FEMM tutorial
Numerical design of a C-shape dipole

Attilio Milanese and Jérémie Bauche

attilio.milanese@cern.ch
jeremie.bauche@cern.ch

John Adams Institute
Accelerator Course
18 Jan. 2024

Support material for this tutorial on the INDICO page

1. These Powerpoint slides:
   "JAI_course_2024_FEMM_tutorial.pptx" (+ .pdf)

2. An Excel spreadsheet: "C-shape_dipole_templates.xlsx" for:
   - Parametric definition of the yoke and coil geometries
   - Post-processing of field homogeneity and harmonics

   (input parameters in green cells)

3. The LUA script of HIE-ISOLDE dipole we'll use as case study:
   "HIE-ISOLDE_dipole.lua" (+ .txt) - for pre-processor only

4. The LUA script "multipoles_femm.lua" (+ .txt), for the computation of the field harmonics

The program of today's tutorial

1. Introduction to the GUI of the FEMM software
2. Methodology to create and analyse a FEMM model
3. Walk-through example of the FEMM model of a C-shape dipole (from the HIE-ISOLDE facility at CERN) <u>with the GUI</u>
   - Creation of the model
   - Analysis and optimisation
4. Brief introduction to scripts with LUA language

To download the software, go to www.femm.info

HomePage:Finite Element Meth ×

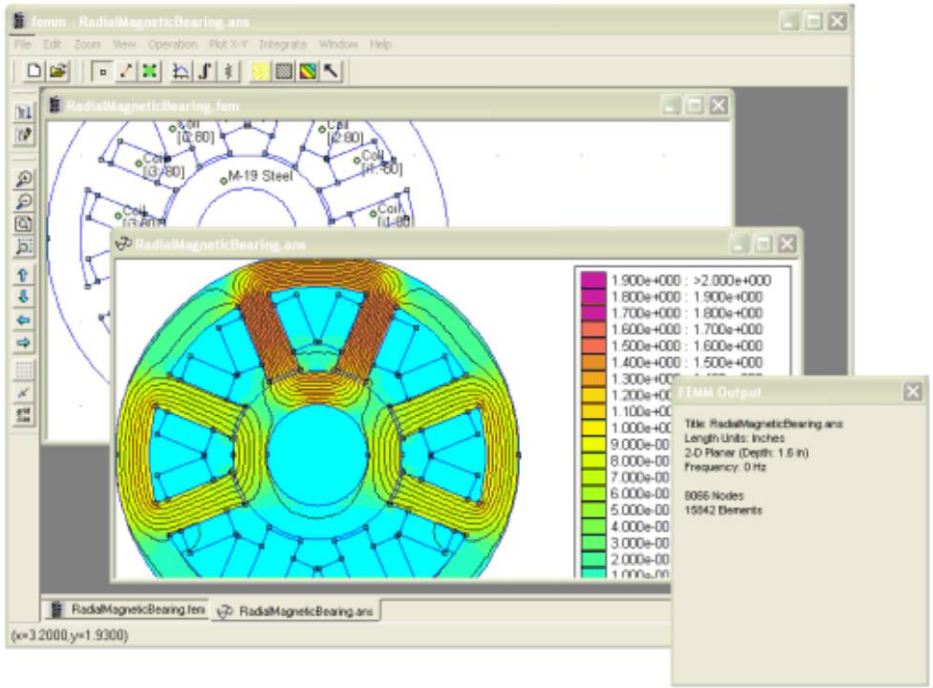
https://www.femm.info/wiki/HomePage 130% Search

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



FEMM is a freeware!

To install the software

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 <u>Windows</u> users: *just install FEMM*

For <u>Linux or Mac</u> users: *see here*

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

A few extra references (for magnet design and this tutorial)

- CAS on Resistive and Superconducting Magnets, 2023

  https://indico.cern.ch/event/1227234/contributions/

  See “RT magnet design, fabrication and testing” lectures from Attilio

  See also “Hands-on - Block 2 - Resistive magnet design”

- J. Bauche and A. Aloev, Design of the beam transfer line magnets for HIE-ISOLDE, IPAC2014 conference, Dresden

  https://accelconf.web.cern.ch/IPAC2014/papers/tupro104.pdf

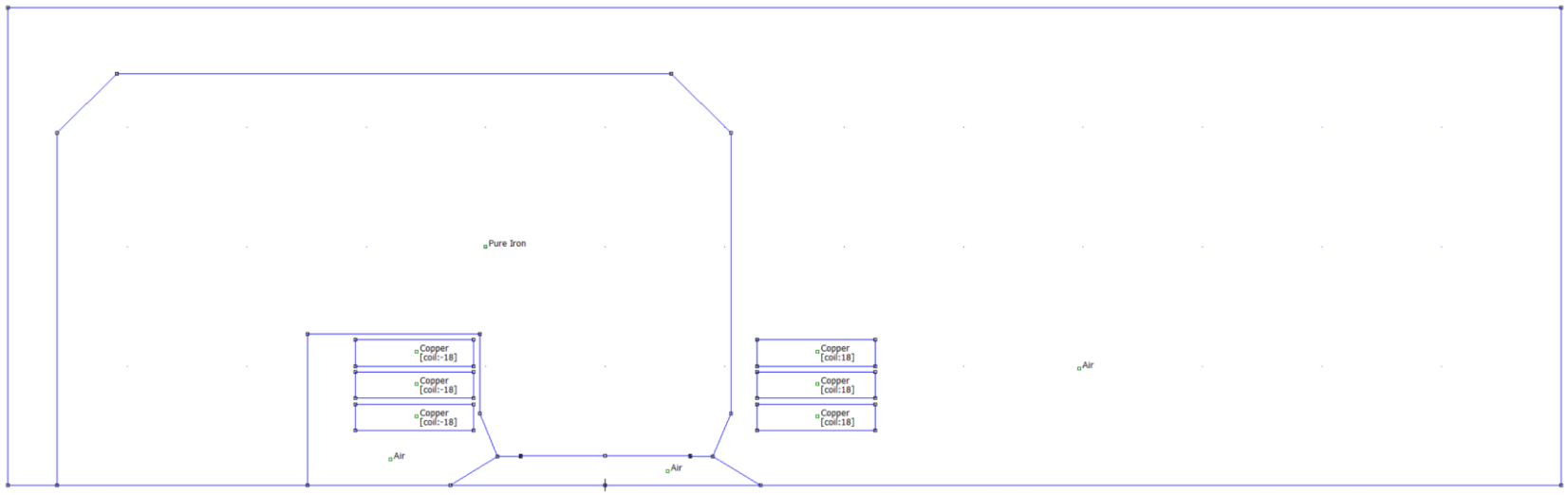
  This describes the bending magnet of the tutorial

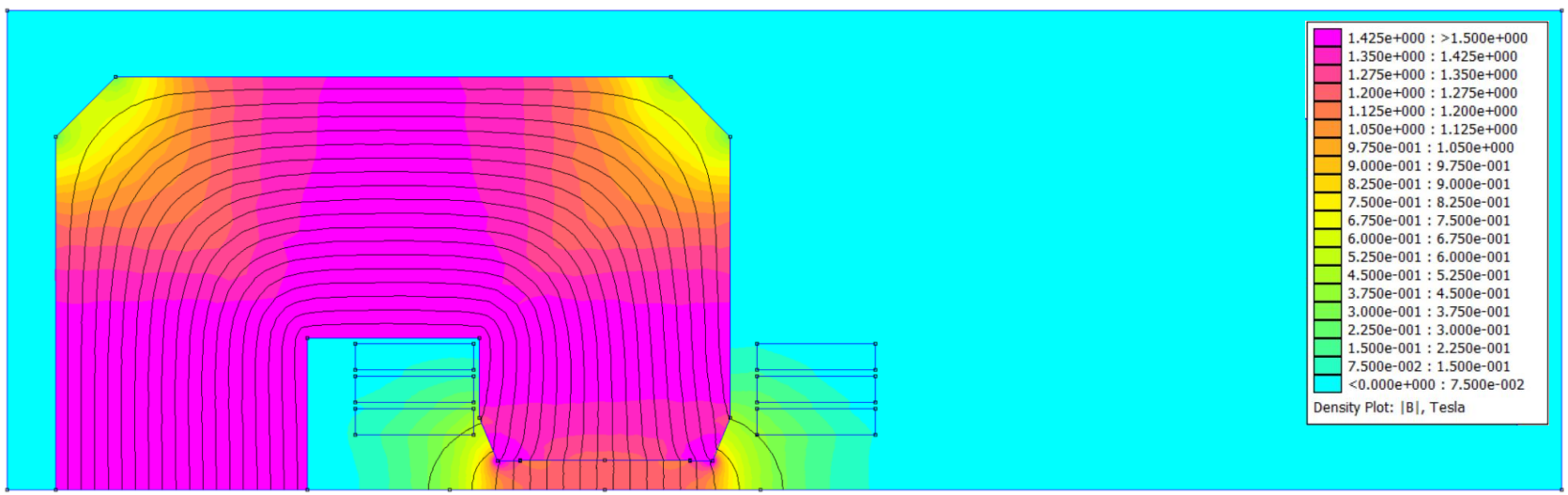
- T. Zickler, Numerical design of a normal-conducting, iron-dominated electro-magnet using FEMM 4.2, JUAS2016

  https://indico.cern.ch/event/471931/contributions/1149654

  [though you need to ask for access now]

Here is the geometry in the FEMM preprocessor and the solution in the postprocessor of the HIE-ISOLDE dipole (2D)





The details of the geometry of the HIE-ISOLDE dipole, for the tutorial with GUI

yoke

	x [mm]	y [mm]
1	0	25
2	71	25
3	71	24.2
4	90	24.2
5	105	60
6	105	295
7	55	345
8	-409	345
9	-459	295
10	-459	0
11	-249	0
12	-249	127
13	-105	127

coil

Number of coil pancakes: **3**
(i.e. 6 blocks in the 2D cross-section)

Outer top coil block, top left corner:
(127,122) mm

Coil block size:

$\mathbf{w_{coil} = 99\ mm}$
$\mathbf{h_{coil} = 22\ mm}$

Distance between coil blocks

Vertical spacing = 5 mm
Horizontal spacing = 237 mm

Coil block ampere-turns:

NI = 18 × 450 A *(I max)*
NI = 18 × 110 A *(I min)*

Overall, this is a short decalogue for a FEMM simulation

1. Create a new file, “magnetics” category
2. Set main problem parameters (ex. planar, mm, 0 frequency)
3. Define the geometry (iron, coils, air, background)
4. Load and set material properties (on regions)
5. Set circuits properties
6. Set and apply boundary conditions on lines (see next slide)
7. Mesh and refine mesh if needed
8. Solve
9. Postprocess
10. Perform some sanity checks (shape of flux lines, saturation, sensitivity to background, mesh, etc.)

Hot keys and mouse button actions for the preprocessor

Keys

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)

Mouse

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

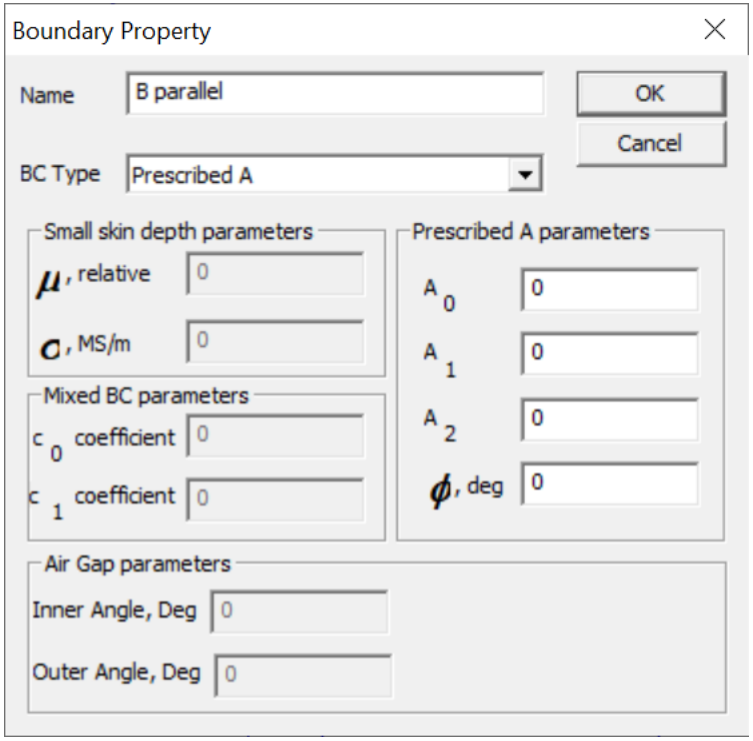
Block Label Mode	
Action	Function
Left Button Click	Create a new block label at the current mouse pointer location
Right Button Click	Select the nearest block label
Right Button DblClick	Display coordinates of the nearest block label

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

[from the FEMM manual]

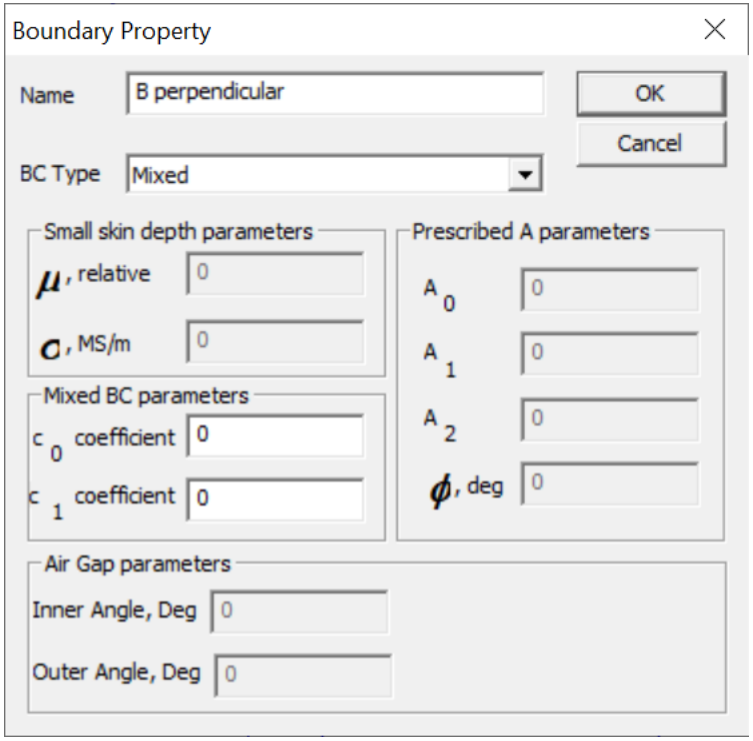
How to set the two boundary properties that we use in FEMM

B parallel



Also called “Dirichlet” boundary condition

B perpendicular



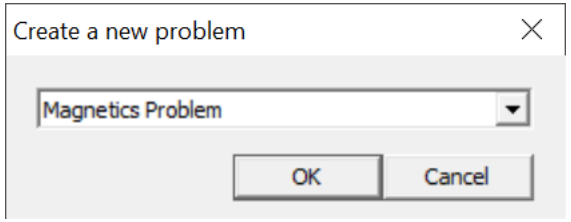
Also called “Neumann” boundary condition

SPARE SLIDES (from CAS 2023):

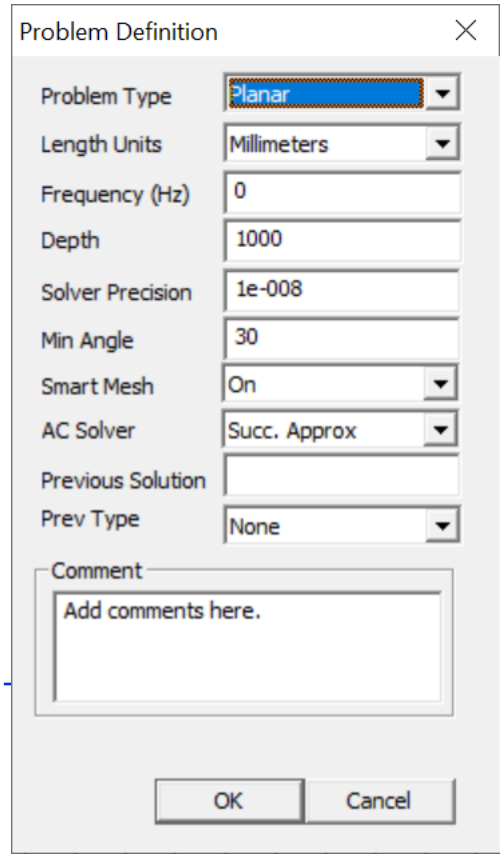
Walk-through of the C-shape dipole modelling, with details of GUI and of LUA scripts

1. Create a new file and set main problem parameters

---> File ---> New



---> Problem



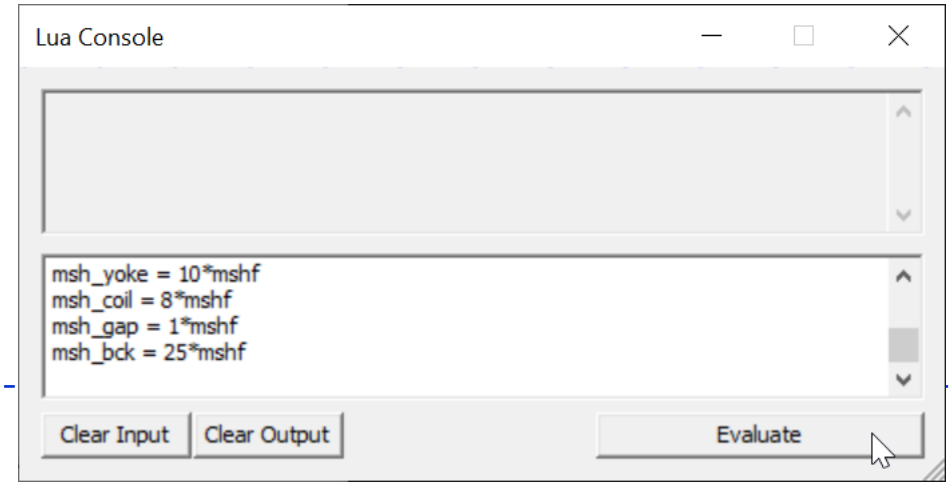
1 m depth, so results (energy, inductance, force, …) will be per m length

```
-- 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



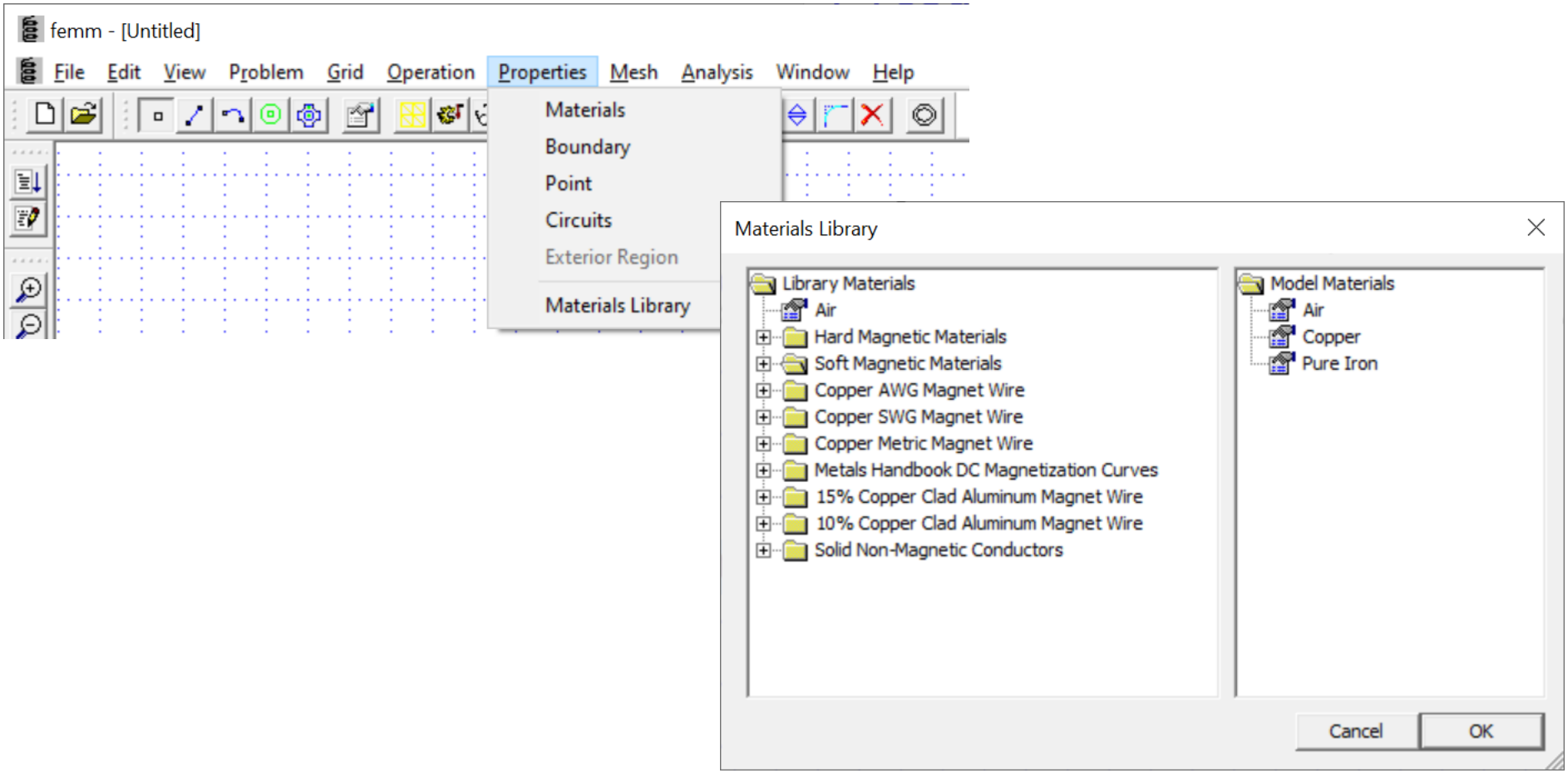

```
-- A few variables
w_coil = 99
h_coil = 22
x_bck = 800
y_bck = 400
workfolder = "C:\\temp\\"
filename = "dipole"

-- 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
```

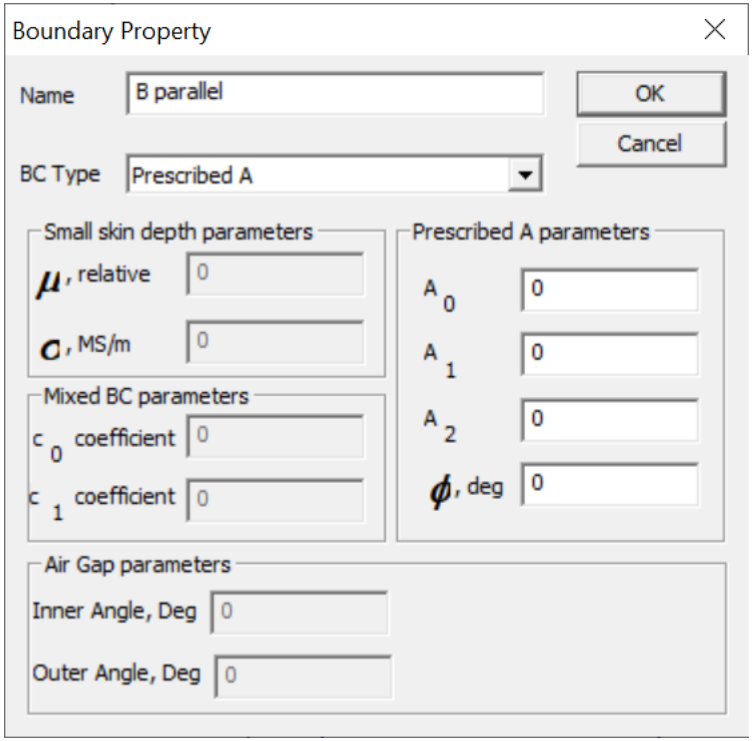
mesh size define via a scaling factor

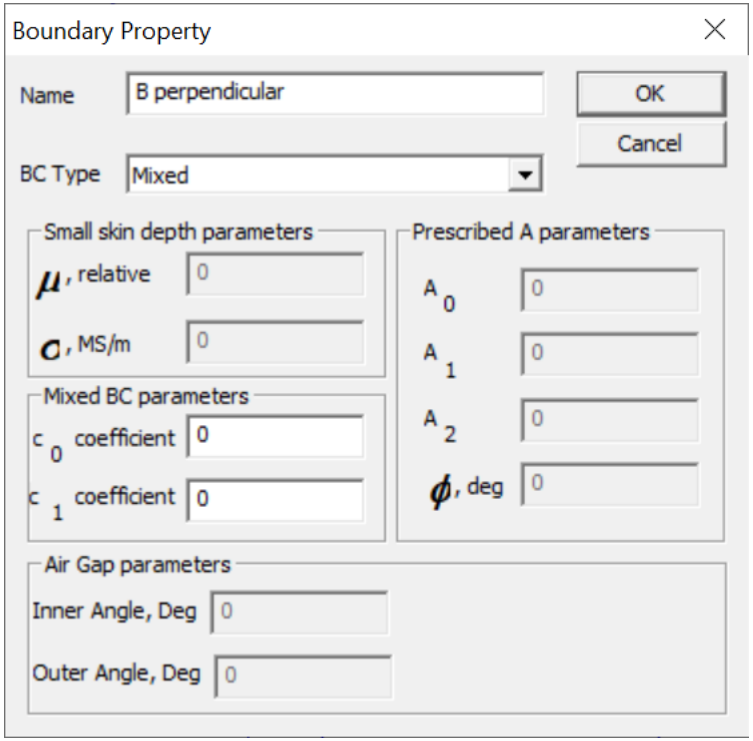
3. Load or prepare material properties (from the available library), boundary conditions and circuit elements




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

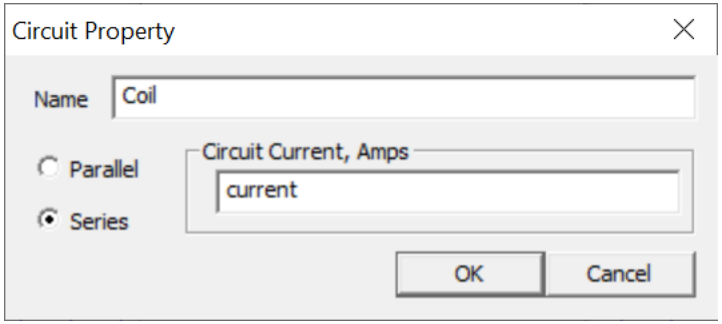
3. Load or prepare material properties (from the available library), boundary conditions and circuit elements






```
-- 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)
```

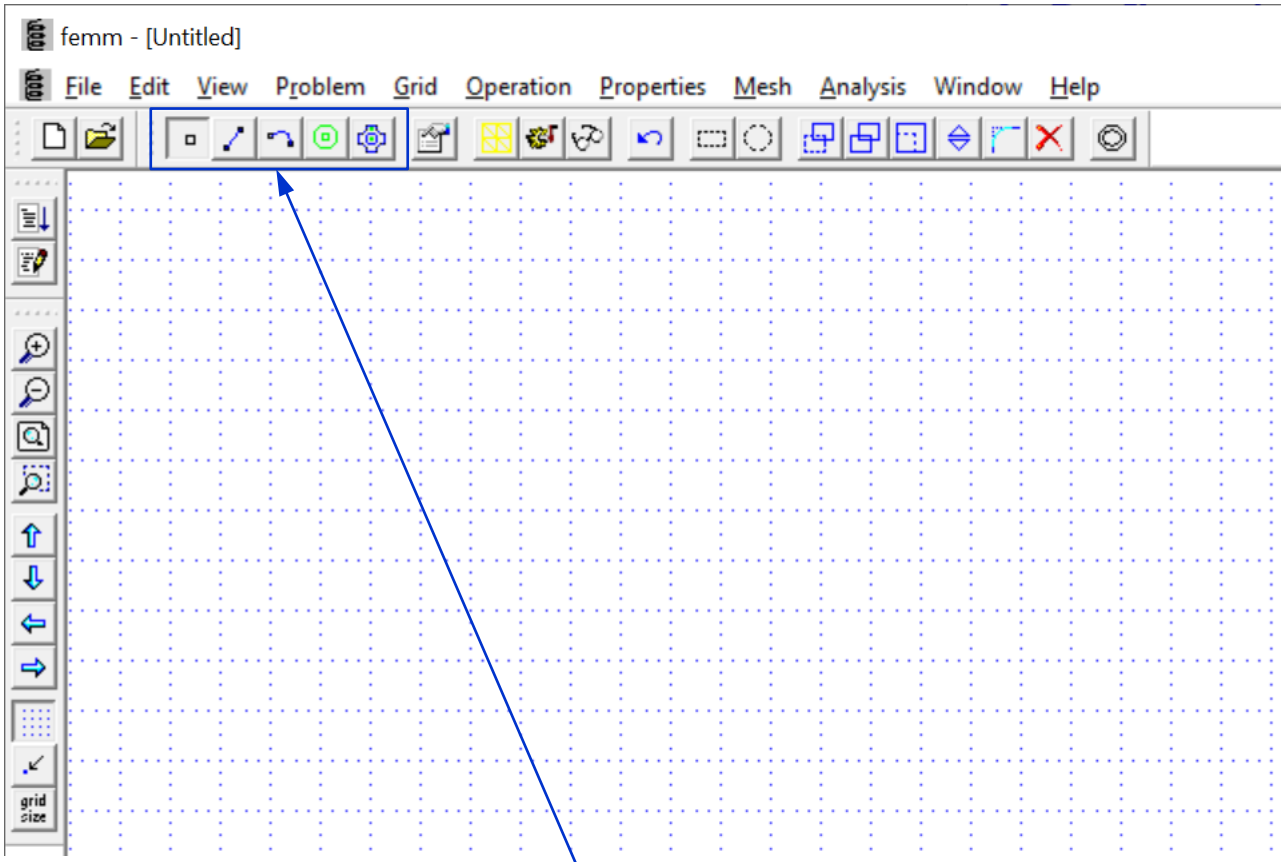
3. Load or prepare material properties (from the available library), boundary conditions and circuit elements



"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)

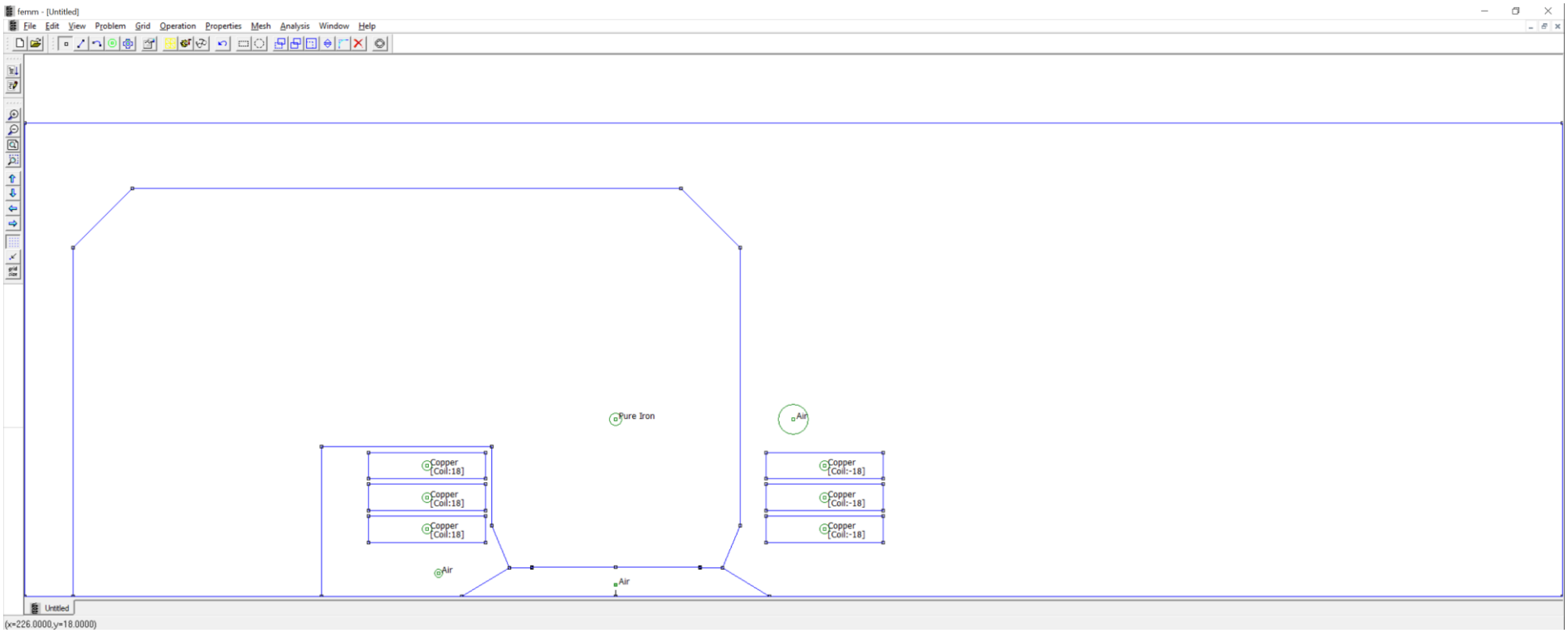
```
-- hide lines in post-processor
mi_selectsegment((130+x_yoke[4])/2, y_yoke[4]/2)
mi_selectsegment((-130+x_yoke[15])/2, y_yoke[15]/2)
mi_setsegmentprop("", 0, 1, 1)
mi_clearselected()

-- 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

The model is now set in the pre-processor, we are ready to mesh and compute the solution



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

5. Save and mesh

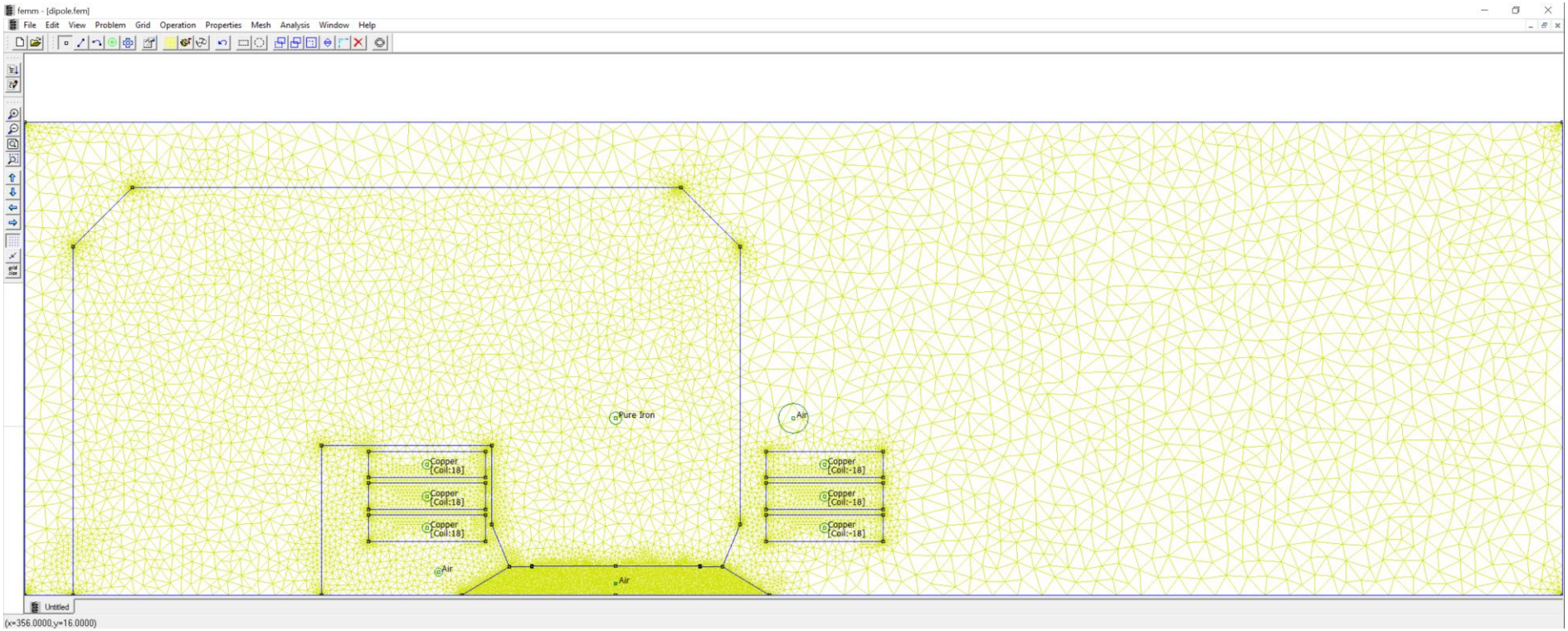
Probably best to save before, in any case you need to save before you mesh

---> Mesh ---> Create Mesh

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

-- Mesh
mi_createmesh()
```

This is the meshed model



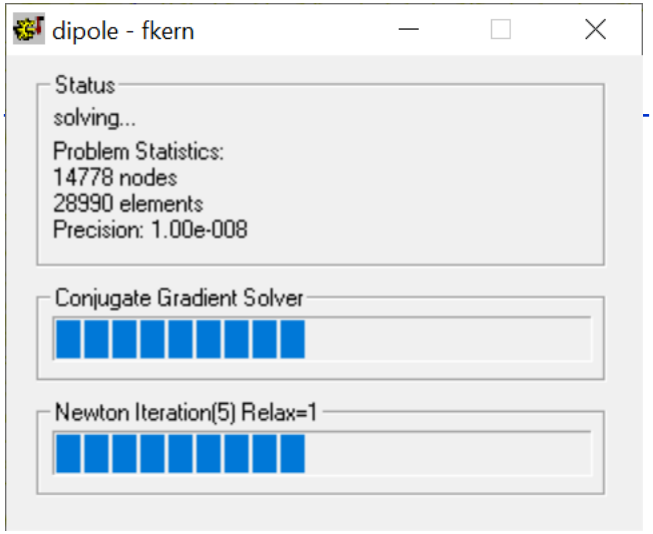
Notice the finer mesh in the gap; sometimes a separate region can be created for the pole tip, to allow for a finer mesh. The background region can have a coarser mesh – the size of the element is inversely proportional to the field gradient.

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

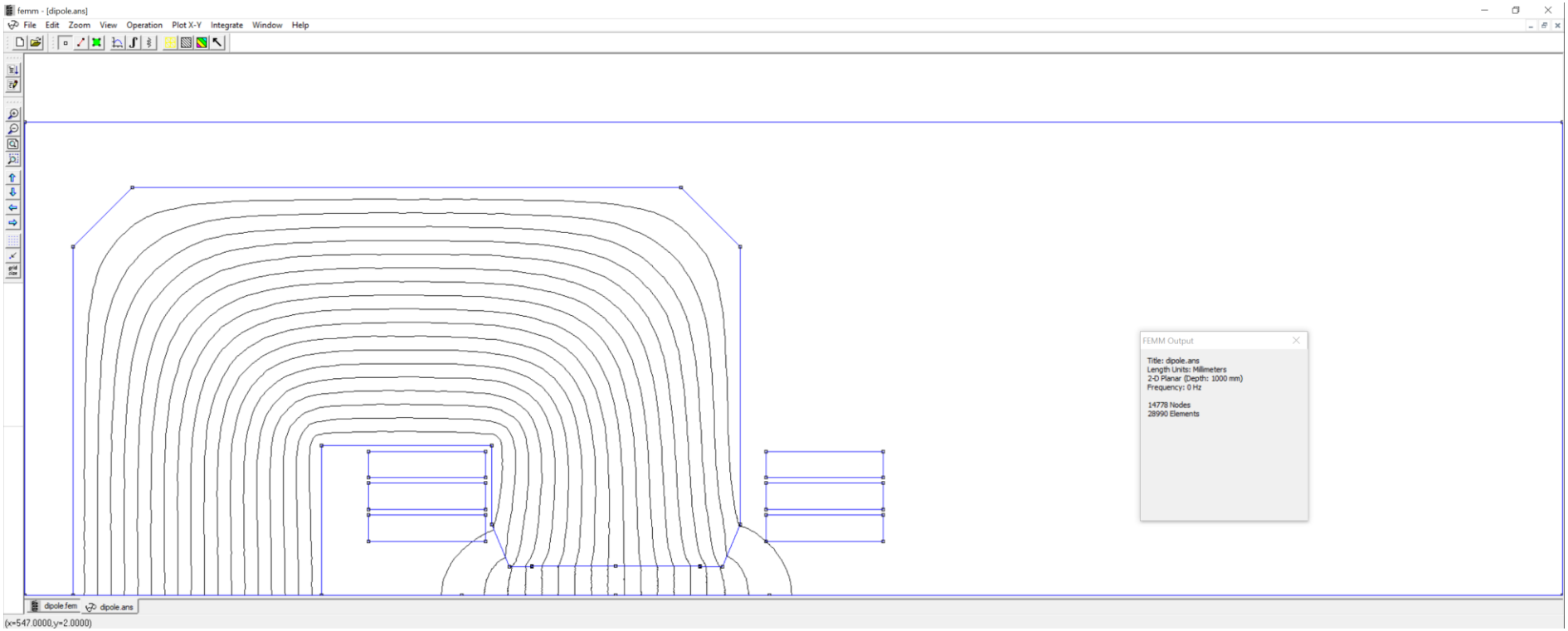
…

7. Post-processing: load solution

---> Analysis ---> View Results

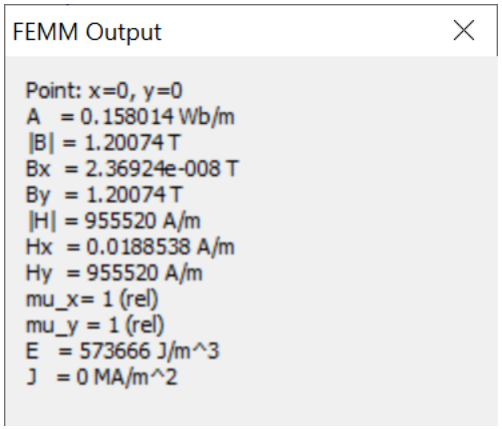
```
-- Post-processing
mi_loadsolution()
```

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

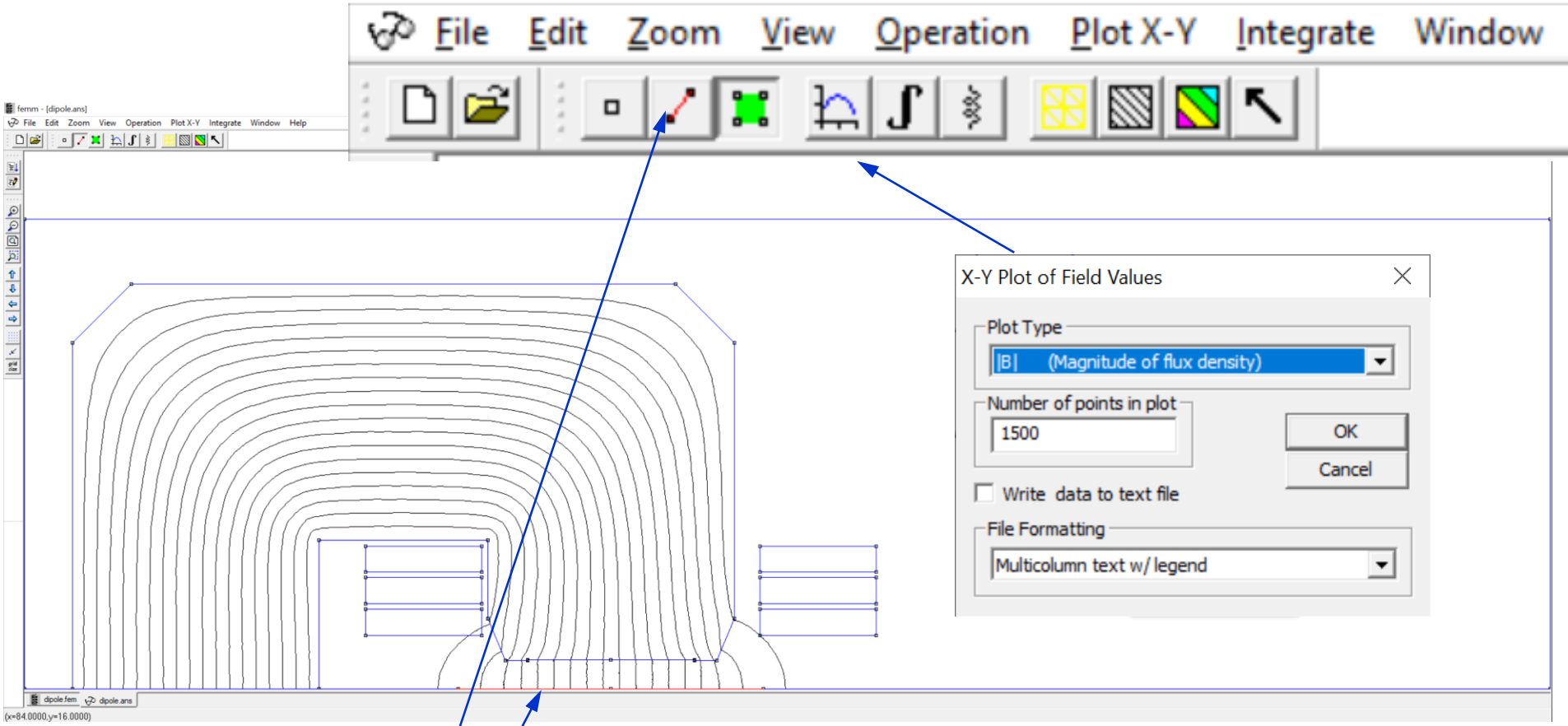


B ≈ 0.98*4e-7*pi*18*6*450/0.050 = 1.197 T

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



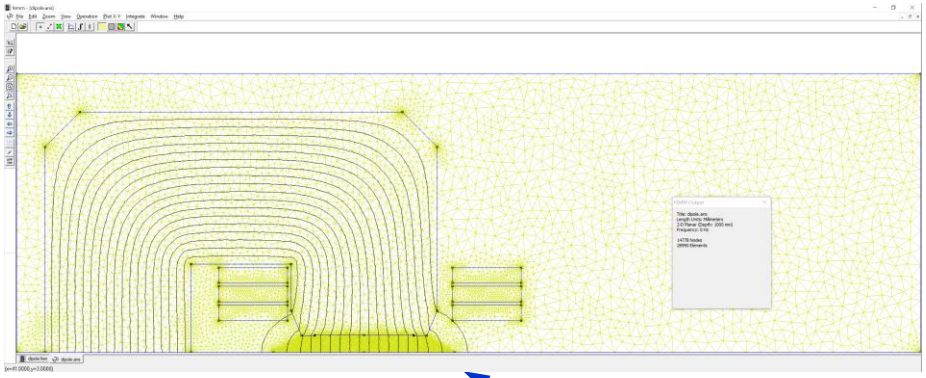
7. Post-processing: line plots

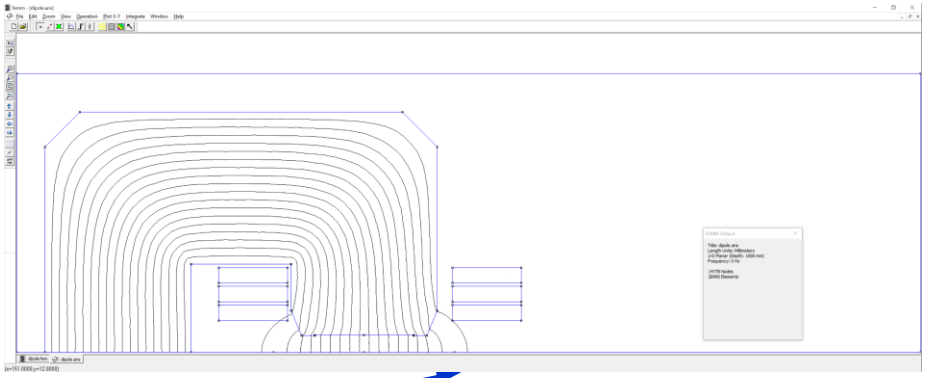


select a contour

- 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



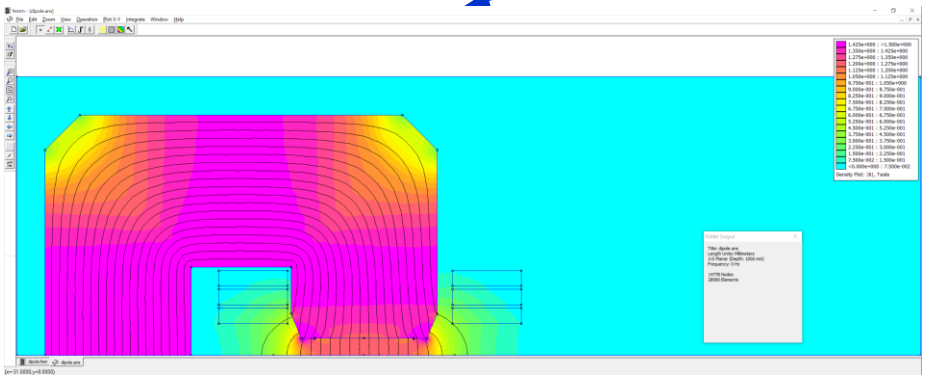


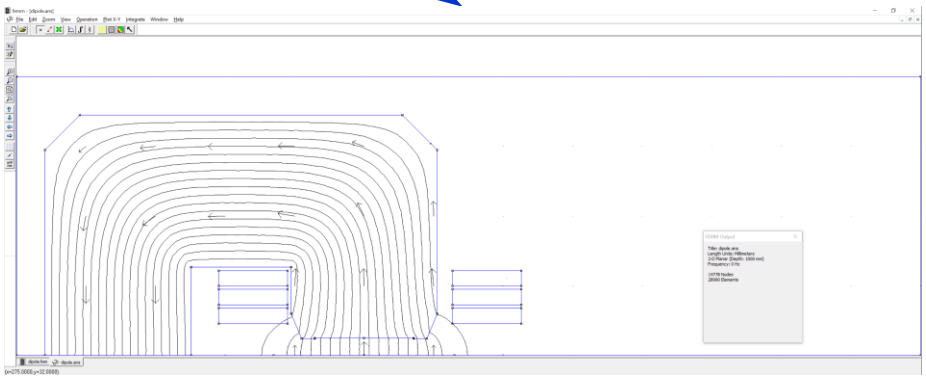
mesh

flux lines

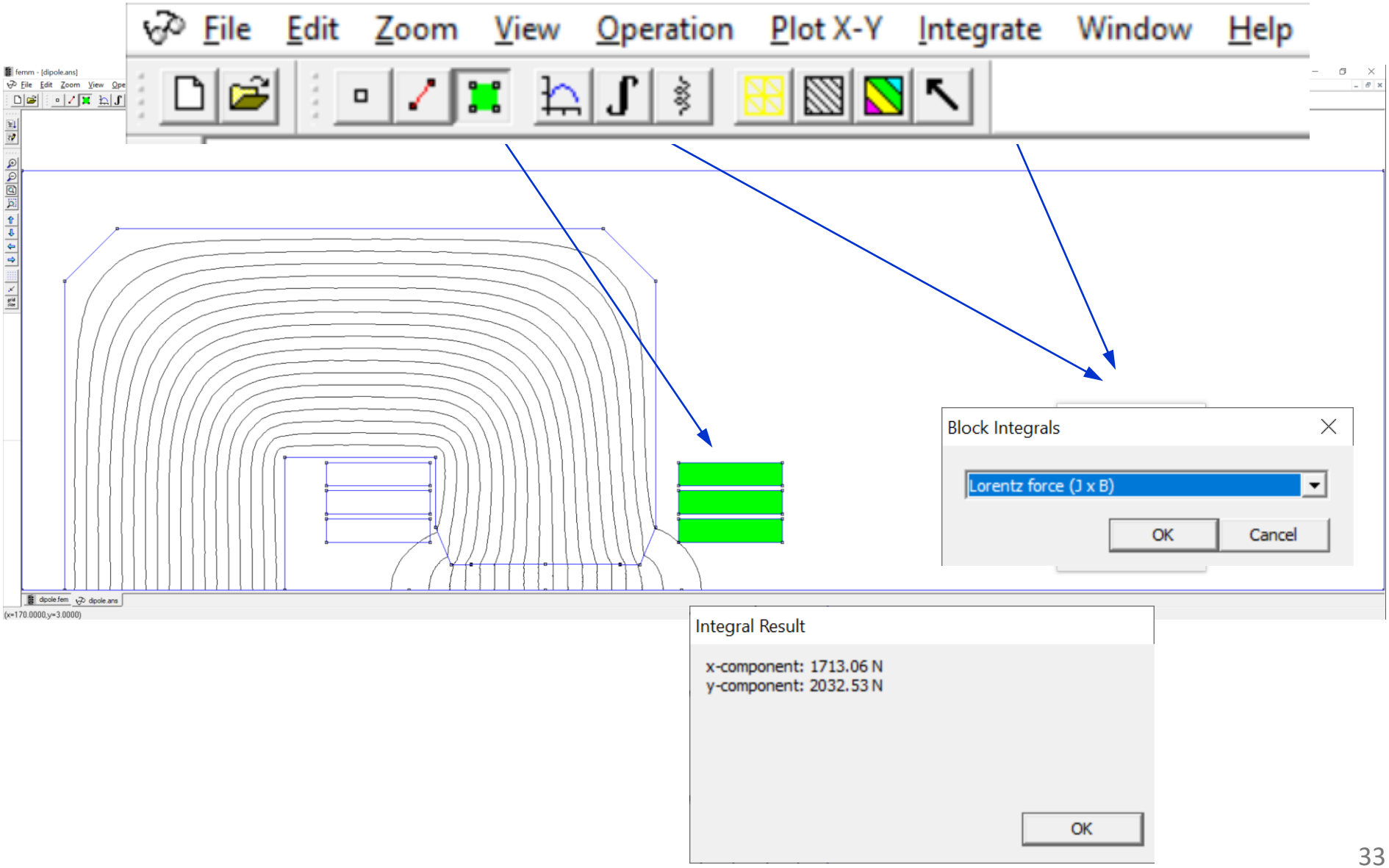
field density

field vectors

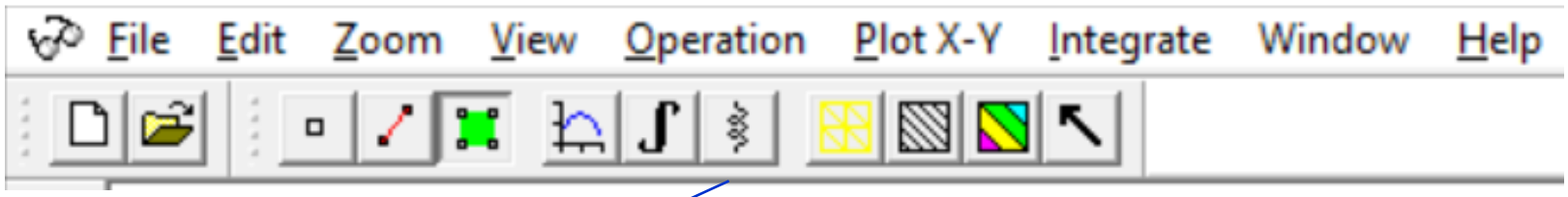


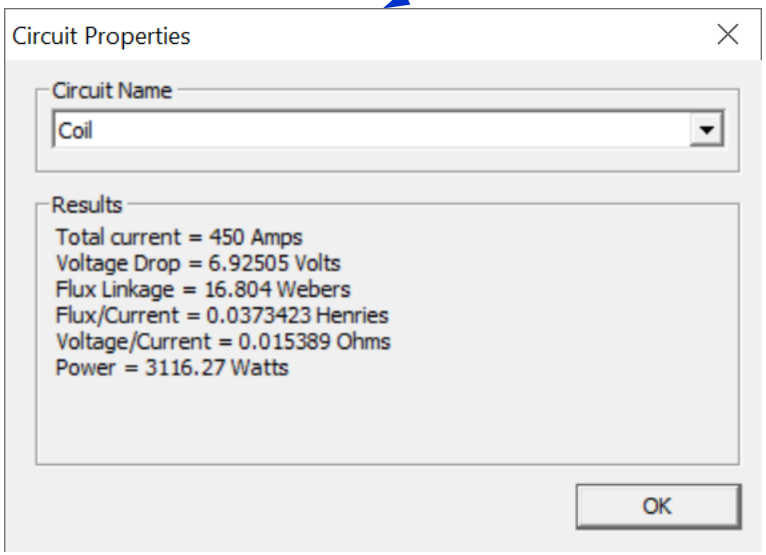


7. Post-processing: integrals over a block



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





L = 2*37.3 = 74.6 mH/m

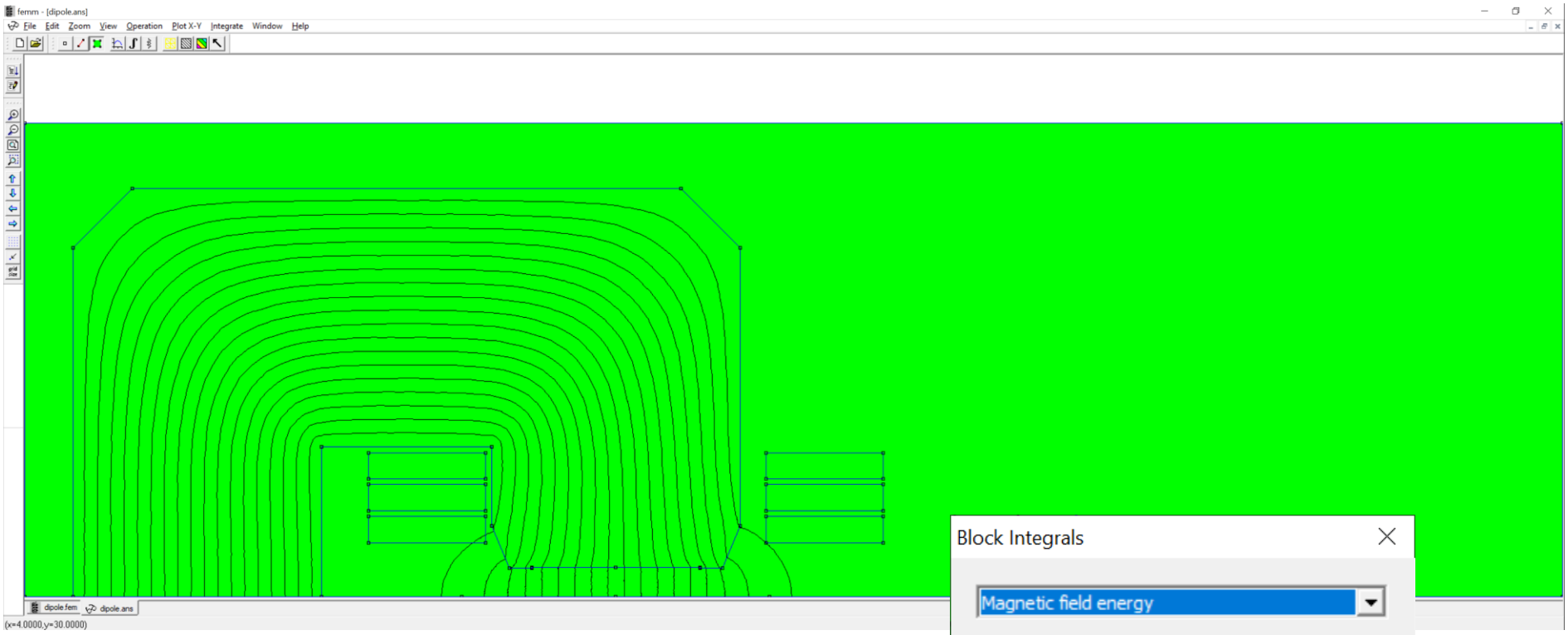
The factor 2 is because we model only half the dipole

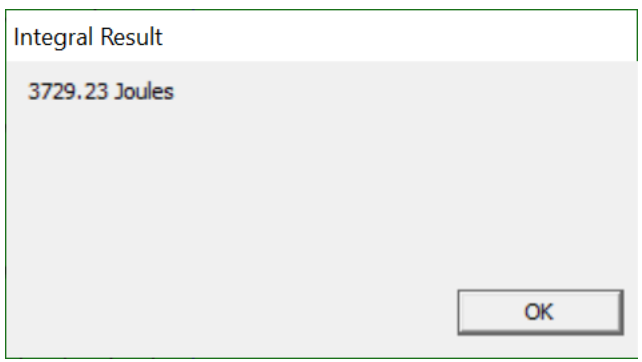
The result is per m, as we set 1 m depth; to get the total inductance, you can multiply by the magnetic length

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

In reality, the inductance is non-linear 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

(This is just the start, see the script multipoles_femm.lua on the Indico page)

```
--
-- 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
```