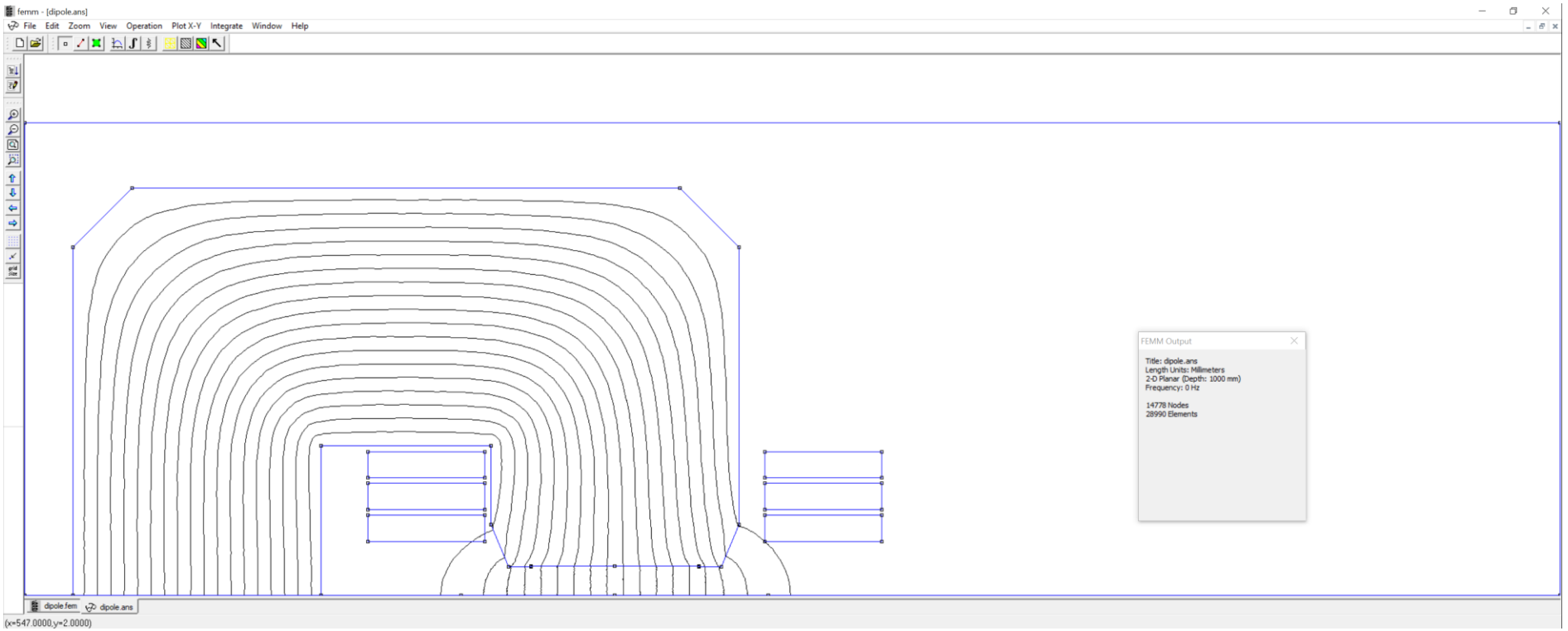
…

7. Post-processing: load solution

---> Analysis ---> View Results

```
-- Post-processing
mi_loadsolution()
```

7. Post-processing: flux lines and B in the center (as a first check)



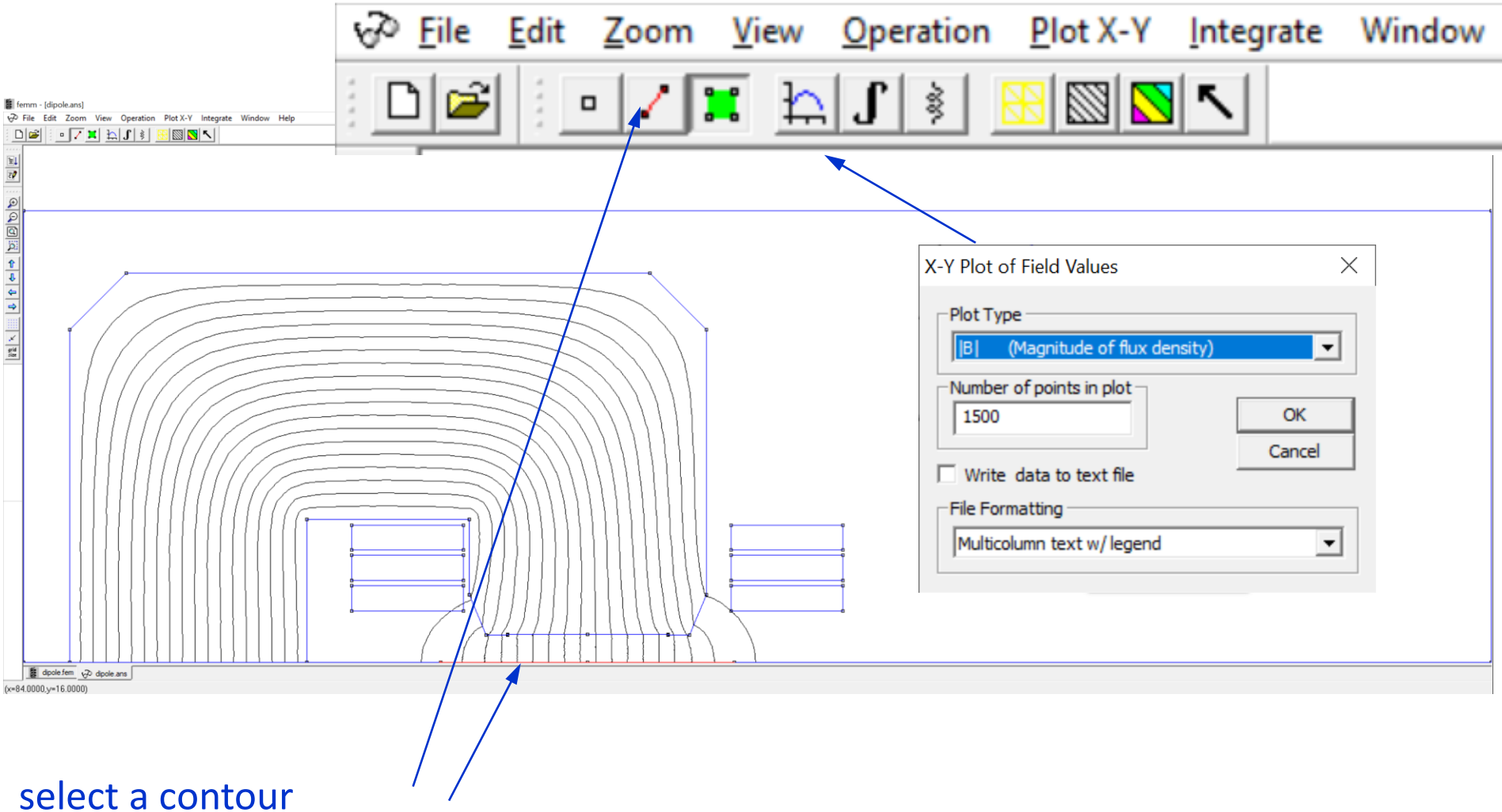
B ≈ 0.98*4e-7*pi*18*6*450/0.050 = 1.19702 T

(about 3‰ difference, we report so many digits just to compare, not because they are significant)

FEMM Output

Point: x=0, y=0
A = 0.158014 Wb/m
|B| = 1.20074 T
Bx = 2.36924e-008 T
By = 1.20074 T
|H| = 955520 A/m
Hx = 0.0188538 A/m
Hy = 955520 A/m
mu_x= 1 (rel)
mu_y = 1 (rel)
E = 573666 J/m^3
J = 0 MA/m^2

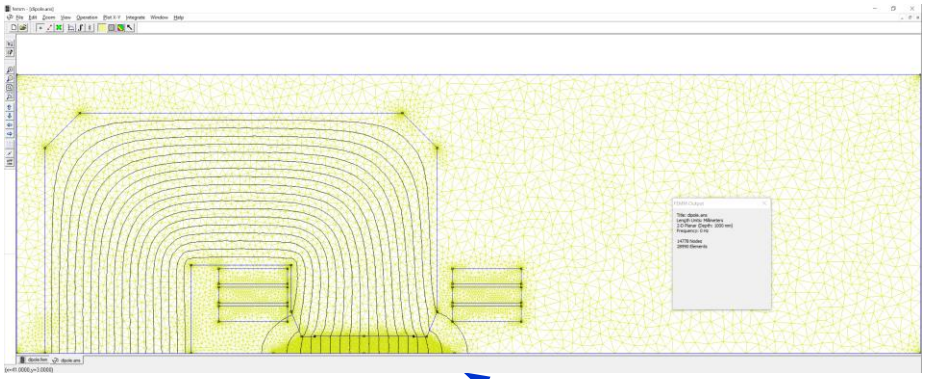
7. Post-processing: line plots

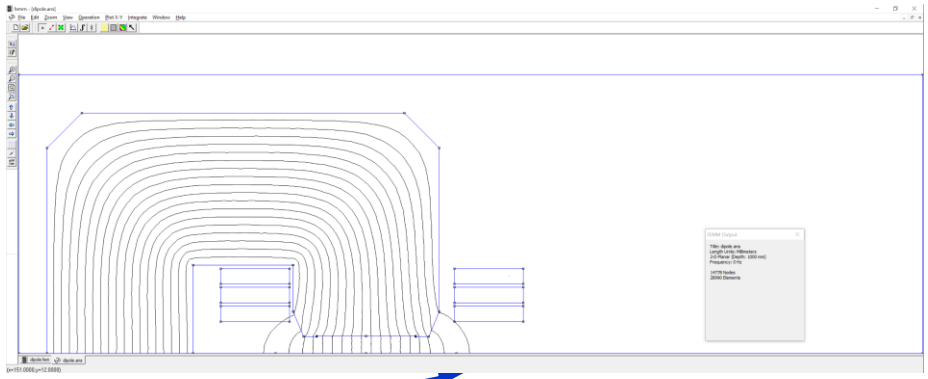


select a contour

- left mouse click for existing point
- right mouse click to add a point (snapping to grid points if convenient)
- Del to remove the last selected node
- Esc to unselect the contour

7. Post-processing: 2D plots



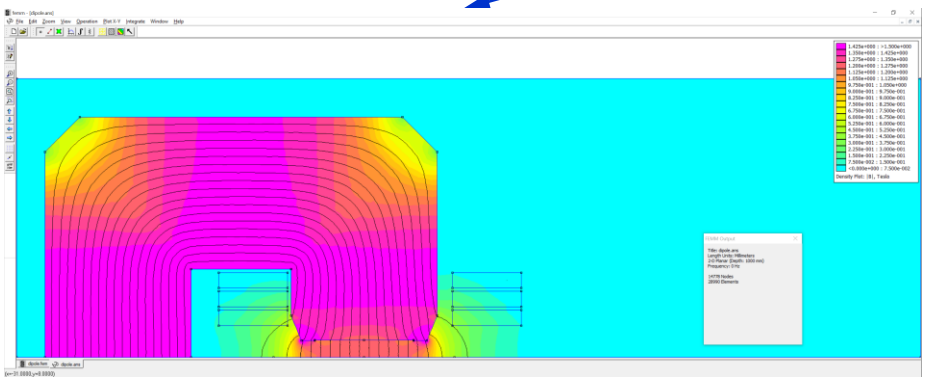


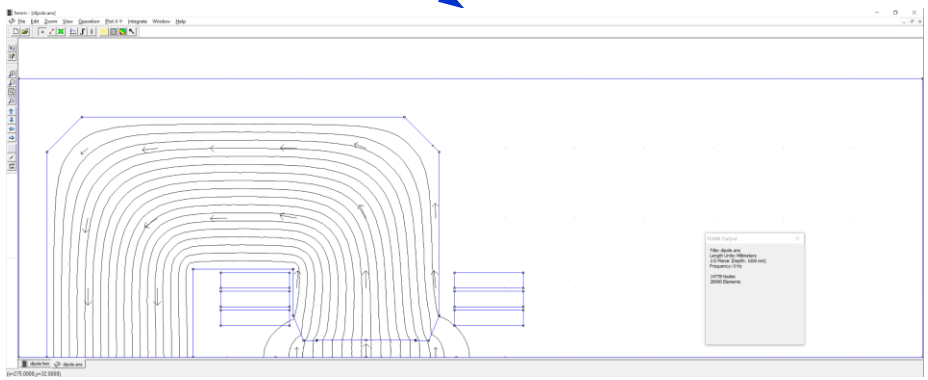
mesh

flux lines

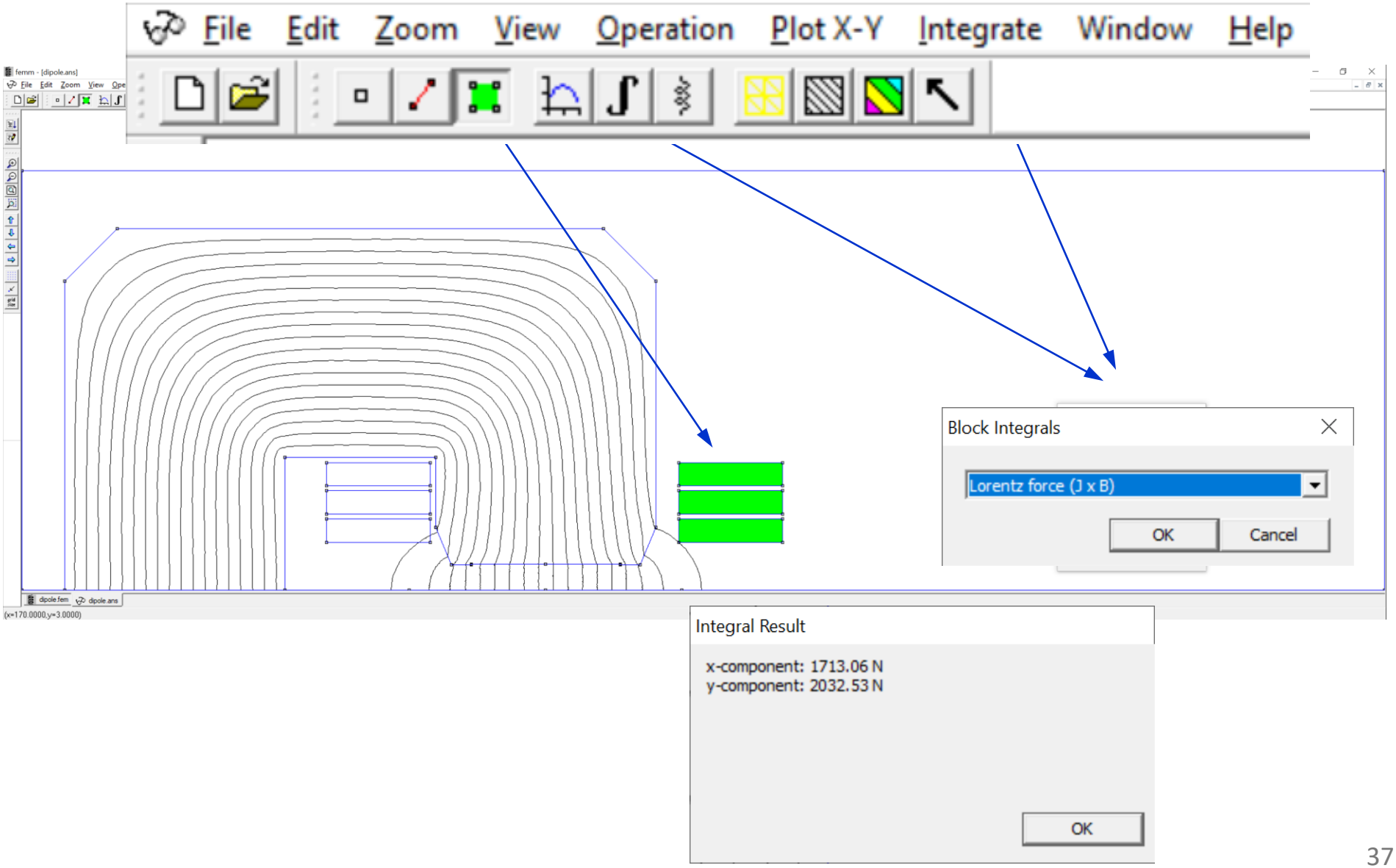
field density

field vectors

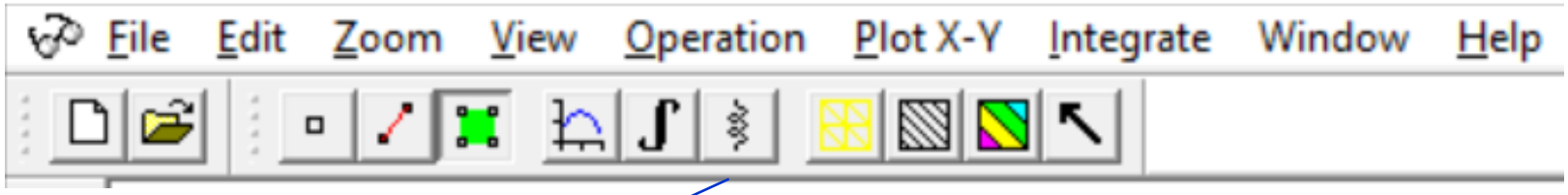


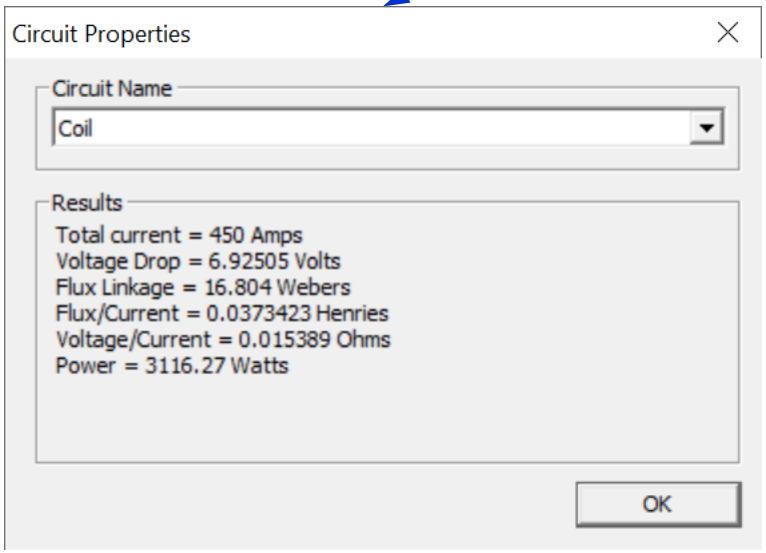


7. Post-processing: integrals over a block



7. Post-processing: inductance (from concatenated flux)





L = 2*37.3 = 74.6 mH/m

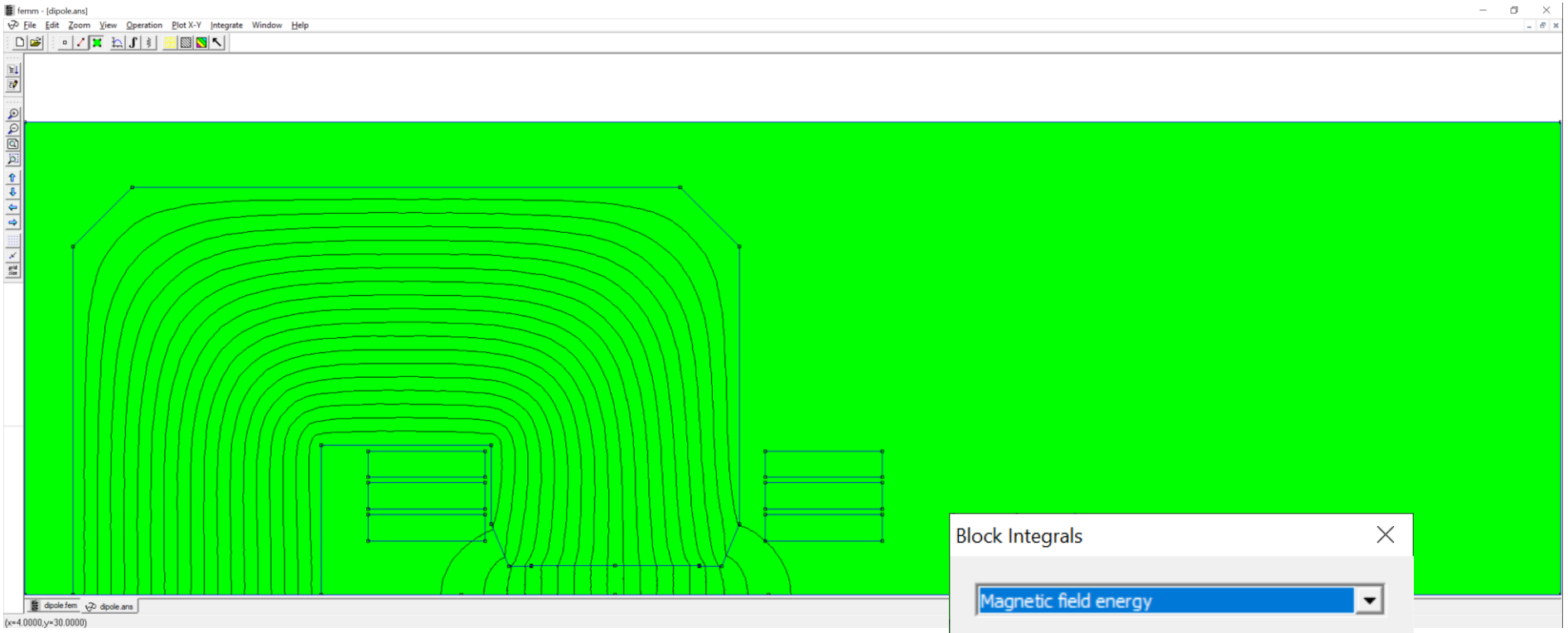
The factor 2 is because we model only half the dipole

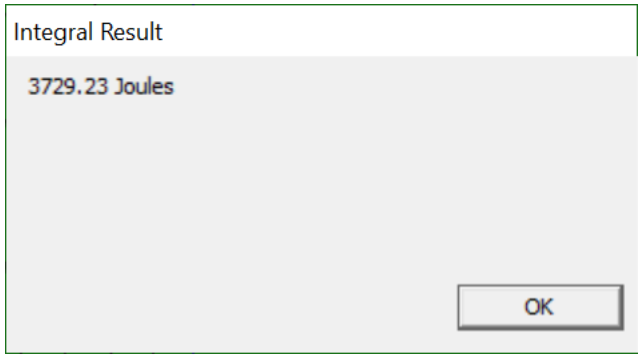
The result is per m, as we set 1 m depth; to get the total inductance, you can multiply by the magnetic length

L ≈ 0.98*4e-7*pi*(18*6)^2*(180+1.2*50)/50 = 68.9 mH/m

In reality, the inductance is nonlinear and it depends on the current (and on the frequency), plus there is the contribution of the coil heads

7. Post-processing: inductance (from energy)





L = 2*2*3729.23/450^2 = 73.7 mH/m

7. Post-processing: a few Lua commands

```
-- Post-processing
-- The flux lines plot is loaded by default
mi_loadsolution()
-- mo_savebitmap(workfolder .. filename .. "_flux.bmp")
mo_savemetafile(workfolder .. filename .. "_flux.emf")

-- Field density plot
B_min = 0
B_max = 1.5
mo_showdensityplot(1,B_min,B_max,0,"bmag")

-- Field at 0,0
A, B1, B2 = mo_getpointvalues(0, 0)
print("B @ x=0; y=0")
print("Bx = ", B1, " T")
print("By = ", B2, " T")

-- Plot field in the aperture
w_GFR = 120
mo_addcontour(-w_GFR/2, 0)
mo_addcontour(w_GFR/2, 0)
-- mo_makeplot(2, 200) -- plot in FEMM
mo_makeplot(2, 50, workfolder .. filename .. "_By_midplane.emf") -- save image
mo_makeplot(2, 50, workfolder .. filename .. "_By_midplane.txt", 0) -- print it to a file
mo_clearcontour()
```

7. Post-processing: a few Lua commands

```
-- Lorentz forces in the coil
mo_selectblock(127+w_coil/2, 100+h_coil/2)
mo_selectblock(127+w_coil/2, 100+h_coil/2-(h_coil+5))
mo_selectblock(127+w_coil/2, 100+h_coil/2-2*(h_coil+5))
Fx = mo_blockintegral(11)
print("Fx = ", Fx, " N")
Fy = mo_blockintegral(12)
print("Fy = ", Fy, " N")
mo_clearblock()

-- Energy
mo_groupselectblock()
U = mo_blockintegral(2)
print("Energy = ", U, " J")
mo_clearblock()

-- Inductance
-- --> from concatenated flux
curr, volts, flux_re = mo_getcircuitproperties("Coil")
print("Fy = ", flux_re, " N")
L_fl = 2*flux_re/curr
print("Inductance (from concatenated flux) = ", L_fl*1000, " mH")
-- alternative: select all coil blocks then get the flux linkage as mo_blockintegral(0)
-- --> from energy
mo_groupselectblock()
U = mo_blockintegral(2)
mo_clearblock()
L_en = 2*2*U/curr^2
print("Inductance (from energy) = ", L_en*1000, " mH")
```

7. Post-processing: allowed harmonics

multipoles_femm.lua

(this is just the start, with the header to set the parameters)

```
--
-- LUA script to compute multipoles in FEMM
--
-- Few standard cases are considered:
--  * dipole 180 deg (ex. C shape)
--  * dipole 90 deg (ex. H shape)
--  * quadrupole 45 deg (ex. standard symmetric quadrupole)
--
-- In all cases, the center is 0, 0 and the skew coefficients are 0
--
-- The script computes two sets of multipoles:
--  * one from A (the vector potential)
--  * another one from a radial projection of B
-- They should be the same, so the difference in a way shows
-- how much to trust these numbers; the ones from A should be better,
-- as this is the finite element solution without further manipulations
-- (derivation, radial projection) while B is rougher (linear elements,
-- so B is constant over each triangle), but then it's smoothed out in the postprocessor
--

case_index = 1
-- 1 ===> dipole 180 deg (ex. C shape)
-- 2 ===> dipole 90 deg (ex. H shape)
-- 3 ===> quadrupole 45 deg (ex. standard symmetric quadrupole)

nh = 15 -- number of harmonics
np = 256 -- number of samples points
R = 2*25/3 -- reference radius
Rs = 0.95*25 -- sampling radius, can be the same as R or the largest still in the air
```

List of input files

dipole.lua: Lua script for this example of a resistive C dipole

quad_PXMQNAHNAP.lua: Lua script for a resistive quadrupole (the same that you will measure)

corrector_PXMCCATWAP.fem: FEMM file for a resistive dual plane corrector (the same that you will measure) [courtesy of A. Newborough]

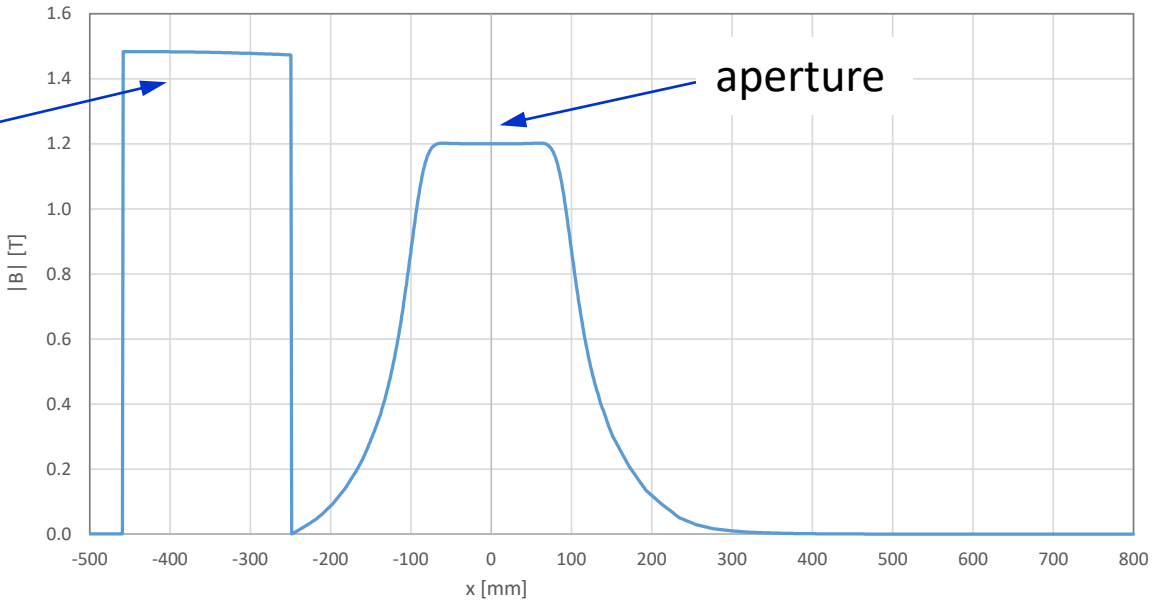
sector_dipole.lua: Lua script for a superconducting sector dipole [courtesy of S. Izquierdo Bermudez]

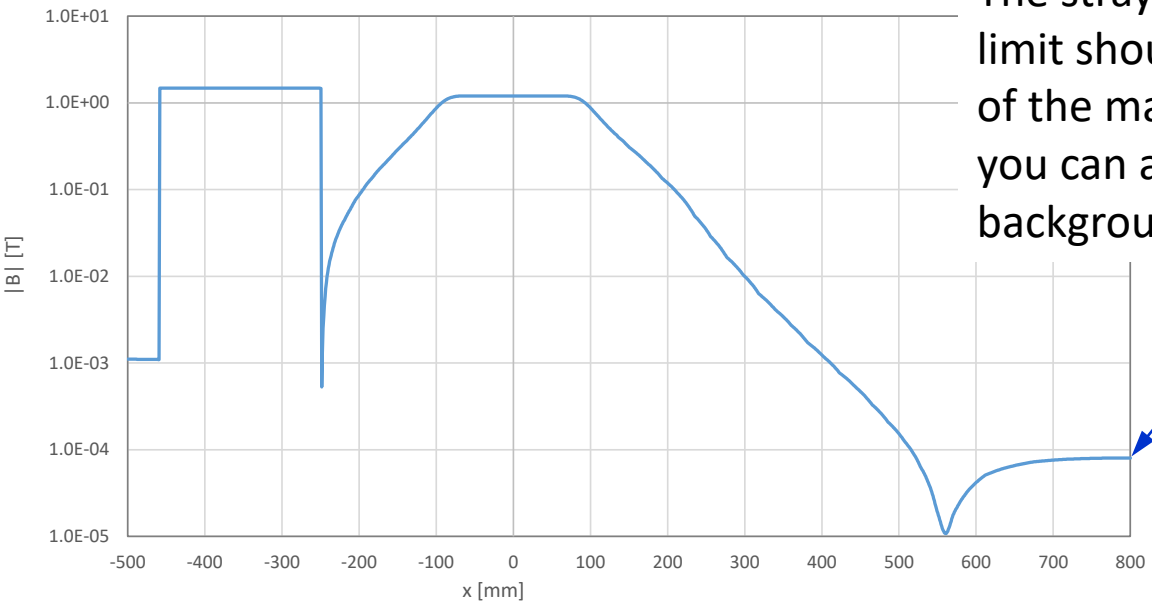
multipoles_femm.lua: Lua script to compute multipoles

Thank you

Bonus

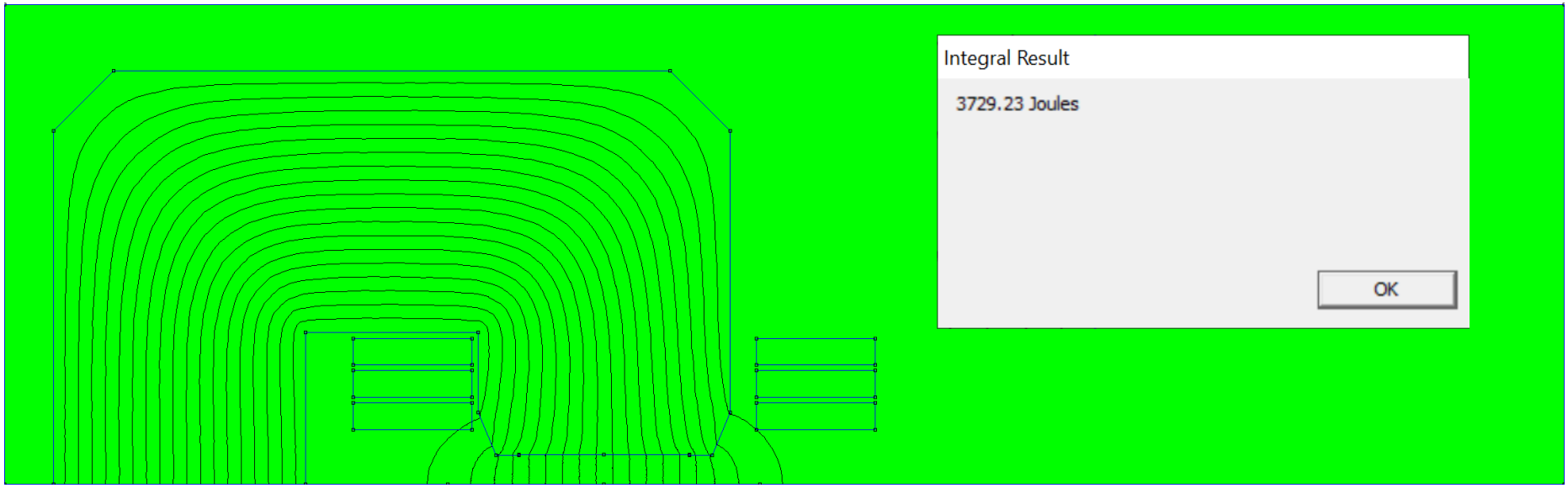
Field along the midplane and stray field

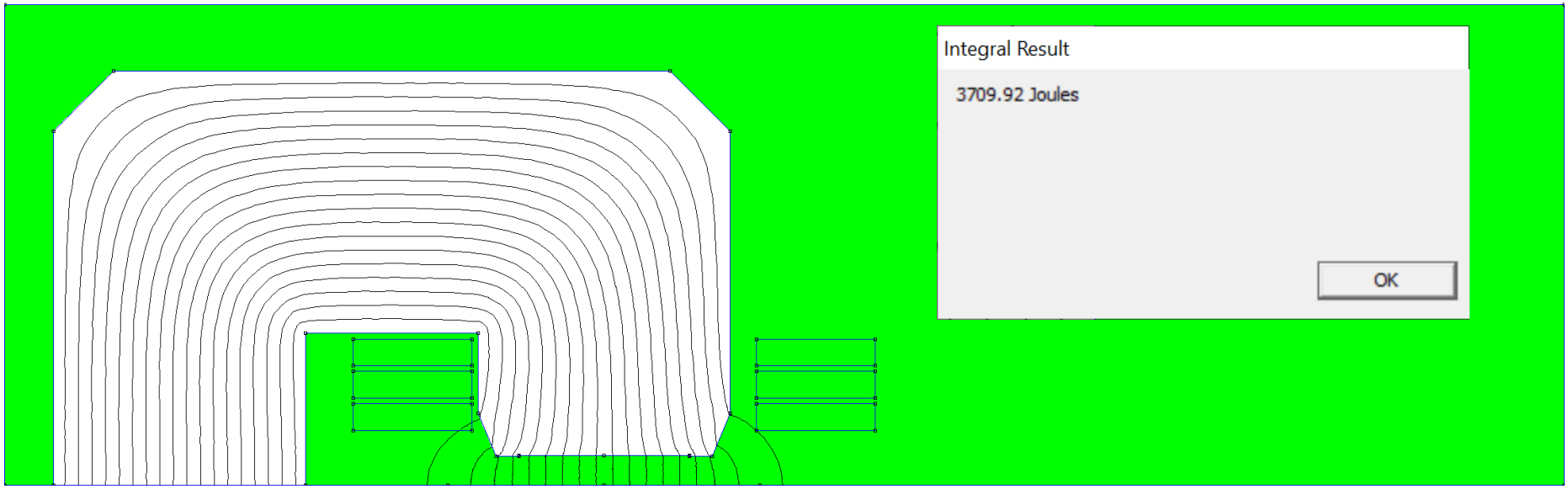




The stray field at the background limit should be a few ‰ (or less) of the main field in the aperture – you can also check varying the background dimensions

Energy is in the air!

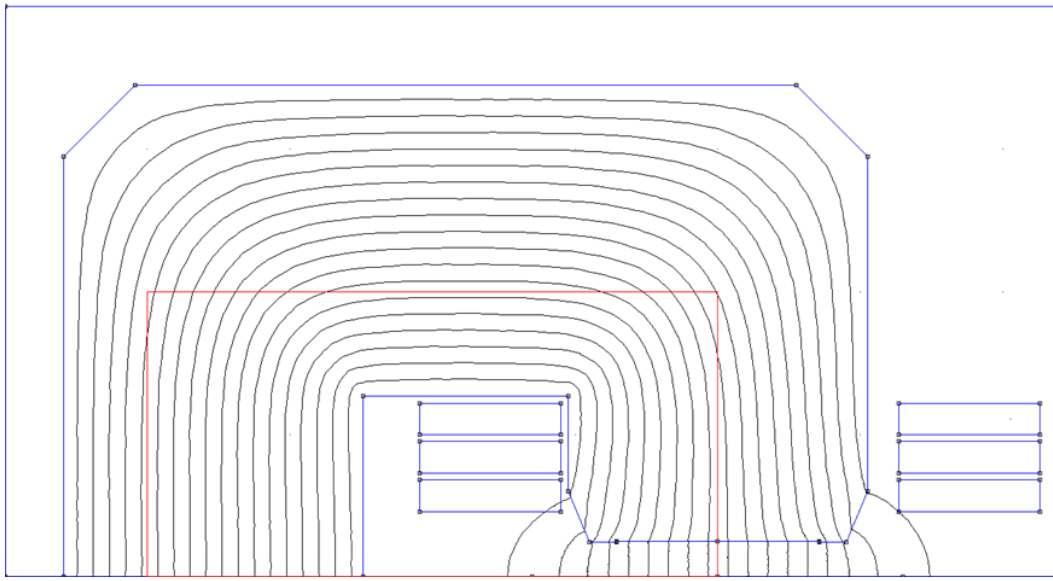


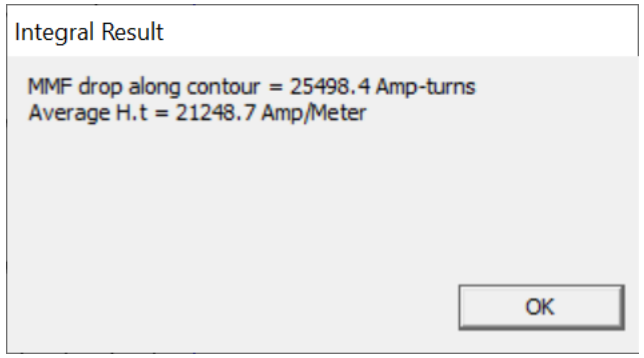


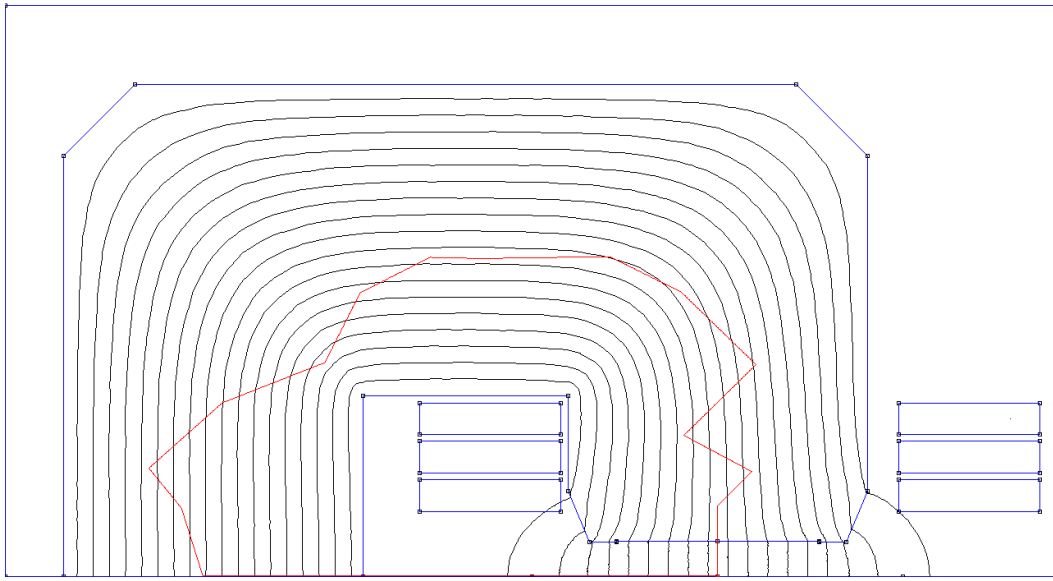
Ampere's law in action

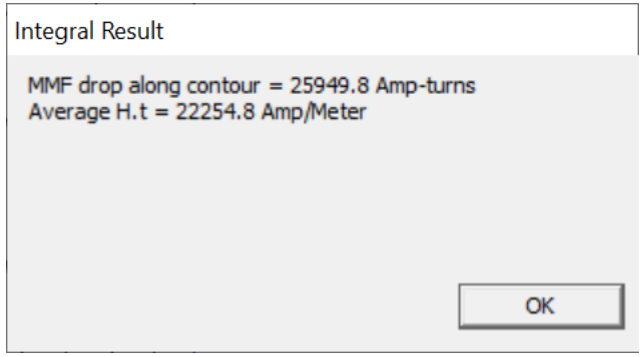
NI = 450*18*3 = 24300 A

5-7% difference (with this mesh)

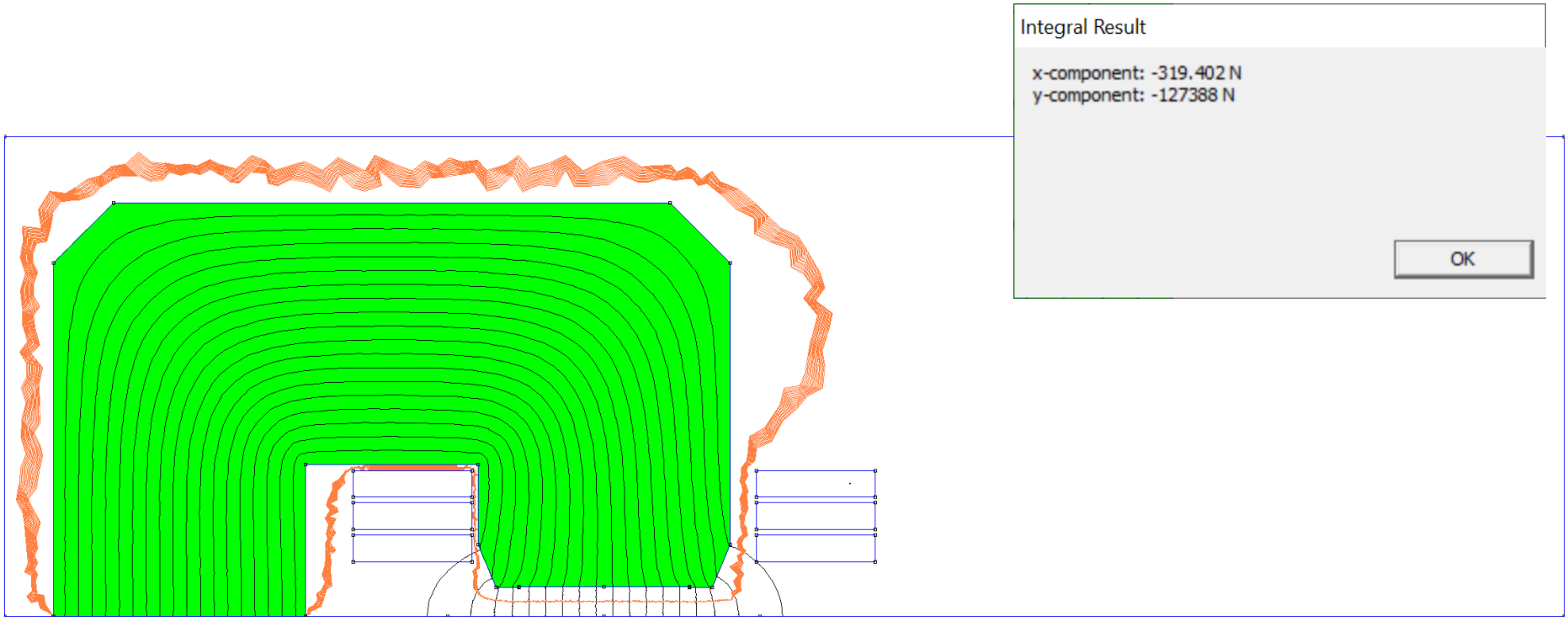








Force on iron via weighted stress sensor

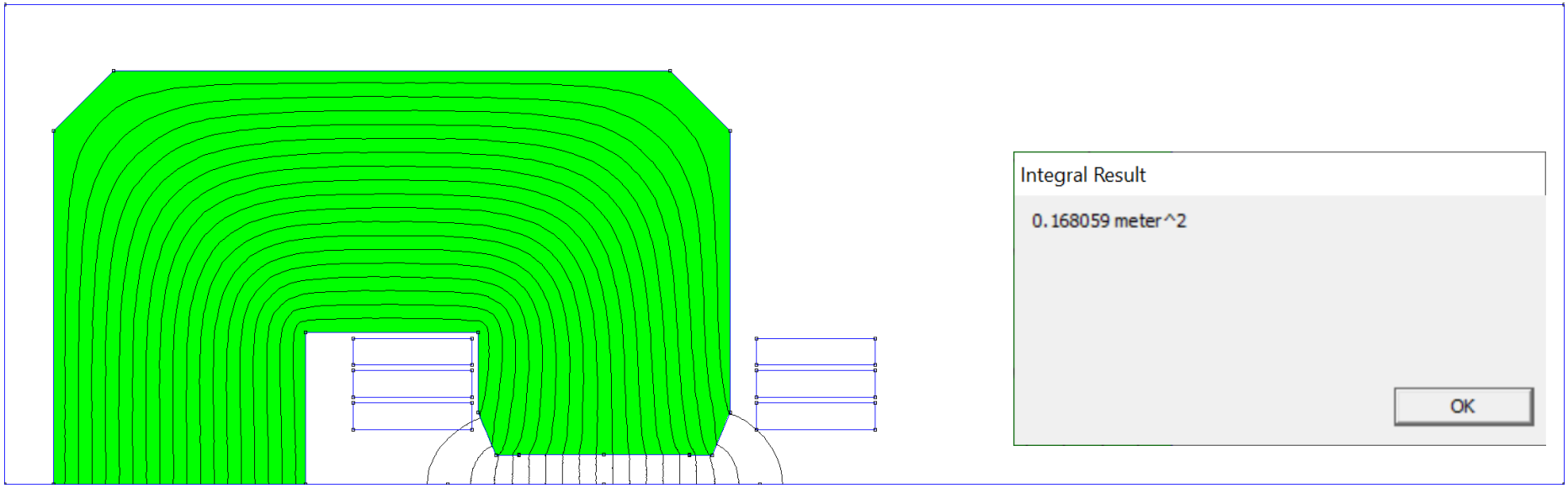


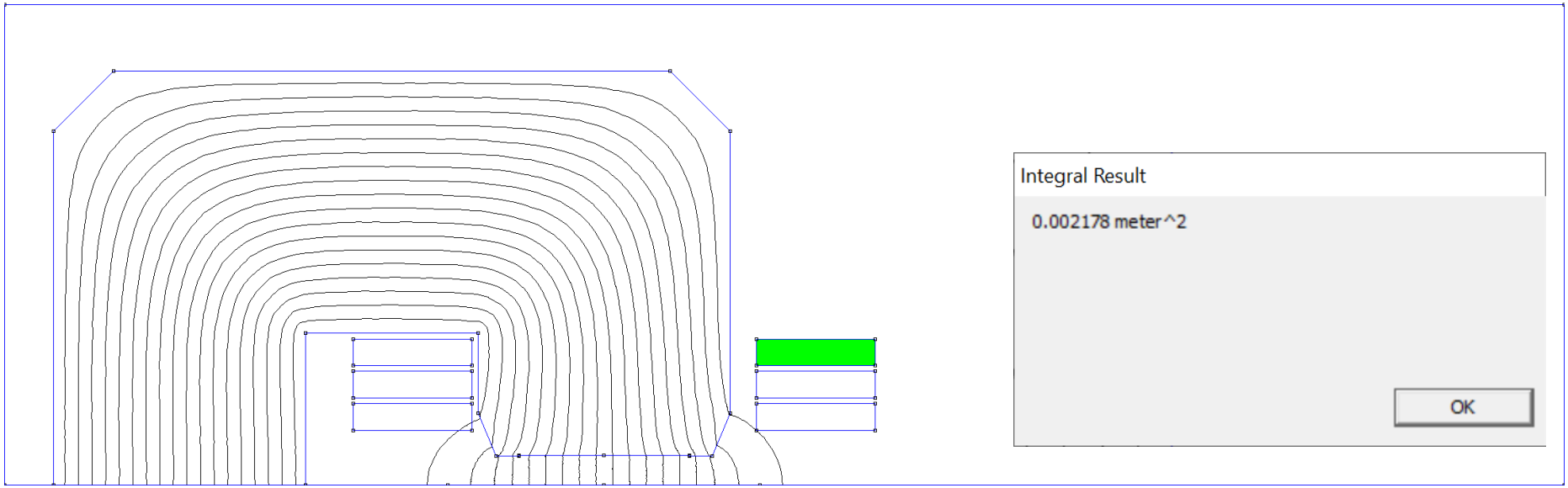
check with magnetic pressure on pole

F ≈ (B^2)/(2*mu0)*w_pole = (1.20^2)/(2*4e-7*pi)*(0.180+1.2*0.05) = 137510 N/m

8% difference, but the order of magnitude is correct

Getting areas





Field smoothing: on by default (and leave it on)

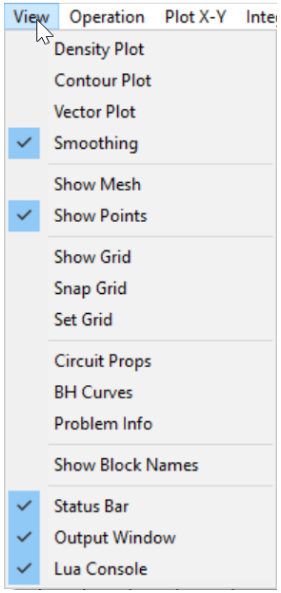
6.3 Field Smoothing

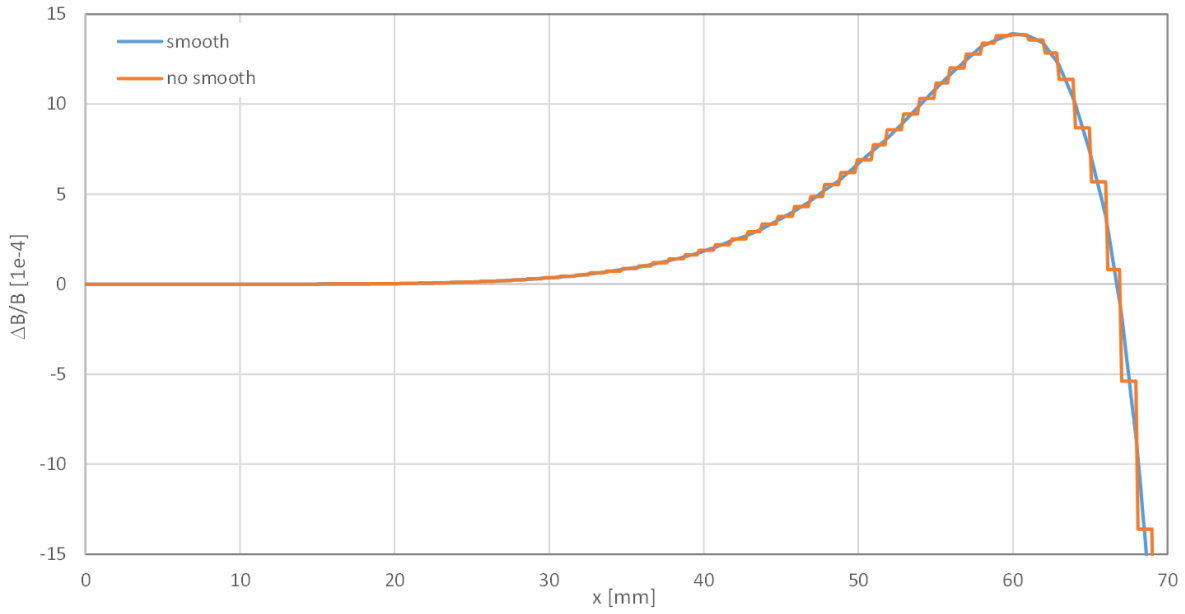
Since first-order triangles are used by FEMM, the resulting solution for B and H obtained by differentiating A is constant over each element. If the raw B and H are used by the postprocessor, density plots of B and 2-D plots of field quantities along user-defined contours look terrible. The values of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

The use of smoothing to recover the accuracy lost by differentiating A is known as *superconvergence*. Of the greatest interest to FEMM are so-called "patch recovery" techniques. The basic idea is the the solutions for B are most accurate at the centroid of the triangular element (known as its *Gauss Point*). One desires a continuous profile of B that can be interpolated from nodal values, in the same way that vector potential A can be represented. The problem is, the "raw" solution of B is multivalued at any node point, those values being the different constant values of B in each element surrounding the node point. The general approach to estimating the "true" value of B at any node point is to fit a least-squares plane through the values of B at the Gauss points of all elements that surround a node of interest, and to take the value of the plane at the node point's location as its smoothed value of B [16].

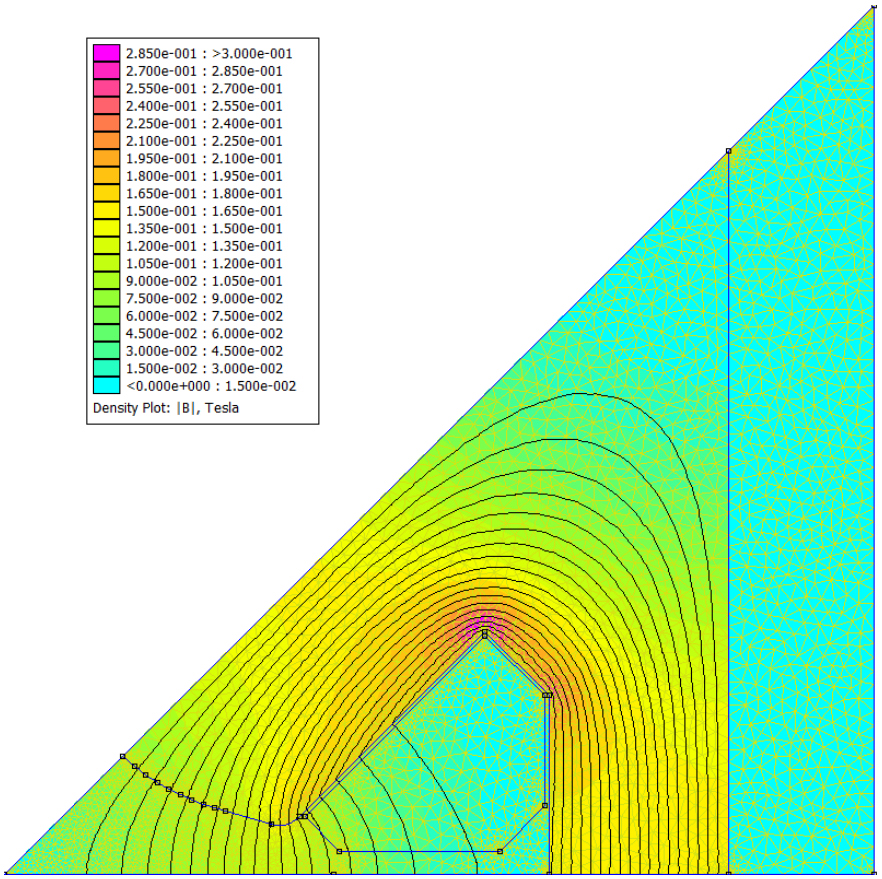
However, this approach to patch recovery has a lot of shortcomings. For the rather irregular meshes that can arise in finite elements, the least-squares fit problem can be ill-condition, or even singular, at some nodes in the finite element mesh. Furthermore, the superconvergence solution can actually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

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





Results (gradient and harmonics) as a function of mesh density



I	[A]	10
turns per pole	[/]	85
r (aperture)	[mm]	29
R (reference)	[mm]	20
Rs (sampling)	[mm]	28
B' (formula)	[T/m]	2.5402

mshf (mesh factor)	[/]	1	0.5	0.25
npAB (points in hyperb. part)	[/]	10	20	40
B' (By/x @ 30 mm)	[T/m]	2.5489	2.5386	2.5410
B' (B2/R)	[T/m]	2.5336	2.5360	2.5366
B' (By/x @ 30 mm) / B' (formula)	[/]	1.0035	0.9994	1.0003
B' (B2/R) / B' (formula)	[/]	0.9974	0.9984	0.9986
b6	[1e-4]	2.2	0.9	0.6
b10	[1e-4]	-0.2	0.1	0.1
b14	[1e-4]	0.0	0.0	0.0