Electrical Circuit Element boundary conditions for parameter extraction

Implementation in `onelab`:
electromagnetic field, full-wave regime, linear devices
formulation in **E** (inside the domain), V (on the boundary)
ECE boundary conditions.

Gabriela Ciuprina, Daniel Ioan, Ruth V. Sabariego

This document describes some of the tests that have been done during the research for the paper [CIS22] related to electric circuit element (ECE) boundary conditions (BC) in the finite element method (FEM) for full-wave (FW) passive electromagnetic (EM) devices, which is an extended and improved version of [CIPL21]. In the [CIS22] paper, not only the weak formulation was stated in a new way, more elegant, but also the implementation is new, done in the onelab environment [DG].

The tests described can be found in the archive **ECEinOnelab_E_2021.zip**. *For details about the theory see [CIS22] and [CIPL21], and for details about the implementation in onelab please inspect the .pro and the .geo files. Some explanations are also given in this document.*

ECE BC ensures the perfect compatibility of EM devices with distributed parameters and external electric circuits with lumped parameters (LC), allowing the following types of EM-LC coupled simulations:

- *Numerical extraction of the reduced model of the distributed devices (as frequency characteristics, residuum/poles of the transfer/circuit functions, state equations matrices, or as a netlist of the equivalent circuit) and their simulation together with linear or nonlinear connected circuits using simulators as Spice, Simulink or 20sim;*
- *EM-LC co-simulation of electromagnetic field and electric circuits with lumped parameters;*
- *Coupled simulation as in the GetDP tutorial 8: Coupled FE and circuit equations.*

ECE is an alternative to the traditional approaches such as BC with modal fields for the propagating modes at the waveguide ports or by embedding of the lumped ideal circuit elements in the FEM mesh.

If this work is useful for you, please cite our work in [CIS22] and [CIPL21] as well as [DG]. If you use the AFS procedure, please cite [CILD12] and [GS99] as well.

Should you have questions or comments, do not hesitate to contact us.

Gabriela Ciuprina and **Daniel Ioan**
Politehnica University of Bucharest, Spl. Independentei 313, 060042, Bucharest, Romania
gabriela@lmn.pub.ro, daniel@lmn.pub.ro
Ruth V. Sabariego
KU Leuven, Kasteelpark Arenberg 10, box 2445, 3001 Leuven-Heverlee, Belgium
ruth.sabariego@kuleuven.be

Verdsion from March 13, 2022

"Strong essences are kept in small bottles"

Contents

0 Description of the files in the archive

0.1 Structure of the archive

The zip file contains one folder called **ECEinOnelab_E_2021**. In this folder you can find the following subfolders:

- **problemIndependent_proFiles**

  This folder includes the formulations described in [CIS22] and [CIPL21]. We preferred to group the formulations in a single folder, and include them from here, rather than making copies of the files in each problem folder. It is a template like way of working. The files you find here are `getdp` files (.pro files) that are problem independent. You should not modify the files here, expect for the case when you know what you are doing.

- **MatlabSources**,

  This folder includes some useful post-processing tools that you can call from Matlab. Examples of using these tools are given below. You should not modify the files here, expect for the case when you know what you are doing.

- **docs**

  This folder includes this document two almost final drafts of the papers [CIS22] and [CIPL21].

- **Results_log**

  This folder includes results that can be obtained with the provided examples, as well as some useful matlab scripts for post-processing them. You should use the information here as a reference, to check that you can obtain the same results with the provided example files.

- Folders for each problem described, the number that you can see in the name suggests the order in which you should investigate the files:
  01_Ishape2D*, **02_Thshape2D***, **03_Ishape3D*** and **04_LC***
  are first descriptions of the test problems, **05_*** is an example of how you can call gmsh and getdp from matlab, and **06_LC*** is a call from matlab including order extraction based on the adaptive frequency sampling (AFS) described in [CILD12] and Vector Fitting (VF) described in [GS99].

  If you want to play with other geometries, we recommend that you add new problem folders and call the formulations from the **problemIndependent_proFiles** folder.

0.2 Formulations available

The following formulations are available in the **problemIndependent_proFiles** folder:

1. Formulation for Full Wave (FW), ECE boundary conditions, in $\mathbf{E}$ inside the domain and V on the boundary.

   The core of this formulation, that includes only the function space and the equations can be found in the file
   only_FunctionSpace_and_Formulation_FullWave_E_ece.pro

It uses first order edge elements for **E**, first order nodal elements for V. The degrees of freedom are the electric voltages along inner edges and nodal potentials on the boundary;

This file is included by a general .pro file where the other objects are defined, which are designed separately for the single input single output (SISO) case and for the multiple input multiple output (MIMO) case - with 2 non-grounded terminals:

- **FullWave_E_ece_SISO_Vinside.pro** - here objects are described considering that the problem has two terminals, one is grounded, the other can be voltage or current excited. The frequency response will be saved in a Touchstone file with the extension ***.s1p**.
- **FullWave_E_ece_MIMO2terminals.pro** - here objects are described considering that the problem has three terminals, one is grounded, the other two can be voltage or current excited. Only one of the non-grounded terminals can be set as active, that is why the frequency response will be saved in a Touchstone file with the extension ***.s1p**. But, all the obtained files can be combined in a ***.s2p**. An example will be given in what follows.

2. Formulation for Full Wave (FW), ECE boundary conditions, in **E** inside the domain and V inside and on the boundary.

   The core of this formulation, that includes only the function space and the equations can be found in the file
   only_FunctionSpace_and_Formulation_FullWave_E_ece_Vinside

   This file is included by a general .pro file
   FullWave_E_ece_SISO_Vinside.

   This formulation was used only to check that the results obtained (frequency characteristics) are the same as when V is used only on the boundary.

   You should not use this formulation, unless you would like to do the same check.

3. Formulation for Full Wave (FW), classical boundary conditions, in **E** inside the domain and on the boundary.

   The core of this formulation, that includes only the function space and the equations can be found in the file
   only_FunctionSpace_and_Formulation_FullWave_E_classic.pro

   It uses first order edge elements for **E**. The degrees of freedom are the electric voltages along inner edges and edges on the boundary.

   This file is included by a general .pro file
   FullWave_E_classicBC.pro.

4. Formulation for Electrokinetics (EC), ECE boundary conditions, in V inside the domain and on the boundary.

   This is a steady state conduction, resistor type element, with 2 terminals: one is grounded and the other can be either voltage excited or current excited. The surface that does not belong to terminals has zero Neumann boundary condition.

The core of this formulation, that includes only the function space and the equations can be found in the file
only_FunctionSpace_and_Formulation_Electrokinetics_V_ece.pro

This file is included by a general .pro file
Electrokinetics_V_ece.pro.

This formulation was used for testing purposes only (correct implementation of global quantities associated to parts of the boundary).

Some brief explanations can be found in Appendix D.

0.3 Matlab tools

The archive provides some matlab functions we have previously developed in other projects, useful to do simple things such as compare frequency responses, change between frequency file formats, or more interesting things such as sampling the frequency range in an adaptive way, embedded with a model reduction based on vector fitting. The source files are in the folder **MatlabSources**.

Examples of how you can use them are given below.

0.4 Short description of the problems

Implementing ECE in onelab was for some of us a learning experience as well. In fact, the ECE formulation means only several lines of code, that describe the function spaces and the equations. The rest is only the ability to work in gmsh and getdp.

Here there is a short description of the files you can find in the archive. From the point of view of a new user of onelab, each test brings something new.

1. **Ishape2D**

   - **01_Ishape2D_1freq**
     2D problem (rectangle), boundary representation, SISO, solve for one frequency, use of onelab GUI to look qualitatively at the fields (see color maps, vector field representations).
   - **01_Ishape2D_ece_s1p**
     The same problem as above, but solving for several imposed frequencies and writing the frequency response in a Touchstone file (.s1p) (https://en.wikipedia.org/wiki/Touchstone_file).
   - **01_Ishape2D_ece_s1p_adaptedMesh**
     The same problem as above, but the mesh is conceived so that it takes into consideration the skin depth at high frequencies.

2. **Tshape2D**

   - **02_Tshape2D_ece_1freq**
     2D problem, boundary representation, MIMO, two terminals not grounded, various excitation possible (current, voltage, hybrid direct or reverse), solve for one frequency, use of onelab GUI to look qualitatively at the fields (see color maps, field lines).

- **02_Tshape2D_ece_s2p**

  Same as above, but solving for several frequencies, needed to generate a Touchstone file (.s2p).

3. **Ishape3D**

   - **03_Ishape3D_ece_Brep_1freq**

     3D problem (cylinder), boundary representation, SISO, solve for one frequency, use of onelab GUI to look qualitatively at the fields (see color maps, field representations).

   - **03_Ishape3D_ece_s1p_OCC_adaptedMesh**

     The same problem as above, use open cascade and a more appropriate mesh. Solve for several frequencies and write a s1p file.

   - **03axi_Ishape2.5D_ece_s1p_adaptedMesh**

     A 2D axisymmetric model for the Ishape3D problem, with a mesh adapted so that it considers the skin depth.

4. **LC**

   - **04_LC_GeometryInStepFile_ece_s1p**

     A more complicated geometry, described in a step file. Solve for a list of frequencies imposed by the user.

   - **04_LC_ParametricGeometryInGeoFile_ece_s1p**

     The LC test described in a parametric way, with another air box than the previous case.

5. **Calling gmsh and getdp from Matlab**

   This is a stand-alone folder, it contains a simple example (Ishape2D) – classical way of working in the onelab GUI and call for Matlab. It the folder **05_SimpleExampleCallingOnelabFromMatlab** there are two subfolders

   - **01b_Ishape2D_ece_s1p** - classical way of working.
   - **01b_Ishape2D_ece_s1p_callFromMatlab** - call for Matlab in order to solve the same problem.

6. **Model reduction and parameter extraction**

   - **06_LC_GeometryInStepFile_ece_s1p_callFromMatlab_AFS**

     The LC problem with the geometry in STEP format, build a reduced order model with the vector fitting procedure while computing the frequency characteristics with adaptive frequency sampling. It runs from Matlab, gmsh and getdp are called with system calls.

1 Ishape2D

1.1 Model description and analytic solution

This is the test described in the [CIPL21] paper. The computational domain and the placement of terminals is shown in Fig. 1. The following numerical data was used: $a = 2.5$ μm; $l = 10$ μm; $h = 10$ μm (for postprocessing and for current excited case); $\mu = 4\pi \cdot 10^{-7}$ [H/m]; $\varepsilon = 8.854187812812 \cdot 10^{-12}$ F/m; $\sigma = 6.6 \cdot 10^{7}$ [S/m].

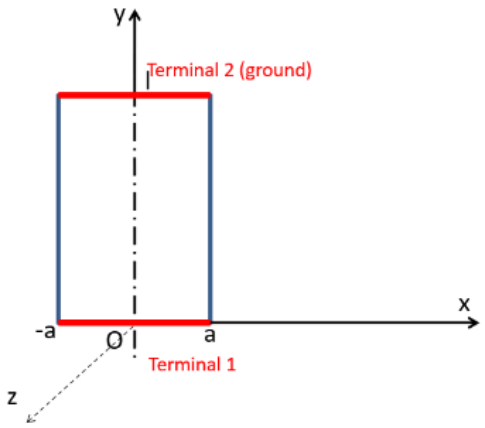


Figure 1: Ishape2D test: Computational domain and placement of terminals.

This was a useful problem because in this case, the FW-ECE BC formulation is equivalent (from the field point of view) to the FW-classical BC formulation, as follows:

	Classical BC	Equivalent ECE BC
Voltage excited type	$y = 0, x \in [-a, a]$ $E_x = 0$ $y = l, x \in [-a, a]$, $E_x = 0$ $x = -a, y \in [0, l]$, $E_y = \text{voltage}/l$ $x = a, y \in [0, l]$, $E_y = \text{voltage}/l$	$y = 0, x \in [-a, a]$, $V_{T1} = \text{voltage}$ $y = l, x \in [-a, a]$, $V_{T2} = 0$ $x = -a, y \in [0, l]$, "natural ECE" $x = a, y \in [0, l]$, "natural ECE"
Current excited type	$y = 0, x \in [-a, a]$ $E_x = 0$ $y = l, x \in [-a, a]$, $E_x = 0$ $x = -a, y \in [0, l]$, $\mathbf{n} \times \mathbf{H}_z = \text{current}/(2h)\mathbf{j}$ $x = a, y \in [0, l]$, $\mathbf{n} \times \mathbf{H}_z = -\text{current}/(2h)\mathbf{j}$	$y = 0, x \in [-a, a]$, $I_{T1} = \text{current}$ $y = l, x \in [-a, a]$, $V_{T2} = 0$ $x = -a, y \in [0, l]$, "natural ECE" $x = a, y \in [0, l]$, "natural ECE"

An analytic solution can be computed easily (see Appendix A). You can a evaluate and visualize the analytic solution with the code **main_2Drectangle.m** that can be found in the folder **Results_log/Ishape2Danalitic**. You only need to change the sourcespath and problempath, according to your settings (lines 8 and 9 at the beginning of the file).

```
1 % File main_2Drectangle.m
2 % Analytic solution of the vector Helmholtz equation in a rectangle
3 % Gabriela Ciuprina, February 4, 2020
4 clc;
5 close all hidden
6 % prepare path to use chamy tools
7 restoredefaultpath;
8 sourcespath = genpath('D:\Gabriela\OneLab\mytests\ECEforOnelab\ECEinOnelab_E_2021\MatlabSources\Chamy')
    ;
9 problempath = 'D:\Gabriela\OneLab\mytests\ECEforOnelab\ECEinOnelab_E_2021\Results_log\01
    _Ishape2Danalitic';
10 addpath(sourcespath);
```

```
addpath(problempath);
chdir(problempath);
```

The frequency characteristic is written in a s1p (lines 60 and 62 below).

```
% File rectangle_analitic_FW.m
function rectangle_analitic_FW(geom,mat,freq,files,idxfig)

a = geom.a;
l = geom.l;
h = geom.h;
sig = mat.sig;
miu = mat.mu;
fvect = freq.fvect;
outpath = files.outpath;
problempath = files.problempath;
chdir(outpath);
epsi = mat.eps;

%%
omega = 2*pi*fvect;
gamma_cplx_patrat = 1i*omega*miu.*(sig + 1i*omega*epsi);
gamma_cplx = sqrt(gamma_cplx_patrat);
% radicalul din nr complexe - cu grija!

% I = 1 (valoarea efectiva a curentului)

gamma_a = gamma_cplx*a;

P_ap_cplx_lineic = (1/(2*h))*(gamma_cplx./(sig + 1j*omega*epsi)).*(cosh(gamma_a)./sinh(gamma_a));
R_h = real(P_ap_cplx_lineic);
X_h = imag(P_ap_cplx_lineic);
L_h = X_h./omega;

idxfig = idxfig+1;
figure(idxfig); clf;
loglog(fvect,R_h,'--m','Linewidth',2);
xlabel('f [Hz]');
ylabel('R [\Omega]');
title('Rezistance - from FW, analytic');
grid on;
%ylim([4e-3,1e-1]);
%xlim([1e-1,100e9]);
print(strcat('fig',num2str(idxfig),'.jpg'),'-djpeg');
print(strcat('fig',num2str(idxfig),'.eps'),'-depsc');

idxfig = idxfig+1;
figure(idxfig); clf;
loglog(fvect,L_h,'--k','Linewidth',2);

xlabel('f [Hz]');
ylabel('L [H]');
title('Inductance -  from FW, analytic');
grid on;
%ylim([1e-13,1e-12]);
print(strcat('fig',num2str(idxfig),'.jpg'),'-djpeg');
print(strcat('fig',num2str(idxfig),'.eps'),'-depsc');

Zfw = R_h + 1i.*2*pi.*fvect.*L_h;
Yfw = 1./Zfw;


filename = files.rootname;
snpZ = strcat(filename,'Z.s1p');
writesnp_v2(snpZ, fvect, Zfw, 'Z', 'Hz', 50, 'RI');
snpY = strcat(filename,'Y.s1p');
writesnp_v2(snpY, fvect, Yfw, 'Y', 'Hz', 50, 'RI');
% snp2snp('',snp_filename,'Z','RI',snpZ);
snp_imag_over_omega('',snpZ,'' );
chdir(problempath);


end
```

1.2 Test 01_Ishape2D_1freq: use of various formulations, solve one frequency

In this example, you can "play" with all classical and ECE BC and see field maps. The frequency, the geometrical and material parameters can be changed from the GUI. The boundary conditions are by default computed so that they correspond either to a voltage excitation with 1 V or to a current excitation with 1 A. The following figures (see descriptions in captions) show typical maps you can see. Just launch onelab, load the **Ishape2d.pro** file and run.

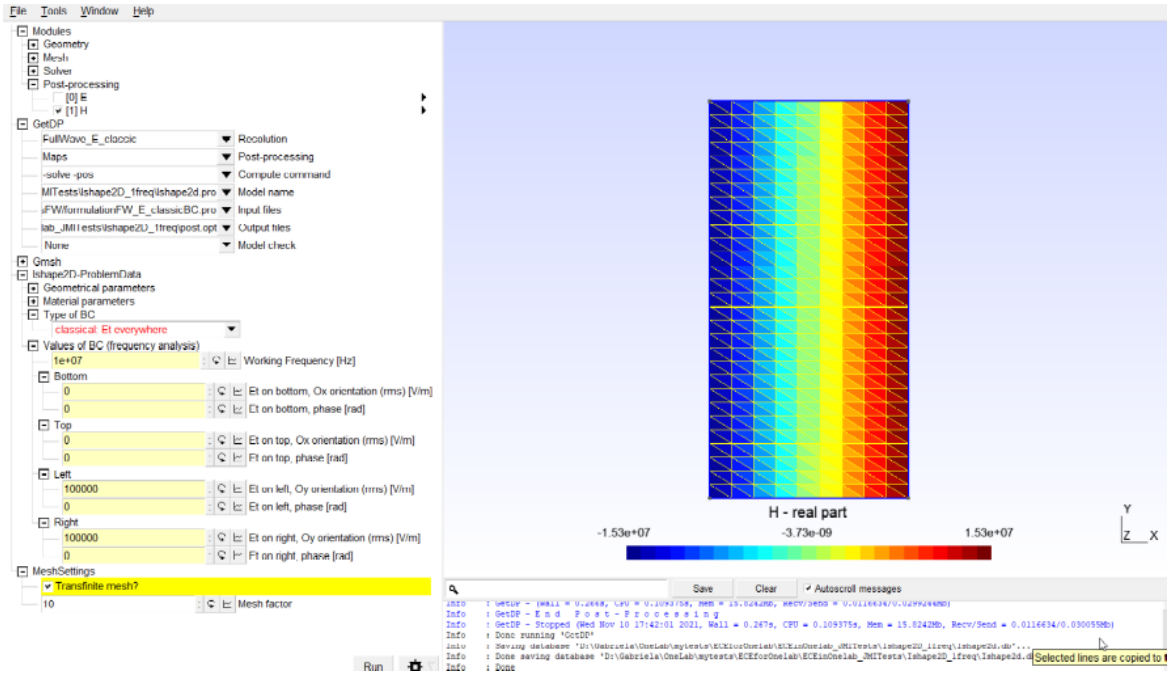


Figure 2: **01_Ishape2D_1freq**: Use of FW with classical BC. Here $\mathbf{E}_t$ was imposed everywhere on the boundary. The map shows the real part of H, at 1e7 Hz.

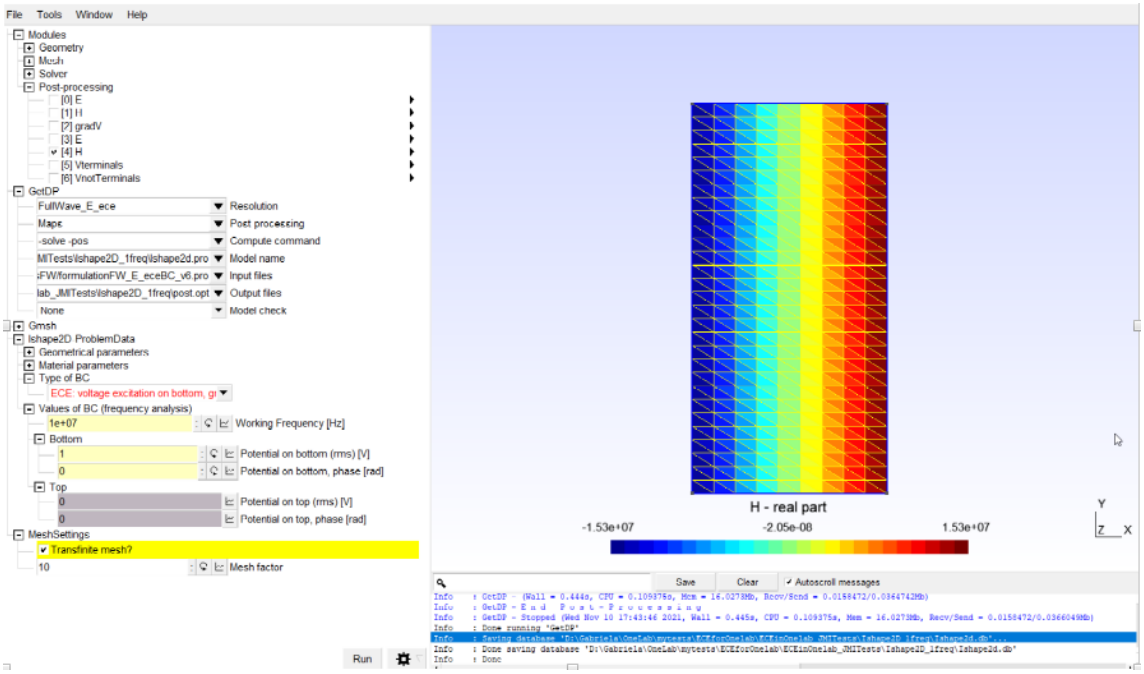


Figure 3: **01_Ishape2D_1freq**: Use of FW with ECE BC, voltage excitation. The field is identical to the field in Fig. 2. This was a first validation of the correct implementation of the ECE.

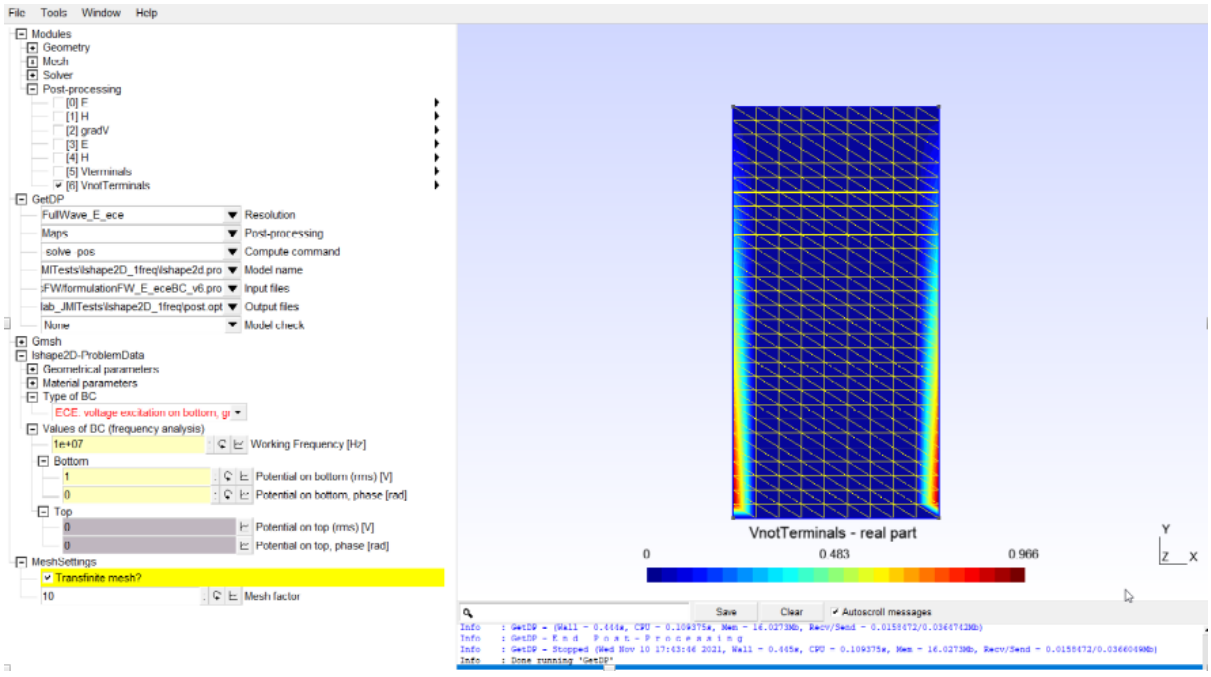


Figure 4: **01_Ishape2D_1freq**: Same test as in Fig. 3, V on the boundary is displayed. The unknowns are strictly on the boundary in fact. Here an interpolation is carried out near the boundary. The figure shows only the potential on the boundary which does not include the terminals.

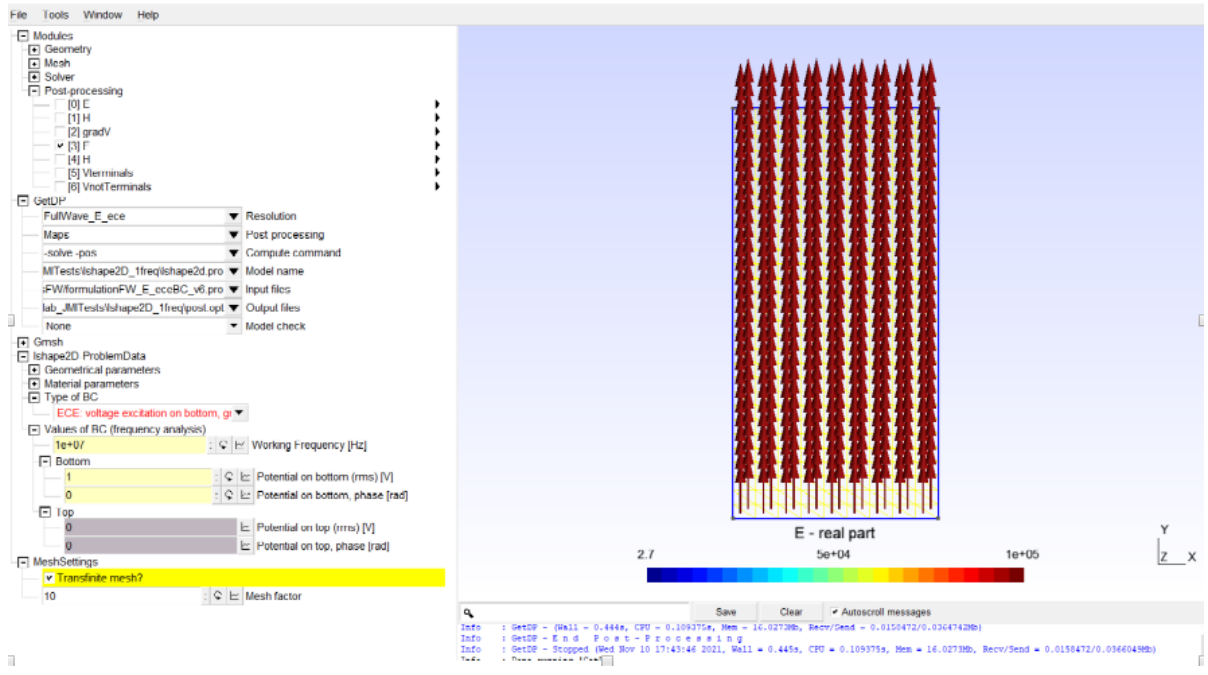


Figure 5: **01_Ishape2D_1freq**: Same test as in Fig. 3, the **E** field is displayed (real part).

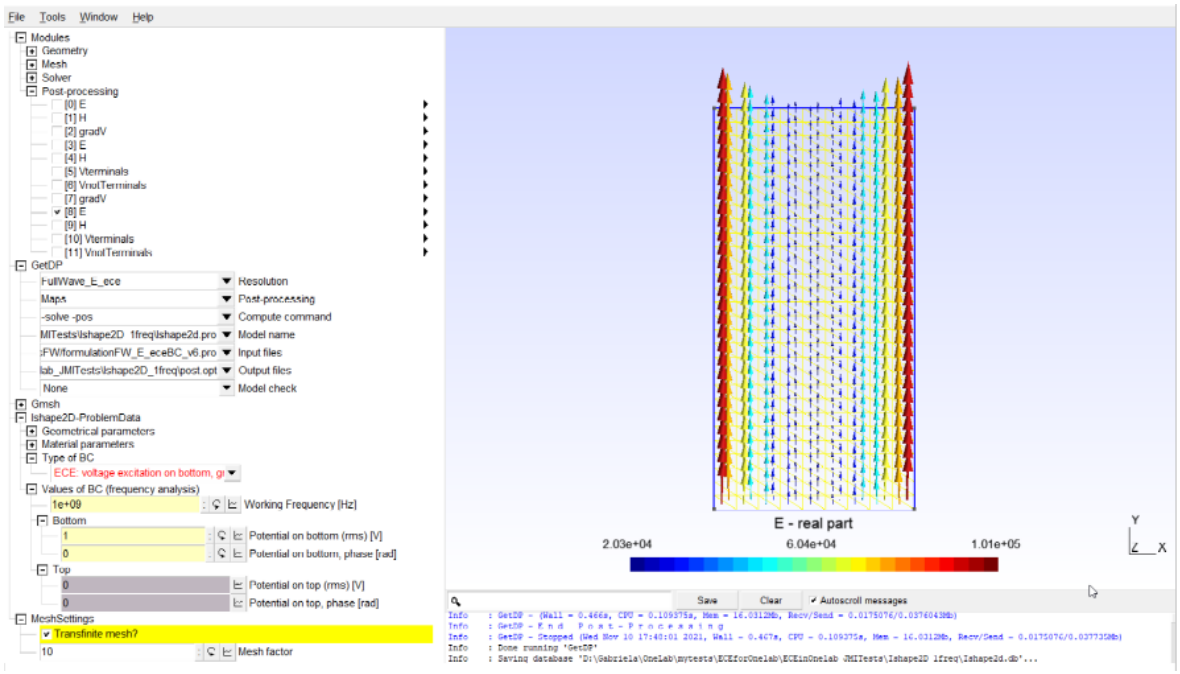


Figure 6: **01_Ishape2D_1freq**: Same test as in Fig. 3, the **E** field is displayed (real part), the frequency is changed to 1 GHz. You can see a strong skin depth effect.

1.3 Test 01_Ishape2D_ece_s1p: ECE formulation, solve several frequencies, write s1p

In this example, only ECE BC were set. From the interface you can change the type of excitation (voltage or current), as well as the frequency values to be computed. In the **Ishape2D_data.pro** file values are initially set for the minimum (fmin) and maximum (fmax) frequencies as well as the number of frequency points (nop) that will be computed. The frequencies are linearly distributed in the frequency range, as shown in the following piece of code.

```
fmin = 1e7;    // Hz
fmax = 100e9; // Hz

nop = 20;
//freqs()= LogSpace[Log10[fmin],Log10[fmax],nop];
freqs()= LinSpace[fmin,fmax,nop];
DefineConstant[
  Freq  = {freqs(0), Choices{freqs()}, Loop, Name StrCat[mValuesBC, "0Working Frequency"],
                  Units "Hz", Highlight  Str[colorMValuesBC], Closed !close_menu }
];
```

Just launch onelab, load the **Ishape2d.pro** file and run.

In order to save the transfer function, you have to chose as Postprocessing the “TransferMatrix” option (Fig. 7).

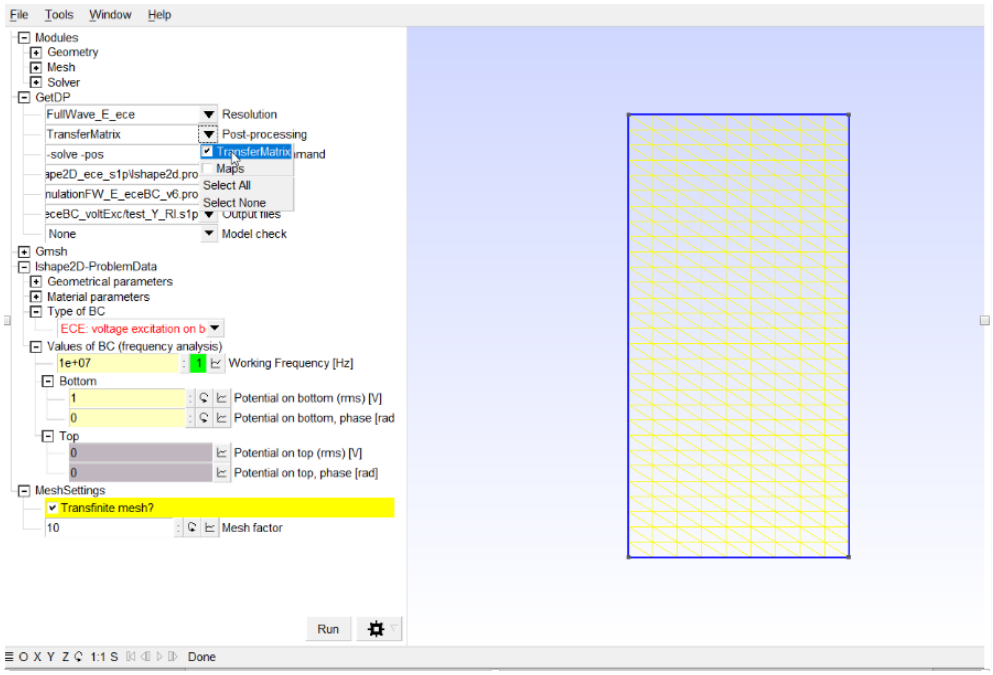


Figure 7: **01_Ishape2D_ece_s1p**: In order to save the s1p file, you have to chose the "TransferMatrix" postprocessing option.

In the case of a voltage excitation, a file called **test_Y_RI.s1p** will be saved in the **res/FWeceBC_voltExc** folder. In the case of a current excitation, a file called **test_Z_RI.s1p** will be saved in the **res/FWeceBC_crtExc** folder. If these files already exists (e.g. from previous simulations), the new results will be appended to the old content of the file.

For example, here it is the content of the **test_Y_RI.s1p** file obtained for the mesh in Fig. 7 and the frequencies set as explained before.

```
# Hz Y RI R 50
10000000 329.9536433200924 -3.552568095219233
5272631578.947369 58.69178665402725 -53.5911706154402
10535263157.89474 43.96697262601042 -36.36720113258519
15797894736.84211 37.68298712699642 -28.55055279773523
21060526315.78947 34.22534693330498 -23.86618485322638
26323157894.73684 32.04127404414925 -20.70721540109225
31585789473.68421 30.54033939516463 -18.42748454051295
36848421052.63158 29.44460426052127 -16.70992144544824
42111052631.57895 28.60497574060061 -15.37627071292636
47373684210.52631 27.93469507011244 -14.31661594613089
52636315789.47369 27.38040869498801 -13.45869828223423
57898947368.42105 26.90796835573502 -12.75277512050838
63161578947.36842 26.49485036939355 -12.16341022001878
68424210526.31579 26.12584763395962 -11.66469038712076
73686842105.26315 25.79049974658071 -11.23727674697945
78949473684.21053 25.48149971513116 -10.86650262662929
84212105263.1579 25.19367498371496 -10.54110027623597
89474736842.10527 24.92331911237773 -10.25232204102804
94737368421.05263 24.66774434919038 -9.993318096548821
100000000000 24.42497705330639 -9.758686293920348
```

You can compare the obtained frequency response with a reference one (e.g. the analytic one in this case). Let's assume that we placed (moved) the important results in a specific folder, e.g. **Results_log**. For example, in **Results_log/Ishape2Danalitic_vs_onelab** you can find various results, and in each sub-folder you can see a **main_compare.m** file, which is a the matlab function allowing you to compare snp files (snpdiff). Let's look in this file:

```
1 % File main_compare.m
2 clc;
```

```
3  clear all;
4  close all hidden
5  % prepare path to use chamy without gui
6  restoredefaultpath;
7  sourcespath = genpath('D:\Gabriela\OneLab\mytests\ECEforOnelab\ECEinOnelab_E_2021\MatlabSources');
8  problempath = 'D:\Gabriela\OneLab\mytests\ECEforOnelab\ECEinOnelab_E_2021\Results_log\01
       _Ishape2Danalitic_vs_onelab\results9mar22';
9  addpath(sourcespath);
10 addpath(problempath);
11 chdir(problempath);
12
13 if 1 == 1
14     % comparison of Z files (from crt exc) and computation of L
15     snp_ref_pathname = '../../01_Ishape2Danalitic/out_analitic_FW_100pct_forReference/';
16     snp_ref_filename = '2Drectangle_xy_FW_Z.s1p';
17     snp_an_pathname{1} = './res/FWeceBC_crtExc/';
18     snp_an_filename{1} = 'test_Z_RI.s1p';
19     snpdiff(snp_ref_filename,snp_ref_pathname,snp_an_filename,snp_an_pathname);
20     chdir(snp_an_pathname{1});
21     snp_imag_over_omega('',snp_an_filename{1},'');
22     chdir(problempath);
23 end
24
25
26 if 1 == 0
27     % comparison of Y files (from voltage exc)
28     snp_ref_pathname = '../../01_Ishape2Danalitic/out_analitic_FW_100pct_forReference/';
29     snp_ref_filename = '2Drectangle_xy_FW_Y.s1p';
30     snp_an_pathname{1} = './res/FWeceBC_voltExc/';
31     snp_an_filename{1} = 'test_Y_RI.s1p';
32     snpdiff(snp_ref_filename,snp_ref_pathname,snp_an_filename,snp_an_pathname);
33 end
34
35 if 1 == 0
36     % comparison of RL files
37     snp_ref_pathname = '../../01_Ishape2Danalitic/out_analitic_FW_100pct_forReference/';
38     snp_ref_filename = '2Drectangle_xy_FW_Z_imag_over_omega.s1p';  % this has to be computed, see the
           first if above
39     snp_an_pathname{1} = './res/FWeceBC_crtExc/';
40     snp_an_filename{1} = 'test_Z_RI_imag_over_omega.s1p';
41     snpdiff(snp_ref_filename,snp_ref_pathname,snp_an_filename,snp_an_pathname);
42
43 end
44
45 if 1 == 0
46     % conversions to other formats
47     snp_an_pathname{1} = './res/FWeceBC_crtExc/';
48     snp_an_filename{1} = 'test_Z_RI.s1p';
49     chdir(snp_an_pathname{1});
50     snp2snp('',snp_an_filename{1},'S','DB',strcat(snp_an_filename{1},'_S_DB.s1p'));
51     snp2snp('',snp_an_filename{1},'S','MA',strcat(snp_an_filename{1},'_S_MA.s1p'));
52     snp2snp('',snp_an_filename{1},'Z','MA',strcat(snp_an_filename{1},'_Z_MA.s1p'));
53     chdir(problempath);
54
55
56 end
```

Lines 7 and 8 - set the paths to the codes, you have to change them. Lines 13, 26, 35, 45 are just flag type lines. For instance, as it is now, you can compare s1p files of Z type (impedance). In this case another file is created, which is a s1p in which the imaginary part was divided by the angular frequency ω (line 21), so it will contain the resistance and the inductance. If you want to compare s1p files of Y type (admittance), then set to true the condition at line 26. If you want to compare resistances and inductances, then set to true the condition at line 35. If you want to do conversions to other types (S) or representations (MA), then set to true the condition at line 45.

Alternatively (and maybe easier), if you have set the path to the matlab sources, you can type at the Matlab console

```
snpdiff_tool
```

which will open a short dialog, allowing you to select snp files. The first one you select is the reference one, with respect to which a global error will be computed. This is how figures such as the one in Fig. 8 can be obtained, which can be found in the folder **Results_log/01_Ishape2Danalitic_vs_onelab/results9mar22**

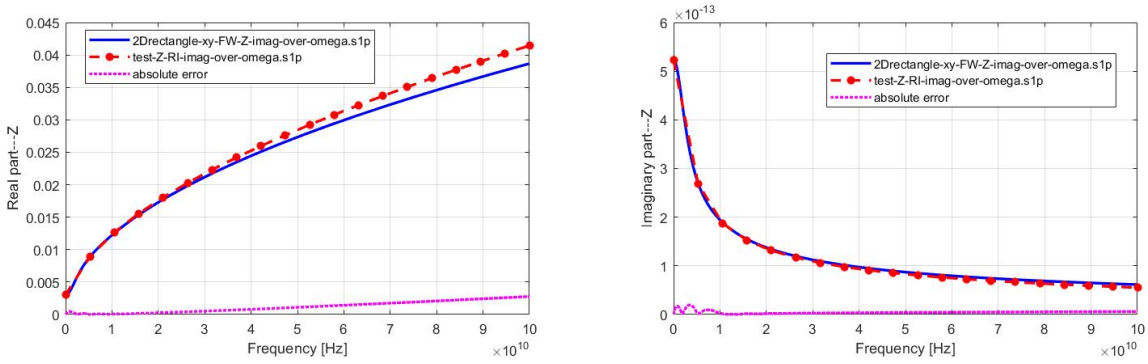


Figure 8: **01_Ishape2D_ece_s1p**: Such figures are produced with the snpdiff function. Of course, for a nicer representation you have to change the labels which are, by default, the names of the files that have been chosen. In this case the first file chosen is the reference - the blue curve, and the second file chosen is the numerical computation - the red curve.

1.4 Test 01_Ishape2D_ece_s1p_adaptedMesh: ECE formulation, solve several frequencies, write s1p, with a mesh that considers the eddy current effect

This test is similar to **01_Ishape2D_ece_s1p**, but the mesh is built according to the skin depth.

In the GUI you can set the number of elements per skin depth (Figs. 9 and Figs. 10), and this affects the size of the mesh near the left and right boundary. The setting is done in the **Ishape2D.geo** file (folder **01_Ishape2D_ece_s1p_adaptedMesh**), which is also shown below (see lines 14,15, 44-61).

```
1  /*  Ishape2d.geo
2  Geometrical description (for gmsh) of Ishape2D test for ECE
3  For details, see comments in the Ishape2d_data.pro file
4  Meshing information is also defined here.
5  The mesh size depends on the skin depth.
6  */
7
8
9  Include "Ishape2d_data.pro";
10
11 /* Definition of parameters for local mesh dimensions */
12 //p0 = s*l/10; // characteristic length of mesh element
13 delta = Sqrt(2.0/(2*Pi*Freq*mu*sigma));
14 If (delta < a)
15         p0 = delta/nbDelta;
16 Else
17         p0 = l/10/2;
18 EndIf
19
20 /* Definition of gemetrical points */
21 Point(1) = { -a, 0, 0, p0} ;
22 Point(2) = {  a, 0, 0, p0} ;
23 Point(3) = {  a, l, 0, p0} ;
24 Point(4) = { -a, l, 0, p0} ;
25
26 /* Definition of gemetrical lines */
27 Line(1) = {1,2};
28 Line(2) = {2,3};
29 Line(3) = {3,4};
30 Line(4) = {4,1};
31
32 /* Definition of geometrical surfaces */
33 Line Loop(5) = {1, 2, 3, 4};
34 Plane Surface(6) = {5};
35
36 /* this is not used now
37 If(_use_transfinite)
38   Transfinite Line {2,4} = 3*s;
39   Transfinite Line {1,3} = 1*s;
40   Transfinite Surface {6};
41 EndIf
42 */
```

```
43
44 Field[1] = Distance;
45 Field[1].CurvesList = {2,4};
46 Field[1].Sampling = 50;
47 Field[2] = Threshold;
48 Field[2].InField = 1;
49 Field[2].SizeMin = p0;
50 Field[2].SizeMax = a/5;
51 Field[2].DistMin = 0;
52 Field[2].DistMax = 3*delta;
53
54
55 Field[3] = Min;
56 Field[3].FieldsList = {2};
57 Background Field = 3;
58 Mesh.MeshSizeExtendFromBoundary = 0;
59 Mesh.MeshSizeFromPoints = 0;
60 Mesh.MeshSizeFromCurvature = 0;
61 Mesh.Algorithm = 5;
62
63 /* Definition of Physical entities (surfaces, lines). The Physical
64    entities tell GMSH the elements and their associated region numbers
65    to save in the file 'Ishape2d.msh'. */
66
67 Physical Surface ("MaterialX", 100) = {6} ;       /* MaterialX */
68
69 Physical Line ("Ground",    120) = {3} ;          /* Ground */
70 Physical Line ("Terminal", 121) = {1} ;          /* Terminal */
71 Physical Line ("RightBoundary", 131) = {2} ;     /* RightBoundary */
72 Physical Line ("LeftBoundary", 132) = {4} ;      /* LeftBoundary */
```

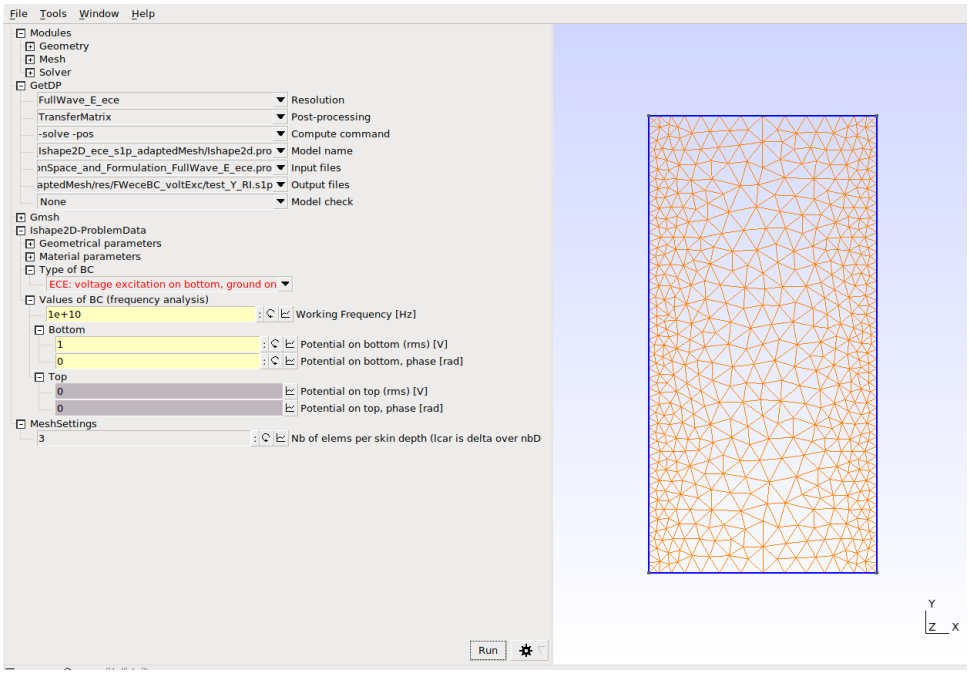


Figure 9: **01_Ishape2D_ece_s1p_adaptedMesh**: Mesh generated for $f = 10^10$ Hz.

The extracted R and L are done as explained in the previous test, the results can be found in the folder **Results_log/01_Ishape2Danalitic_vs_onelab/results9mar22_adaptedMesh**, and they can be seen in Fig. 11.

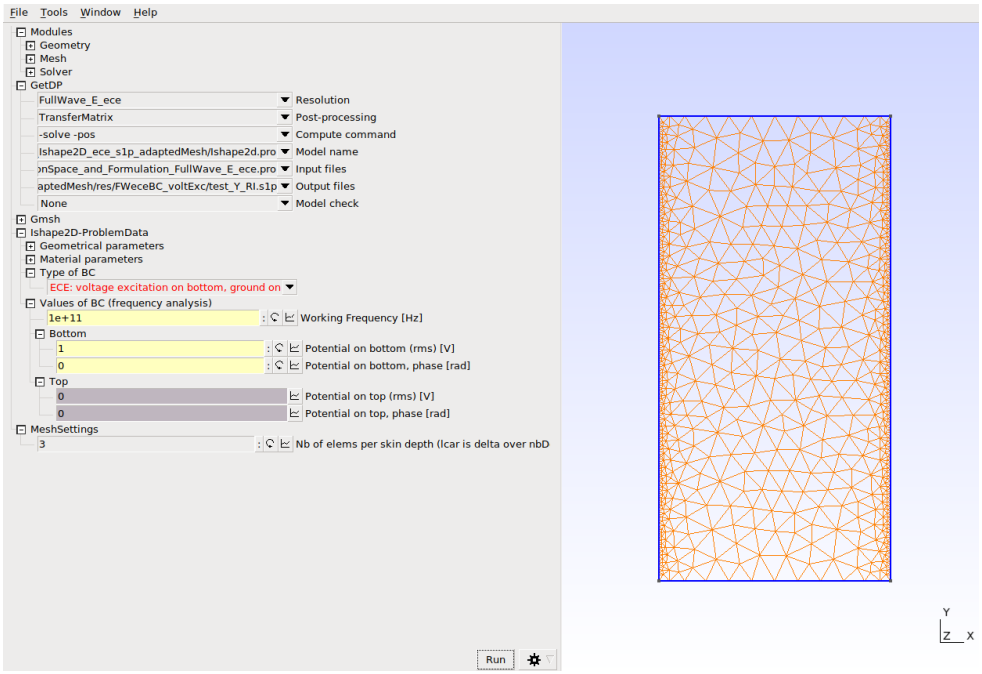


Figure 10: **01_Ishape2D_ece_s1p_adaptedMesh**: Mesh generated for $f = 10^11$ Hz.

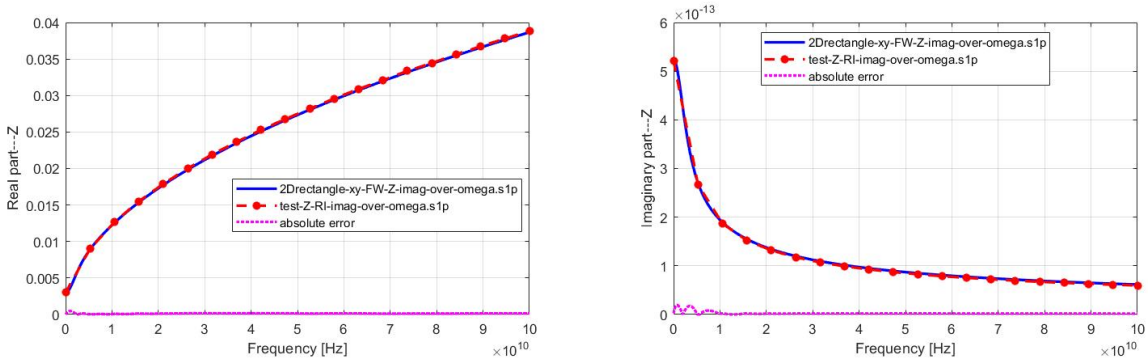


Figure 11: **01_Ishape2D_ece_s1p_adaptedMesh**: Results are better if you adapt the mesh according to the field.

2 Tshape2D

This is the test similar to the one described in the [CIPL21] paper, to check that the hybrid excitation is implemented correctly. The T part is conductive, there are three terminals, the bottom terminal is always the ground.

2.1 Test 02_Tshape2D_ece_1freq: ECE formulation, solve one frequency

This test was done to check qualitatively the obtained results (Figs. 12 and 13).

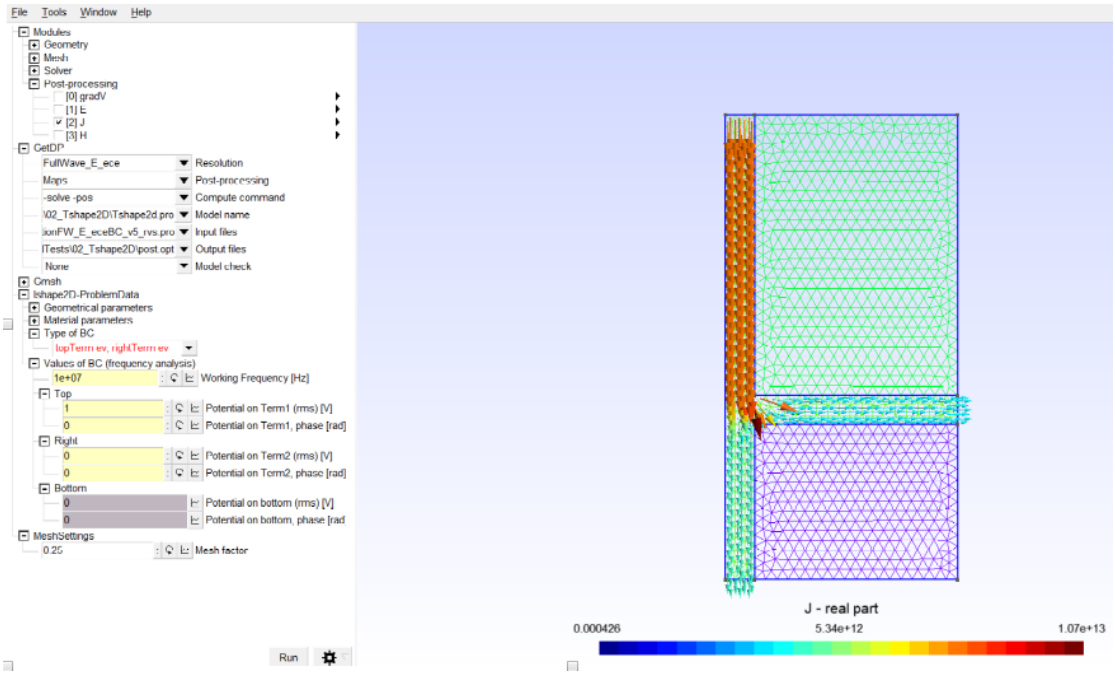


Figure 12: **02_Tshape2D_ece_1freq**: Top voltage excited (ev) with 1 V, bottom ground (gnd), right ev with V = 0 (but the value can be nonzero).

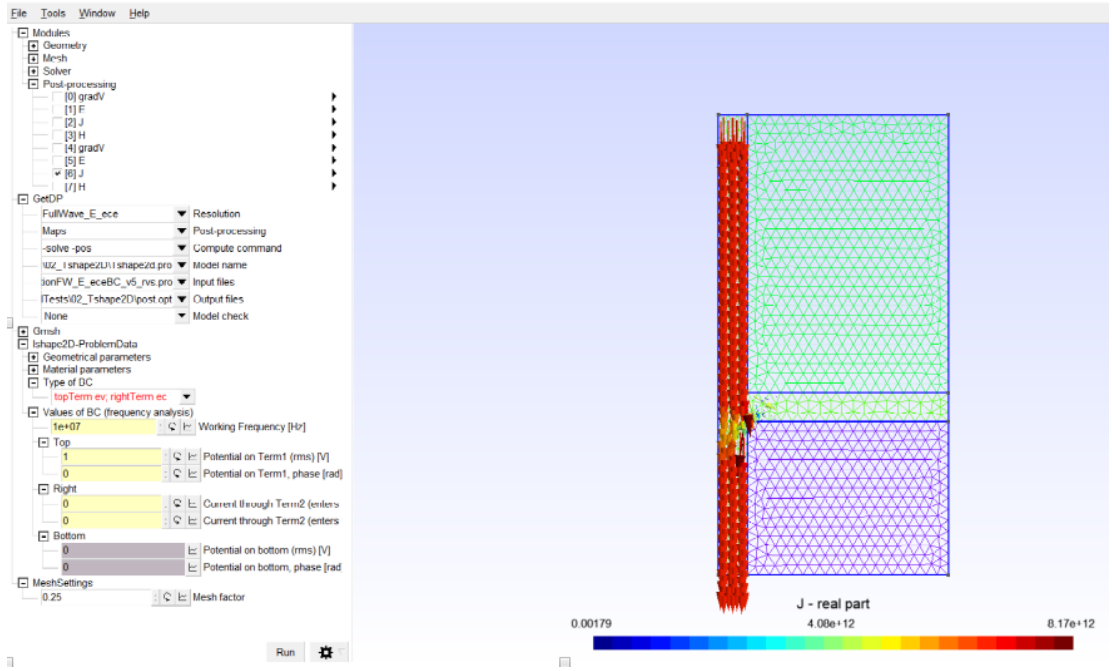


Figure 13: **02_Tshape2D_ece_1freq**: Top ev 1 V, right current excited (ec) 0 A (open)

2.2 Test 02_Tshape2D_ece_s2p: ECE formulation, solve several frequencies, write s2p file

In this test you can chose from the interface the type and excitation and the active terminal (Fig. 14).

In order to be able to generate the s2p file, you have to do two simulations, one having the active terminal the one numbered as 1 and the other simulation with the active terminal number 2. Each simulation writes two s1p file. After the 2 simulations you have four s1p files that can be combined in a s2p file with a Matlab script you can find in the **Results_log/02_Tshape2D_MIMO_assembleS2P** folder. Details are given in the **readme.txt** file you can find there.

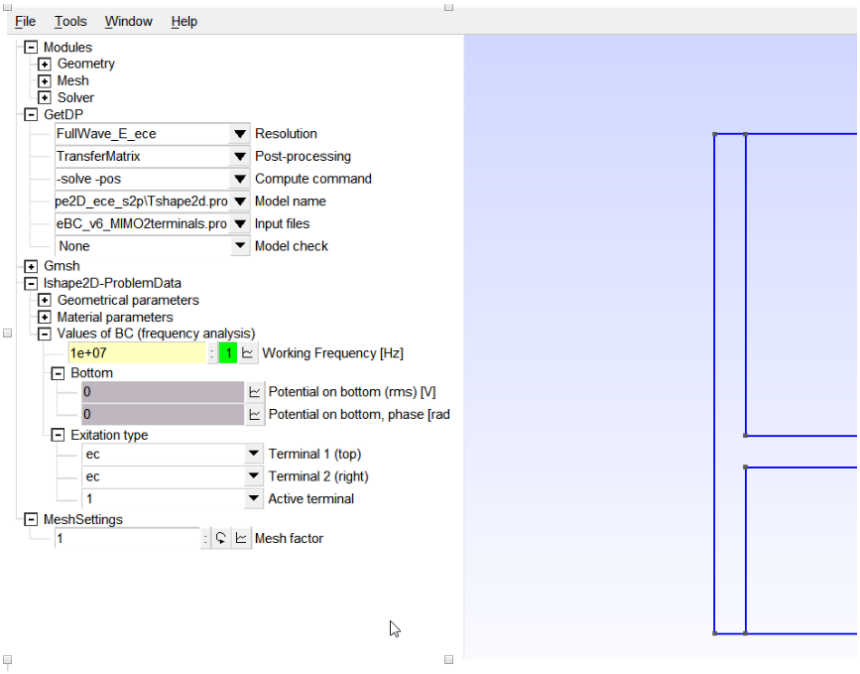


Figure 14: **02_Tshape2D_ece_1freq**: Top voltage excited (ev) with 1 V, bottom ground (gnd), right ev with V = 0 (but the value can be nonzero).

Depending on the excitation type, the generated s2p file will represent an impedance matrix Z, an admittance matrix Y or a hybrid matrix (H or G).

Examples for all these 4 cases can be found in the **Results_log/02_Tshape2D_MIMO_assembleS2P/results9mar22** folder.

3 Ishape3D

3.1 Model description and analytic solution

This is the test described in the [CIS22] paper.

This test is a cylindrical domain with radius a and length l, having linear and homogeneous material properties. Its ends are two terminals, one grounded and the other excited either in current or voltage. This configuration has the advantage that a formulation with classical boundary conditions is equivalent to a formulation with ECE boundary conditions. The classical boundary conditions formulation admits an analytic solution in terms of Bessel functions for the current excitation case (see Appendix B). This is used to validate the numerical solution of FEM, in 3D-FW regime with ECE boundary conditions.

The analytic solution is computed with the code **main_Ishape.m** that can be found in the folder **Results_log/Ishape3Danalitic**.

3.2 Test 03_Ishape3D_ece_Brep_1freq: ECE formulation, solve one frequency, Brep for the geometry

In this test the cylinder was defined by using boundary representation, using 5 + 5 points, 2 x 4 quarters of circles, and 4 lines, then 6 surfaces (2 disks + 4 curved surfaces) + one volume (Fig. 15).

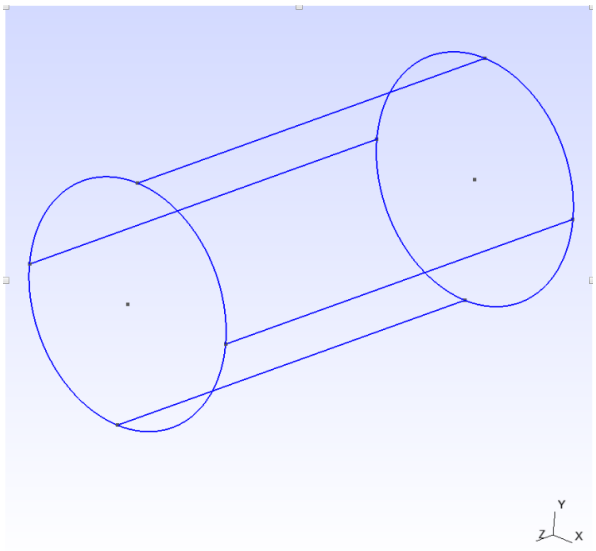


Figure 15: **03_Ishape3D_ece_Brep_1freq**: Description of the Ishape3D using the build-in gmsh Kernel.

In this way, the physical regions associated to the boundary can be set easily and use the same ECE formulation defined in onelab for the Ishape2D problem.

Pay attention to set a problem depth to 1 (so that it is harmless when postprocessing). This is set at the end of the file **Ishape3d_data**.

Figures 16 - 19 there are some results (radius of the cylinder $a = 2.5$ μm, and length $l = 10$ μm).

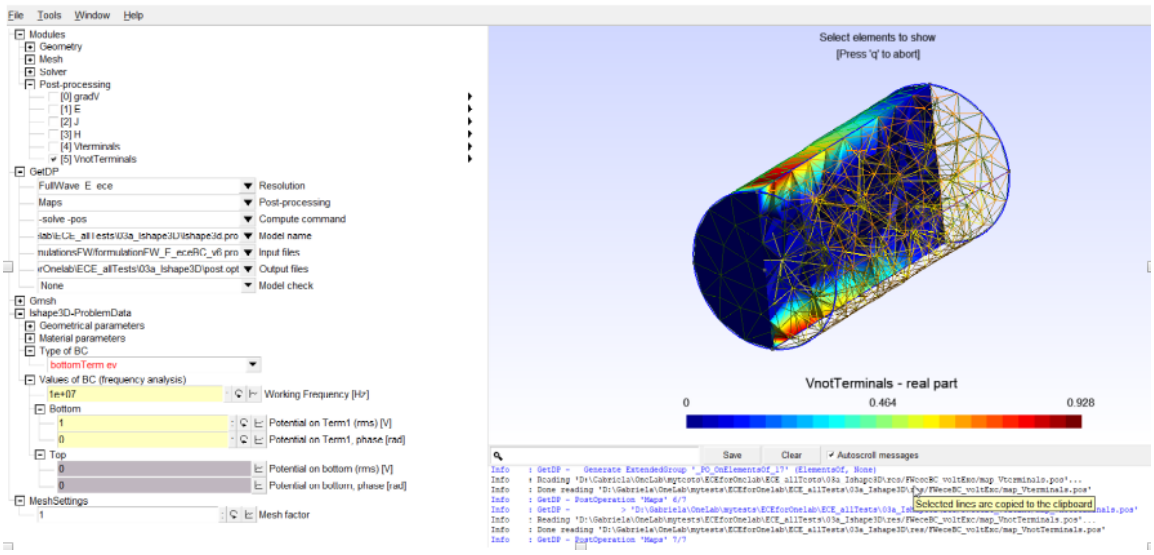


Figure 16: **03_Ishape3D_ece_Brep_1freq**: f = 1e7 Hz, voltage excitation with 1V, Potential on the boundary which does not include terminals.

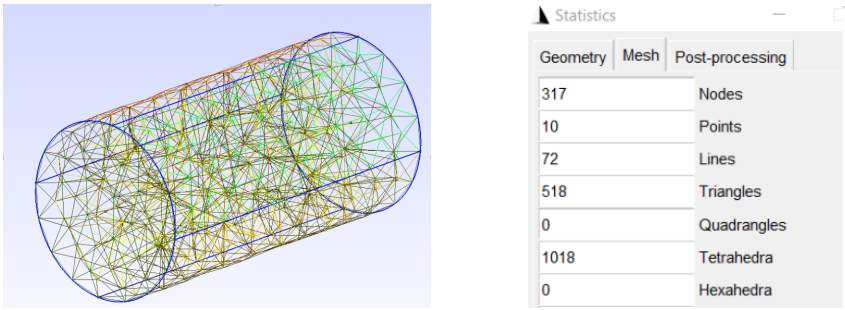


Figure 17: **03_Ishape3D_ece_Brep_1freq**: Mesh (1003 degress of freedom), and its statistics.

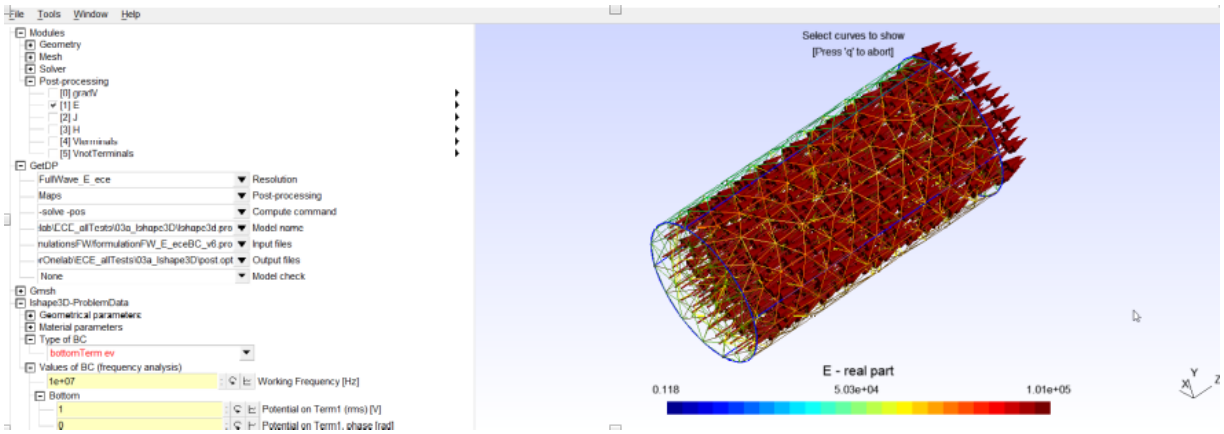


Figure 18: **03_Ishape3D_ece_Brep_1freq**: **E** – real part.

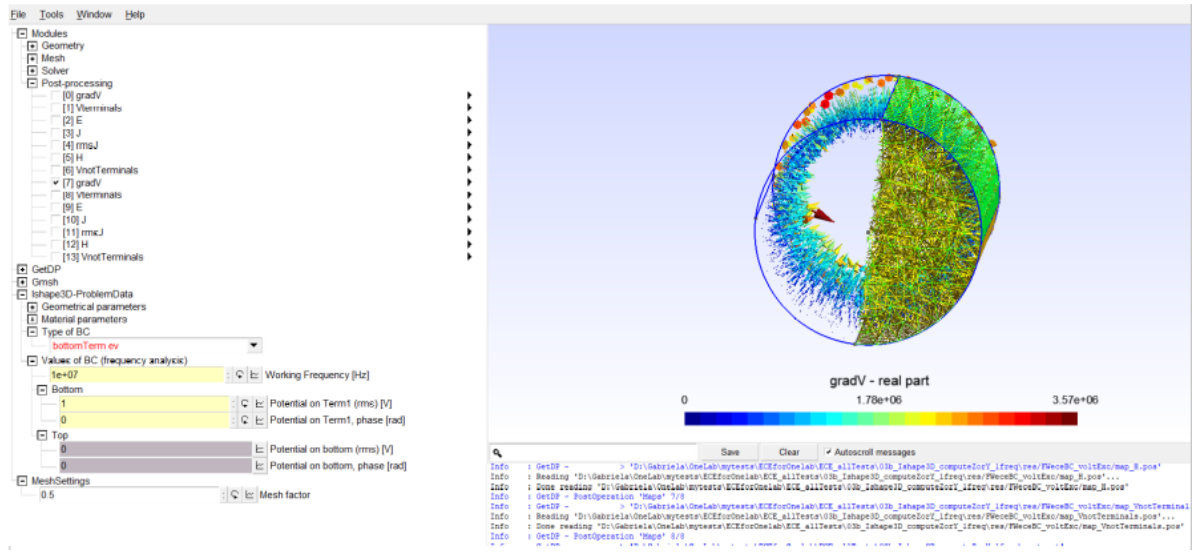


Figure 19: **03_Ishape3D_ece_Brep_1freq**: Gradient of V – only in a layer near the boundary.

3.3 Test 03_Ishape3D_ece_s1p_OCC_adaptedMesh: ECE formulation, solve several frequencies, CSG for the geometry, adapted mesh

Inspired by gmsh tutorial no. 10, here it is another description of the Ishape3D, using constructive solid geometry (Open Cascade Kernel). You can see that the discretization takes into account the skin depth, and the mesh is coarser in the middle of the cylinder.

```
/*  Ishape3d.geo
Geometrical description (for gmsh) of Ishape3D test for ECE
For details, see comments in the Ishape3d_data.pro file
Meshing information is also defined here.

This uses OpenCascade
*/


Include "Ishape3d_data.pro";
SetFactory("OpenCASCADE");

xc = 0; yc = 0; zc = 0;
vx = 0; vy = 0; vz = l;
radius = a;

volDom = newv; Cylinder(newv) = {xc, yc, zc, vx, vy, vz, radius};

b() = Boundary{ Volume{volDom}; };
Printf("    surfaces", b());

// In order to identify which surface is which I played with the GUI!!!

lateralCyl = b(0);
bottomSurf = b(1);
topSurf = b(2);

lc() = Boundary{ Surface{lateralCyl}; };
Printf("    curves", lc());
circle1 = lc(0);
lineCyl = lc(1);
circle2 = lc(2);

// Physical regions
// ------------------------
Physical Volume ("MaterialX", 100) = {volDom} ;       /* MaterialX */

Physical Surface ("Ground",   120) = {bottomSurf} ;                /* Ground */
Physical Surface ("Terminal", 121) = {topSurf} ;                   /* Terminal */
Physical Surface ("BoundaryNotTerminal", 131) = {lateralCyl} ;

/* Definition of parameters for local mesh dimensions */
//lcar = s*l/10; // characteristic length of mesh element
```

```
44
45 // nbDelta from _data.geo,  number of elements per skin depth
46
47 delta = Sqrt(2.0/(2*Pi*Freq*mu*sigma));
48 lcarDelta = delta/nbDelta;
49 lcar = s*l/10;
50
51 Field[1] = Cylinder;
52 Field[1].VIn = lcar;
53 Field[1].VOut = lcar;
54 Field[1].Radius = a;
55 Field[1].XCenter = 0;
56 Field[1].YCenter = 0;
57 Field[1].ZCenter = l/2;
58 Field[1].XAxis = 0;
59 Field[1].YAxis = 0;
60 Field[1].ZAxis = l;
61
62 Field[2] = Cylinder;
63 Field[2].VIn = a/3;
64 Field[2].VOut = lcarDelta;
65 Field[2].Radius = a - delta;
66 Field[2].XCenter = 0;
67 Field[2].YCenter = 0;
68 Field[2].ZCenter = l/2;
69 Field[2].XAxis = 0;
70 Field[2].YAxis = 0;
71 Field[2].ZAxis = l;
72
73 Field[3] = Min;
74 Field[3].FieldsList = {1,2};
75 Background Field = 3;
76 Mesh.MeshSizeExtendFromBoundary = 0;
77 Mesh.MeshSizeFromPoints = 0;
78 Mesh.MeshSizeFromCurvature = 0;
79 Mesh.Algorithm = 5;
```

Now, the mesh generated depends on the frequency, e.g. you can see below the DoFs and how the meshes look like.

Freq no	Freq [Hz]	Dofs-1elem per skin depth	Dofs-2elem per skin depth
1	1e7	1828	1828 (here no skin depth)
2	1.1e10	2981	19945
3	2.2e10	5586	40670
4	3.3e10	8145	60828
5	4.4e10	10223	81200
6	5.6e10	12321	103455
7	6.7e10	14326	124326
8	7.8e10	17852	146441
9	8.9e10	17890	168613
10	1e11	19333	190024

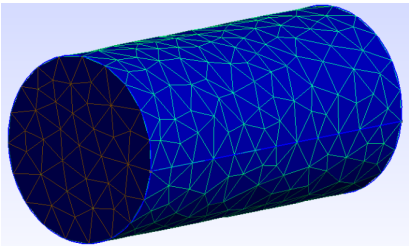


Figure 20: **03_Ishape3D_ece_s1p_OCC_adaptedMesh**: f = 1e7 Hz, no skin effect here.

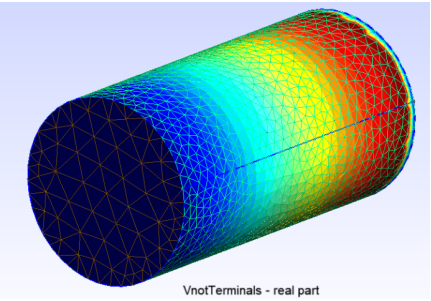


Figure 21: **03_Ishape3D_ece_s1p_OCC_adaptedMesh**: f = 3e10 Hz, some skin effect. 1 elem per skin depth, 7276 dofs.

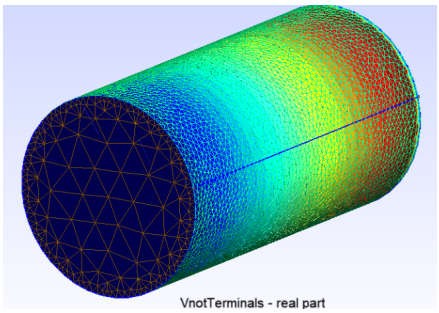


Figure 22: **03_Ishape3D_ece_s1p_OCC_adaptedMesh**: f = 3e10 Hz, 2 elems per skin depth, 54406 dofs.

The results of this test can be found in the file **Results_log/03_Ishape3Danalitic_vs_onelab/results9mar22**.

They were used to obtain the figures in the [CIS22] paper, which are also shown here in Fig. 23.

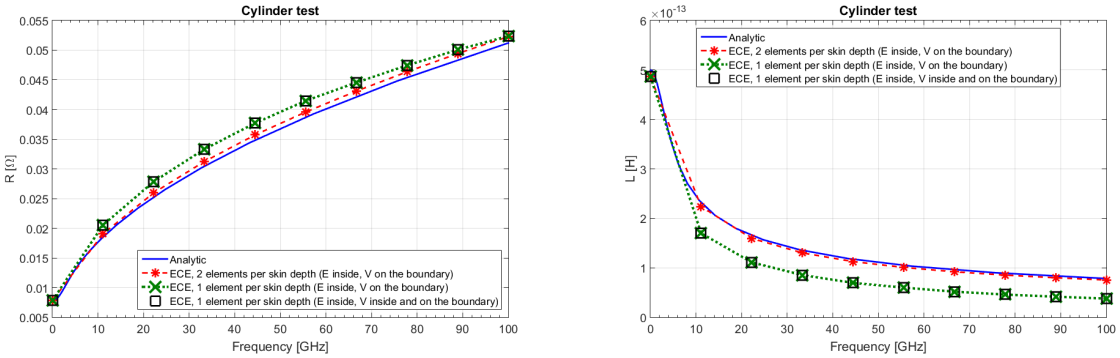


Figure 23: **03_Ishape3D_ece_s1p_OCC_adaptedMesh**: These results validate that the ECE implementation is also correct for 3D models.

Figure 23 also include the formulation with V inside. To obtain the results for this formulation, you have to modify one line in the **Ishape3D.pro** file in the folder **03_Ishape3D_ece_s1p_OCC_adaptedMesh**, as follows:

Instead of:

```
/* The formulation and its tools */
Include "../problemIndependent_proFiles/FullWave_E_ece_SISO.pro"
//Include "../problemIndependent_proFiles/FullWave_E_ece_SISO_Vinside.pro"
```

Change to

```
/* The formulation and its tools */
//Include "../problemIndependent_proFiles/FullWave_E_ece_SISO.pro"
Include "../problemIndependent_proFiles/FullWave_E_ece_SISO_Vinside.pro" -
```

3.4 Test 03axi_Ishape2.5D_ece_s1p_adaptedMesh: 2D AXI model for Ishape 3D

Ishape3D was used in order to verify the implementation for a 3D problem with analytic solution. However, Ishape3D can be more efficiently modeled with a 2D axisymmetric model.

This is given in the folder **03axi_Ishape2.5D_ece_s1p_adaptedMesh**.

What you have to do is just set the correct flags at the end of the file **Ishape2dAXI_data.pro**.

```
modelDim = 2; //
Flag_Axi = 1; // 1 for AXI - it makes sense only for modelDim = 2

If ((modelDim == 2)&&(Flag_Axi == 0))
    h2Ddepth = h;
ElseIf ((modelDim == 2)&&(Flag_Axi == 1))   // 2D AXI
    h2Ddepth = 2*Pi;
Else // 3D
    h2Ddepth = 1;
EndIf
```

The mesh is generated with boundary representation and fineness that depend on the skin depth. Figures 24 - 26 show some results.

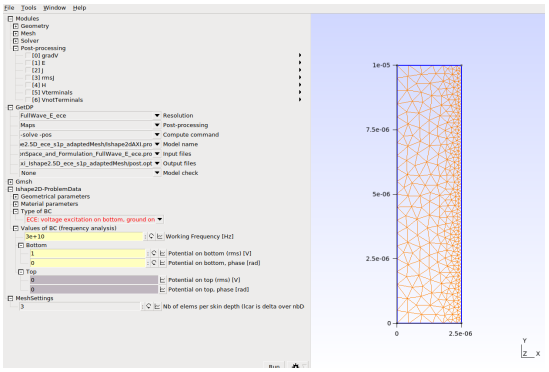


Figure 24: **03axi_Ishape2.5D_ece_s1p_adaptedMesh**: 2D AXI domain for the Ishape3D test case, here f = 3e10, so skin effect is present, the mesh is finer near the right boundary.

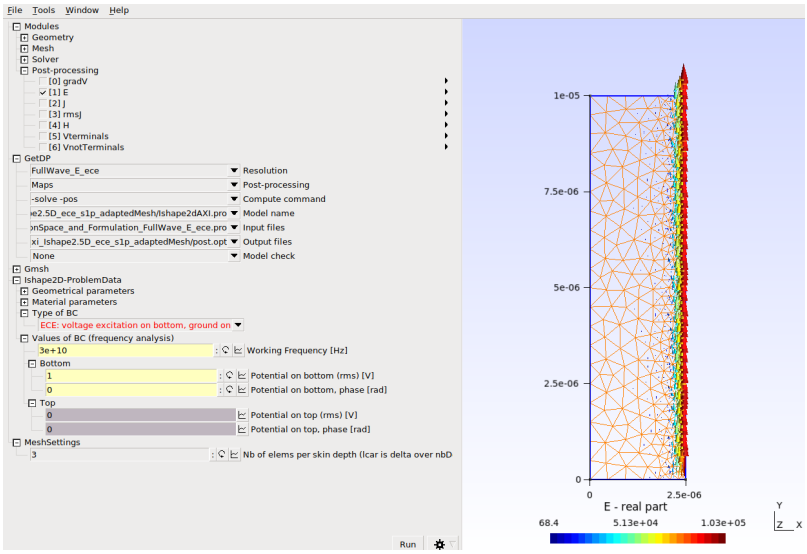


Figure 25: **03axi_Ishape2.5D_ece_s1p_adaptedMesh**: **E** field for the situation described in Fig. 24.

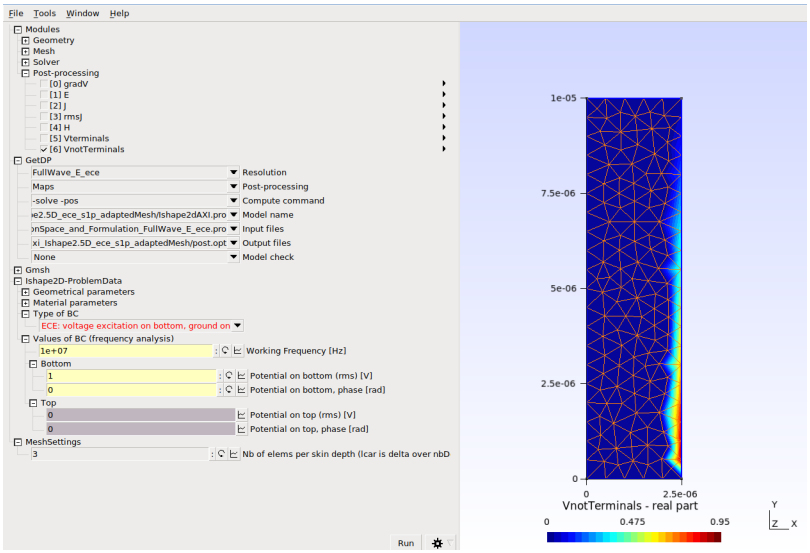


Figure 26: **03axi_Ishape2.5D_ece_s1p_adaptedMesh**: V in the 2D AXI model, for f =1e7 Hz.

4 LC

4.1 Test 04_LC_GeometryInStepFile_ece_s1p: ECE formulation, solve several frequencies, geometry in a STEP file.

This is the example described in [OH21]. The authors provided the step file, which you can find in the folder **04_LC_GeometryInStepFile_ece_s1p**.

The frequencies are set in **LC_data.pro**: 10 points between fmin = 1 kHz, fmax = 80 kHz.

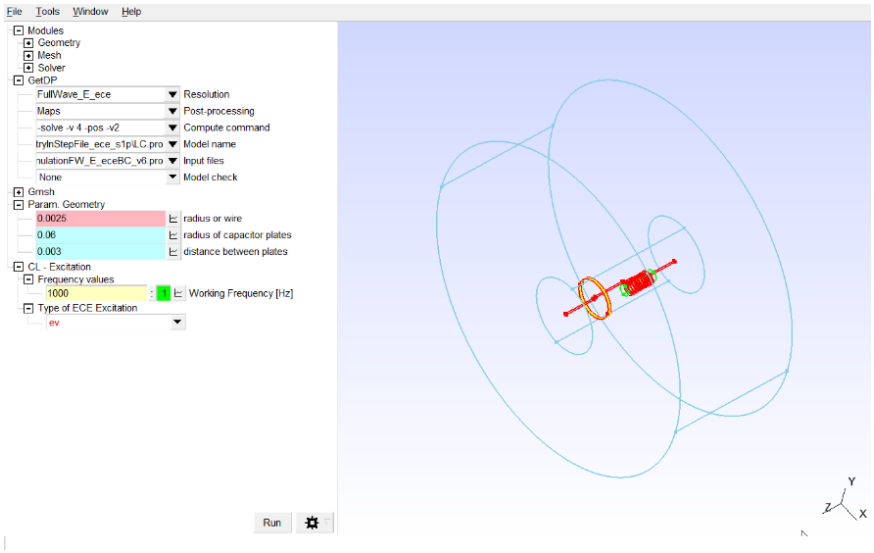


Figure 27: **04_LC_GeometryInStepFile_ece_s1p**: LC test, with the geometry in the STEP file.

Just run the **LC.pro** file. The result obtained for 187560 dofs can be found in the folder **Results_log/04_LC/results9mar22/res_LC_GeometryInStepFile/FWeceBC_voltExc** when voltage excitation was used.

For instance, the **test_Y_RI.s1p** is

```
# Hz Y RI R 50
1000 0.0003516313952579203 0.2114244959595226
1627.250609936924 0.001035845806292761 0.3513803462372218
2647.944547540091 0.00357235280653572 0.6059373855629445
4308.869380063769 0.01784118774430161 1.170036455418785
7011.610326847304 0.2822621657652343 3.708672395856161
11409.64718100232 0.5516073272169192 -3.993703209313727
18566.35533445113 0.08046091682579434 -1.237822757593129
30212.11304229127 0.02977220783987155 -0.6474613541863959
49162.67937555176 0.01266569445264791 -0.3788176066059606
80000.00000000003 0.005567725147544824 -0.2293839213794721
```

You can use the Matlab code **main_Irms_PhaseDiff_LC_GeometryInStepFile.m** so that to compare this numerical result with the reference result (the circuit) from [OH21]. It is important to note that

the comparison makes sense since the shape and dimensions of the air box in this test are exactly the same as in the reference paper.

4.2 Test 04_LC_ParametricGeometryInGeoFile_ece_s1p: ECE formulation, solve several frequencies, parametric geometry in a geo file.

This is, in principle, the same problem as the one in the previous section, but the geometry was built in a parametric way, by using the built-in kernel of gmsh.

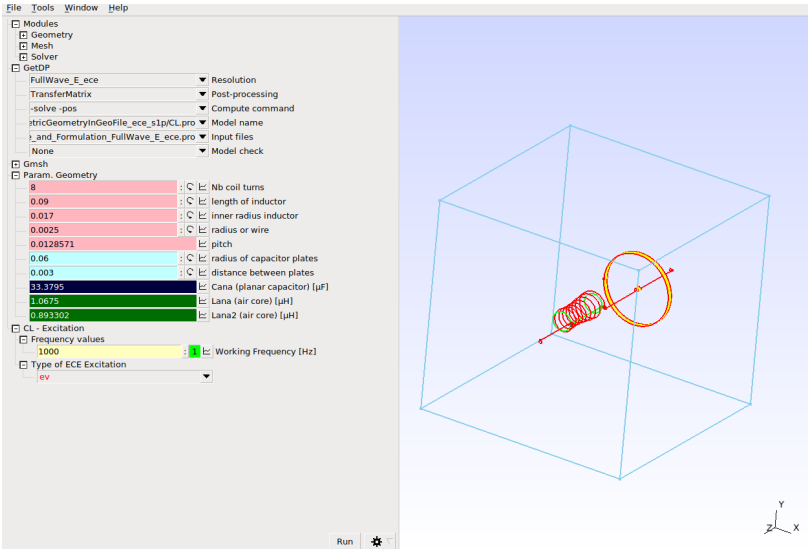


Figure 28: **04_LC_GeometryInStepFile_ece_s1p**: LC test, with a parametric geometry described with Brep.

Just run the **CL.pro** file. The result obtained for 330511 dofs can be found in the folder
Results_log/04_LC/results9mar22/res_LC_ParametricGeometryInGeoFile/FWeceBC_voltExc when voltage excitation was used.

You can use the Matlab code
main_Irms_PhaseDiff_LC_ParametricGeometryInGeoFile.m so that to compare this numerical result with the reference result (the circuit) from [OH21]. It is important to note that the comparison makes sense only partially here, since the shape and dimensions of the air box in this test are not the ones in the reference paper (Fig. 28). See the Conclusion section for more comments on this issue.

5 Call gmsh and getdp from Matlab

In the folder **05_SimpleExampleCallingOnelabFromMatlab** you can find two subfolders:

- **01b_Ishape2D_ece_s1p** - is the classical way of working in onelab.
- **01b_Ishape2D_ece_s1p_callFromMatlab** - the same example run from Matlab, system calls are used for gmsh and getdp.

You should carefully compare the files, as well as read the **readme.tex** file that you will find in that folder.

6 Integrate gmsh and getdp with model order reduction based on AFS and VF

Details about the adaptive frequency sampling (AFS) can be found in [CILD12]. AFS is using which is used in conjunction with the vector fitting procedure (VF) [GS99]. The VF code was downloaded from https://www.sintef.no/projectweb/vectorfitting/ and it is included in the **MatlabSources** folder.

The example discussed in this section can be found in the folder **06_LC_GeometryInStepFile_ece_s1p_callFromMatlab_AFS**.

6.1 Call from Matlab

The information about the parameters for AFS has to be given in a file, here it is called **set_infofreqAFS_vf_LC.m**. The function in this file will return two structures, one for the frequency information, and the other for the VF procedure.

In the example below, the list of frequencies starts with the two end points: 1 kHz and 80 kHz (line 6), the AFS flag is set to True in line 7 (otherwise only the frequency values in the frequency_points vector will be used). The AFS error is set (line 9) to 1 % (which is a reasonable value because it is a local one – see [CILD12]. The order of the reduced transfer function is increased from 1 and the maximum possible value is set to 40 (lines 15 and 16). The other parameters are specific to VF, see [GS99].

An important parameter is the number of points in which the transfer function will be evaluated, below this value is set to 100 (line 27). This is a cheap evaluation, because it is the evaluation of a transfer function for which we know the poles, the residues, and the constant terms. It is important here to have many points so that to catch how the frequency characteristics looks like.

```
1 % File set_infofreqAFS_vf_LC.m
2
3 function [frequency_data, avfitParams] =  set_infofreqAFS_vf_LC()
4
5 frequency_data.frequency_unit = 'Hz';
6 frequency_data.frequency_points = linspace(1000,80000,2);
7 frequency_data.AFS.flag = (1 == 1);
8 frequency_data.AFS.max = 1000;
9 frequency_data.AFS.err = 0.01;
10 frequency_data.AFS.type = 'vfitlinf';  % 'lin', 'poly','vfit' sau 'vfitlinf'
11 method = 'onelab';
12
13 % avfit params
14 avfitParams = [];
15 avfitParams.minOrder = 1;
16 avfitParams.maxOrder = 40;
17 avfitParams.tol = 1e-4;
18 avfitParams.noCalls = 2;
```

```
19 avfitParams.typeTransferFunction = 2; %1 for Strictly Proper, 2 for Proper, 3 for Improper
20 avfitParams.graphicsVfit = 'no'; %'yes', otherwise treated as 'no'
21 avfitParams.stopCriterion = 0; %0 - components for what it is (codestar way), 1 - Frobenius S, 2 -
       Components S
22 % only 0 tested for the moment.
23
24 avfitParams.fileout = 'test.cir';
25 avfitParams.idx = 0; % useful only when combined with sys2snp
26 avfitParams.save.flag = 'no'; % compute and save vfit approximation in many points during iterations; '
       yes' or 'no'
27 avfitParams.save.nopoints = 100; % no of poins for the computation of vfit approximation
28 avfitParams.save.type = 'lin'; % how these points are placed: lin or otherwise log in the freq range
29 avfitParams.save.flagMA = 'no';
30 avfitParams.save.flagMAinDB = 'no';
31 avfitParams.startFrom = 'lowestOrder';
32
33 end
```

You have also to write a Matlab main file, where you set the excitation type, see the file **mainMatlab_LC_AFS.m**, line 11.

```
1  function mainMatlab_LC_AFS()
2
3  restoredefaultpath;
4  sourcespath = genpath('D:\Gabriela\OneLab\mytests\ECEforOnelab\ECEinOnelab_E_2021\MatlabSources');
5  addpath(sourcespath);
6
7  close all;
8  clearvars; format long
9  clc
10
11 Flag_AnalysisType = 0;  % 0 for ev, 1 for ec
12
13 % simulare
14 if (Flag_AnalysisType == 0)
15     snp_info.ptype = 'Y';
16     fileNameFinal = 'res\FWeceBC_voltExc\LC_Y_RI'; % extensions will be added
17 else
18     snp_info.ptype = 'Z';
19     fileNameFinal = 'res\FWeceBC_crtExc\LC_Z_RI'; % extensions will be added
20 end
21
22 snp_info.pformatfile = 'RI';
23 snp_info.Z0 = 50;
24 snp_info.nports = 1; % no of terminals
25 snp_info.tol_poles = 1e-2;
26
27 [frequency_data, avfitParams] =  set_infofreqAFS_vf_LC();
28 [frequency_response,frequency_data,trfct] = ...
29     afs_vf_onelab(frequency_data,snp_info,avfitParams,Flag_AnalysisType); %old sys2snp_vf3
30 trfct
31
32 % write all the solutions
33 pformat = 'RI';
34 Z0 = 50;
35 if frequency_data.AFS.flag
36     snp_filename = strcat(fileNameFinal,'_onelab_AFSvfitlinf_',...
37         sprintf('%f',frequency_data.AFS.err),'_erVF',...
38         sprintf('%f',avfitParams.tol ),'.s',num2str(snp_info.nports),'p');
39 else
40     snp_filename = strcat(fileNameFinal,'.s1p');
41 end
42 writesnp_v2(snp_filename, frequency_data.frequency_points, ...
43     frequency_response, snp_info.ptype, ...
44     frequency_data.frequency_unit,snp_info.Z0,pformat);
45 if (Flag_AnalysisType == 0)
46     system(strcat('move *.s1p res\FWeceBC_voltExc\.'));
47     system(strcat('move *.cir res\FWeceBC_voltExc\.'));
48 else
49     system(strcat('move *.s1p res\FWeceBC_crtExc\.'));
50     system(strcat('move *.cir res\FWeceBC_crtExc\.'));
51 end
52
53 end
```

The call to onelab is done in the **afs_vf_onelab.m** function, which calls the **solve_onelab.m** function which is in the problem folder, where you can see the system calls to gmsh (line 12) and getdp (line 23).

```
1  function value = solve_onelab(freqs,Flag_AnalysisType)
2  % path to gmsh and getdp should be set
3
4  if (Flag_AnalysisType == 0)
5      fileName = 'res\FWeceBC_voltExc\test_Y_RI.s1p';
6  else
7      fileName = 'res\FWeceBC_crtExc\test_Z_RI.s1p';
8  end
9
10 NbFreqs = length(freqs);
11
12 system(sprintf('gmsh LC.geo -setnumber Flag_AnalysisType %d -3 -v 2 ', Flag_AnalysisType));
13
14 for k = 1:NbFreqs
15     disp(k);
```

```
16      disp(freqs(k));
17      freqsk = freqs(k);
18      if k == 1
19          firstFreq = 1;
20      else
21          firstFreq = 0;
22      end
23      system(sprintf(['getdp LC.pro -setnumber Freq %g freqs %g -setnumber Flag_AnalysisType %d -
            setnumber firstFreq %d -solve FullWave_E_ece -v 3 -pos TransferMatrix'],...
24          freqsk, freqsk, Flag_AnalysisType, firstFreq));
25  end
26  [ptype,pformatfile,frequency_points,value,Z0,...
27      freq_unit,nports] = loadsnp(strcat('./',fileName));
28  system(strcat('del',strcat(' .\',fileName)));
29  end
```

6.2 Looking at the results

Information is written in the Matlab console as well as in several files. For instance, we can see that a final reduced order model of order 4 was obtained, and only the evaluation of 7 points was needed, info written in the console:

```
Poles
-9.984919457926706e+04 + 0.000000000000000e+00i
-4.096739685388231e+05 + 0.000000000000000e+00i
-1.442744503658989e+03 + 5.667359773381735e+04i
-1.442744503658989e+03 - 5.667359773381735e+04i

Number of computed frequencies = 7

The transfer function is a structure obtained as output

trfct =
order: 4
poles: [4x1 double]
residues: [1x1x4 double]
kinf: 1.765940138632837e-04
prop: 0
```

The files are written in the folder **res/FWeceBC_voltExc**. You can look at what happened throughout all the AFS iterations.

The following files were created:

```
LC_Y_RI_onelab_AFSvfitlinf_0.010000_erVF0.000100.s1p
vfit_final_iter1_ord1.s1p
vfit_final_iter2_ord2.s1p
vfit_final_iter3_ord4.s1p
vfit_final_iter4_ord4.s1p
vfit_final_points1_ord1.s1p
vfit_final_points2_ord2.s1p
vfit_final_points3_ord4.s1p
vfit_final_points4_ord4.s1p
test.cir
```

You can inspect them easily with `snpdiff_tool command`, but let's look only at the final result. The files ***points*** contain the frequencies that were evaluated with onelab.

In this case in this case **vfit_final_points4_ord4.s1p** is the last file, it contains 7 frequencies.

The file **LC_Y_RI_onelab_AFSvfitlinf_0.010000_erVF0.000100.s1p** is just a copy of **vfit_final_points4_ord4.s1p**, the name however is generated automatically and keeps the information used in the simulation:

vfit_final_points4_ord4.s1p | vfit_final_iter4_ord4.s1p

```
# GHz Y RI R 50
9.9999999999999995e-07 3.5164558291968489e-04 2.1142293823792699e-01
1.0875000000000000e-05 8.3077751302172798e-01 -5.0012553807739089e+00
2.0750000000000000e-05 6.2405066310669471e-02 -1.0480952819610769e+00
3.0624999999999999e-05 2.9041526477945570e-02 -6.3772414658543453e-01
4.0500000000000002e-05 1.7670535720792170e-02 -4.6623528347448989e-01
6.0250000000000001e-05 8.9562674500379587e-03 -3.0668911096284962e-01
8.0000000000000007e-05 5.5545118117485970e-03 -2.2951979017521790e-01
```

Figure 29: **06_LC_GeometryInStepFile_ece_s1p_callFromMatlab_AFS**: LC test, with AFS and VF - only 7 points evaluated with FEM.

The files ***iter*** contain the result of evaluating the transfer function in the number of points you have set (here 100):

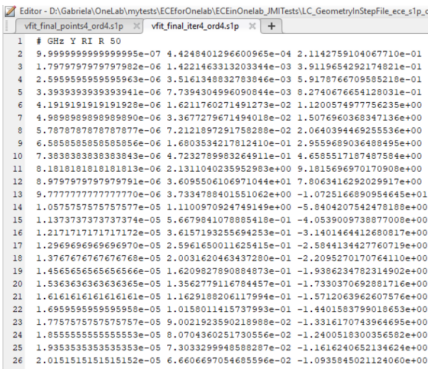


Figure 30: **06_LC_GeometryInStepFile_ece_s1p_callFromMatlab_AFS**: LC test, with AFS and VF - 100 cheap evaluations of the obtained rational approximation.

Now, by using the snpdiff-tool command we can display the points computed by onelab and the points for the rational expression (Fig. 31).

Finally we can compare the result with AFS (and 7 FEM evaluations chosen in an adaptive way) with the previous result, where 10 frequency points were used sampled equidistantly in the frequency range (Fig. 32).

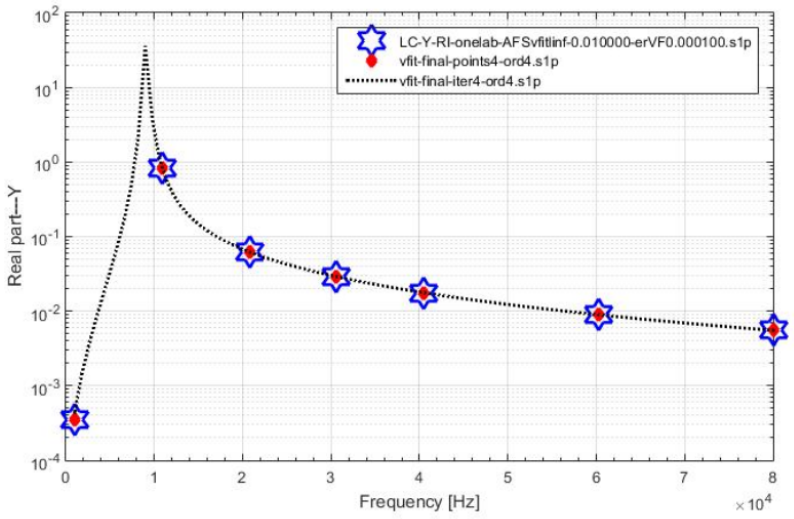


Figure 31: **06_LC_GeometryInStepFile_ece_s1p_callFromMatlab_AFS**: LC test, with AFS and VF - only 7 points evaluated with FEM (red / star points) and the rational approximation (black curve - obtained with 100 cheap evaluations).

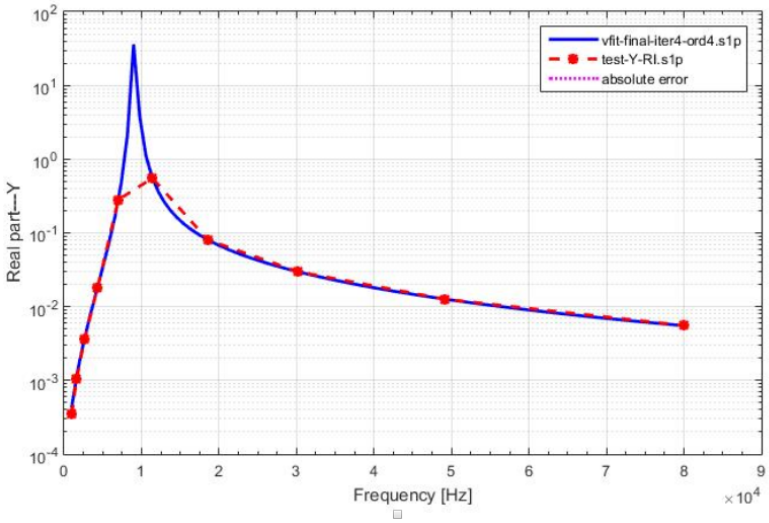


Figure 32: **06_LC_GeometryInStepFile_ece_s1p_callFromMatlab_AFS**: LC test, AFS approximation (black curve - 7 FEM evaluations) and the 10 equidistant points (FEM evaluations) in the frequency range.

These results can be found in the folder **Results_log/04_LC/results27oct21_afs_figuresJMI/res**. You can find there some other useful scrips that allowed us to compare these results with the reference paper. Just explore the folder while reading the paper [CIS22].

7 Conclusions - important notes about parameter extraction

The main advantage of ECE BC for the Maxwell equations is that the ports are well defined, without ambiguity, and compatible with the circuit terminals, even for RF devices. There is no restriction on the field regime (DC to full wave, even including nonlinear media). For MIMO systems, the hybrid excitation is obtained in a natural way.

It is important to be aware that ECE BC for parameter extraction can be applied only to a simply connected subdomain, obtained after partitioning the domain corresponding to a whole system in parts that do not overlap or do not have holes. From this point of view, the LC test above has to be considered with care, since it is like that we know that the current return path is through the boundary. Here, we rather extract a partial inductance. In this case, which is rather coming from a circuit view and not from a field view, the magnetic energy is concentrated around the coils which has a very permeable core. If we imagine that we extend the airbox, the magnetic energy increases and thus the inductance, when the air box would tend to infinity, the inductance would also tend to infinity and thus it will have no meaning. In the modeling of a real device, and not an academic one, it is important to start from a field domain, with computational domain and boundary conditions chosen in a correct way. For instance, how the device is connected to its source or its outside circuit.

As a general conclusion, ECE BC is a result of a careful domain partitioning. When the extracted models are interconnected, the loops ("holes") thus obtained must not be new field sources, i.e. there should not exist a magnetic field crossing them. The issue of the LC test, which has that airbox around it, comes from the fact that the terminals are not close to each other. That is why the extracted inductance is unbounded when the air box goes to infinity. A remedy for this academic example is to bend the conductor so that the terminals are on the same side of the airbox, close to each other. This will cure the problem of an unbounded inductance.

From the inductance extraction point of view, the LC test problem is not proper, as the extracted inductance depends on the size of the airbox and tends to infinity if the airbox goes to infinity. Indeed, the airbox boundary is the support of the current return path. This issue is inherent to the model and independent of the boundary conditions. However, we have adopted the same airbox (size and shape) as in the reference paper [OH21], we can thus extract and compare the values to those in the reference paper.

An illustrative explanation related to the discussion above can be found here http://literature.cdn.keysight.com/litweb/pdf/5989-9526EN.pdf, and Agillent calles this an "unphysical port". This is strongly related to the discussion here https://www.protoexpress.com/blog/current-return-path-signal-integrity/.

At measurements and parameter extraction, the terminals of the device under test (DUT) have to be placed as closed as possible, so that big loops do not appear that involved the wires that connect the measured port and the measuring device.

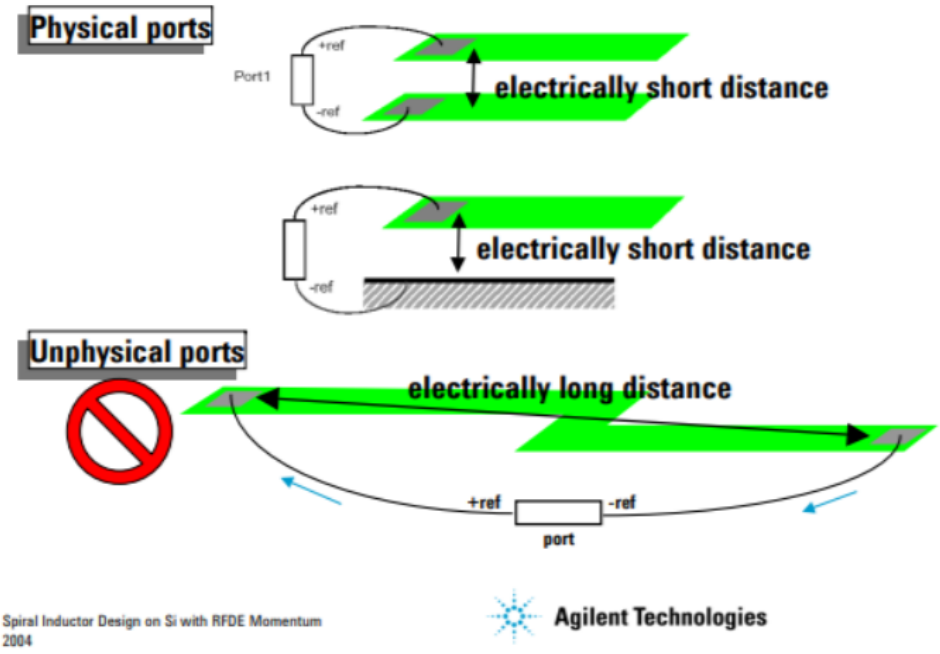


Figure 33: Unphysical port: Figure from
http://literature.cdn.keysight.com/litweb/pdf/5989-9526EN.pdf

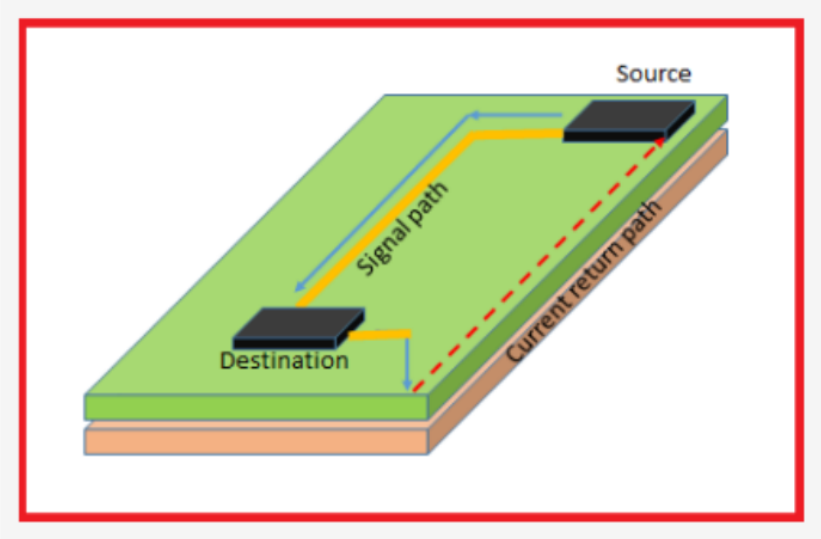


Figure 34: Importance of current return path for signal integrity. Figure from
https://www.protoexpress.com/blog/current-return-path-signal-integrity/.

References

[CILD12] G. Ciuprina, D. Ioan, I.A. Lazar, and C.B. Dita. Vector fitting based adaptive frequency sampling for compact model extraction on hpc systems. *IEEE Transactions on Magnetics*, 48(2):431–434, 2012.

[CIPL21] G. Ciuprina, D. Ioan, M. Popescu, and S. Lup. Electric circuit element boundary conditions in the finite element method for full-wave frequency domain passive devices. In M. van Beurden, N. Budko, and W. Schilders, editors, *Scientific Computing in Electrical Engineering, Springer Series Mathematics in Industry, vol 36*. Springer, 2021. https://doi.org/10.1007/978-3-030-84238-3_10.

[CIS22] G. Ciuprina, D. Ioan, and R.V. Sabariego. Electric circuit element boundary conditions in the finite element method for full-wave passive electromagnetic devices. *Journal of Mathematics in Industry, Springer Open*, 12(7):1–13, 2022. https://doi.org/10.1186/s13362-022-00122-1.

[DG] P. Dular and C. Geuzaine. Getdp reference manual: the documentation for getdp, a general environment for the treatment of discrete problems. http://getdp.info.

[GS99] B. Gustavsen and A. Semlyen. Rational approximation of frequency domain responses by vector fitting. *IEEE Trans. Power Deliv.*, 14(3):2109–2121, 1999.

[OH21] J. Ostrowski and R. Hiptmair. Frequency-stable full Maxwell in electroquasistatic gauge. *SIAM J. Sci. Comput., Computational Methods in Science and Engineering*, 43(4):B1008–B1028, 2021.

A Ishape2D

2D domain, of dimensions $(2a) \times l$. It is useful to imagine a depth h, it will be used when computing R and L parameters Axes are chosen so the the computational domain is $x \in [-a, a]$, $y \in [0, l]$ The plane $y = 0$ (intersected with the domain) is a terminal. The plane $y = l$ is the second terminal, grounded.

It is like we have an infinitely extended bar, we excited it on the bottom surface (which is infinitely extended), but we focus only at a zone of area $2a \times h$. The electric field is solely oriented along Oy and depends only on x. The magnetic field is oriented solely along Oz and depends only on x. The quantities do not depend on y.

The problem is symmetrical from the point of view of the electric field and anti-symmetric from the point of view of the magnetic field

Analytic solution – dc:

$$Rh_{cc} = l/(\sigma 2ah)$$

$$Lh_{cc} = \mu la/(6h)$$

Rh_{cc} – obvious; Lh_{cc} – can be computed with an energetic reasoning

Analytic solution – strong skin effect

$$\delta = \sqrt{2./(2\pi. * f\mu\sigma)}$$

$$Rh_{pp} = l./(\sigma 2\delta h)$$

obvious

$$Lh_{pp} = l\mu\delta/(4h)$$

- energetically

The Helmholtz vector equation in H is in this case (2D, homogeneous domain) becomes a differential ODE of second order for Hz, which is easy to be solved. After imposing symmetry conditions, Hz has the form 2 C sh(gamma x), where C is computed from the imposed boundary condition given by I. The electric field E is computed from the complex form of the magnetic circuit law, and the complex power received by the domain is computed easily, by integrating the Poynting vector.

```
gamma_cplx_patrat = 1i*omega*mu0.*(sigma + 1i*omega*eps0);
gamma_cplx = sqrt(gamma_cplx_patrat);
shga = sinh(gamma_cplx*a);
chga = cosh(gamma_cplx*a);
crt = 1;
Ey_complex = (gamma_cplx./(sigma + 1j*omega*eps0)).*(chga./shga)*crt/(2*h);
Hz_complex = crt/(2*h);
P_ap_cplx_lineic = 2*Ey_complex*conj(Hz_complex)*l*h;
Ranalitic = real(P_ap_cplx_lineic);
X_h = imag(P_ap_cplx_lineic);
Lanalitic = X_h./omega;
```

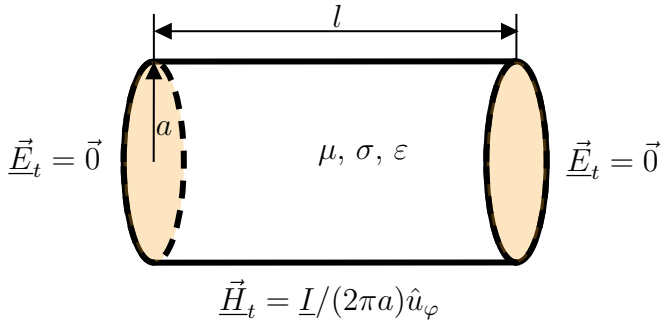
B Ishape3D

A cylindrical conductor, of length l, circular cross-section of radius a, with linear and homogenous material, of conductivity σ, permeability μ, permittivity ε, is excited with an alternating current, of frequency ω and root mean square I, initial phase zero (so complex representation $\underline{I} = I$).

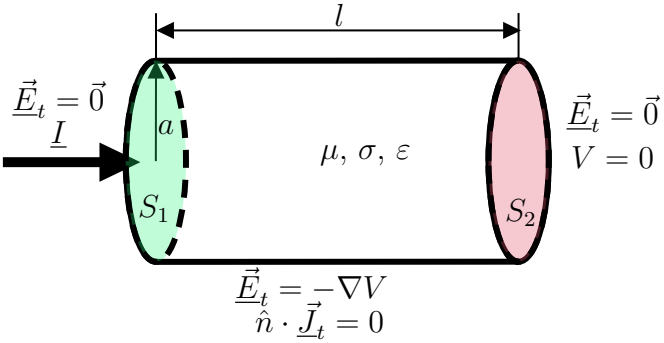
This est problem has the following advantages:

- it admits an equivalent formulation with classical boundary conditions (with $\vec{\underline{E}}_t$ and $\vec{\underline{H}}_t$);
- it admits an analytical solution, so we can compare numerical results with analytic ones.

Formulation with classical boundary conditions



Formulation with ECE boundary conditions



Analytic solution

Below we will asume that the conductor is infinitely long and insulated from other conductors. In FEM the model is 3D, with a length l.

Let's start with the extreme cases which are easily to obtain: the DC case and the case with strong skin depth.

Line resistance - extreme cases[1]

- in DC

$$R_{l,cc} = \frac{1}{\sigma \pi a^2} \tag{1}$$

- in the case of strong skin depth,

$$R_{l,pp} = \frac{1}{\sigma 2\pi a \delta} \tag{2}$$

[1] cc - curent continuu (direct current); pp - efect pelicular pronunţat (strong skin depth)

where

$$\delta = \sqrt{\frac{2}{\omega\mu\sigma}} \tag{3}$$

is the skin depth.

Internal line inductance - extreme cases

- in DC

$$L_{l,cc} = \frac{\mu}{8\pi} \tag{4}$$

- in the case of strong skin depth

$$L_{l,pp} = \frac{\mu\delta}{4\pi a} \tag{5}$$

The D.C line inductance can be computed energetically, as follows. If we denote by I the total current, the magnetic field inside the conductor has the modulus $H(r) = Ir/(2\pi a^2)$ and the magnetic energy density $w_m(r) = \mu H^2/2$. The energy density is integrated in the whole volume, considering a volume element with only one dimension infinitely small $\mathrm{d}v = 2\pi r l\,\mathrm{d}r$, where r goes from 0 to a. The magnetic energy stored inside the conductor is $W_m = \mu I^2 l/(16\pi)$ and thus expression (4) is obtained.

In the case of a strong skin depth, the current can be assumed distributed along a circular crown of internal radius $a-\delta$ and external radius a. The internal magnetic field is non-uniform, for $r = a$ it has the modulus

$$\underline{H}(a) = \frac{\underline{I}}{2\pi a}$$

and for $r = a - \delta$,

$$\underline{H}(a-\delta) = 0.$$

This is in fact an MQS problem, and thus

$$\vec{E} = -\nabla V - \frac{\partial \vec{A}}{\partial t},$$

so in harmonic case

$$\underline{\vec{E}} = -\nabla \underline{V} - j\omega \underline{\vec{A}}.$$

The component $\nabla \underline{V}$: we can assume that it correspond to a current that is uniformly distributed on the circular grown of width δ, so $-\nabla \underline{V} = \underline{I}/(\sigma 2\pi a\delta)\vec{k}$, where we assumed that the axis of the cylinder is Oz.

To compute the magnetiv vector potential $\underline{\vec{A}} = \underline{A}\vec{k}$, we considere a closed curve having a rectangular shape, with an edge of length h along the cylinder axis, and the edge that is paralel to this placed on the cylinder surface. We express the magnetic field in two ways. $\oint_{\Gamma_1} \underline{\vec{A}} \cdot \vec{dl} = \int_{S_{\Gamma_1}} \underline{\vec{B}} \cdot \vec{dA}$, and thus we obtain $\underline{A}(a)h = \mu_0 h \int_{a-\delta}^{a} \underline{H}(r)\,\mathrm{d}r$ and because δ is very small, we can use an approximate formula (similar to the trapezoidal rule) and thus $\underline{A}(a) = \mu_0 \underline{H}(a)/2\delta = \mu_0 \underline{I}/(2\pi a)/2\delta = \mu_0 \underline{I}\delta/(4\pi a)$. It follows that

$$\underline{\vec{E}}(a) = \left(\frac{\underline{I}}{\sigma 2\pi a\delta} + j\omega\frac{\mu_0 \underline{I}\delta}{4\pi a}\right)\vec{k}$$

In order to compute the complex power transferred from outside to inside, we will need

$$|\underline{\vec{E}}(a) \times \underline{\vec{H}}^*(a)| = \left(\frac{\underline{I}}{\sigma 2\pi a\delta} + j\omega\frac{\mu_0 \underline{I}\delta}{4\pi a}\right)\frac{\underline{I}^*}{2\pi a} = \frac{I^2}{2\pi a}\left(\frac{1}{\sigma 2\pi a\delta} + j\omega\frac{\mu_0\delta}{4\pi a}\right)$$

and from this it follows that the complex power transferred to a conductor of length l is

$$\underline{P}_{ap} = |\underline{\vec{E}}(a) \times \underline{\vec{H}}^*(a)|2\pi a l = I^2 l\left(\frac{1}{\sigma 2\pi a\delta} + j\omega\frac{\mu_0\delta}{4\pi a}\right) = I^2 l(R_l + j\omega L_l)$$

and thus we obtain the expressions of the line resistance and inductance for a strong skin effect

$$R_l = \frac{1}{\sigma 2\pi a \delta} \qquad L_l = \frac{\mu_0 \delta}{4\pi a}$$

Numerical example

Let's consider a conductor with a diameter $d = 5\mu$m, length 10 μm, conductivity $\sigma = 6.6 \cdot 10^7$ S/m (Aluminiu), and an extremely large frequency range, from 0.1 Hz, to 100GHz. Figures 35 and 36 hold the frequency depence of these extreme formulas.

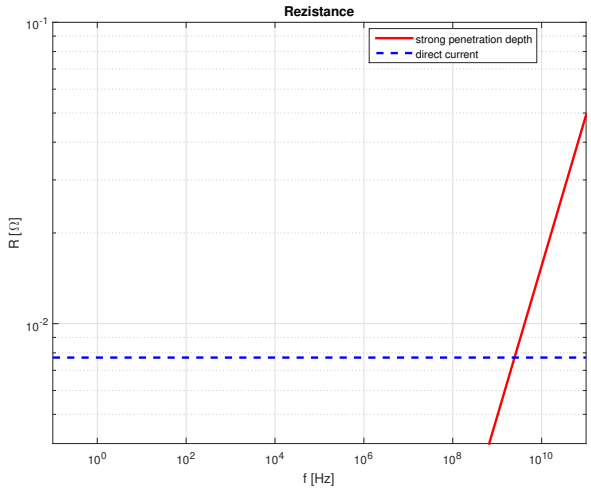


Figure 35: Resistance - extreme cases: direct current (blue) and strong skin depth (red).

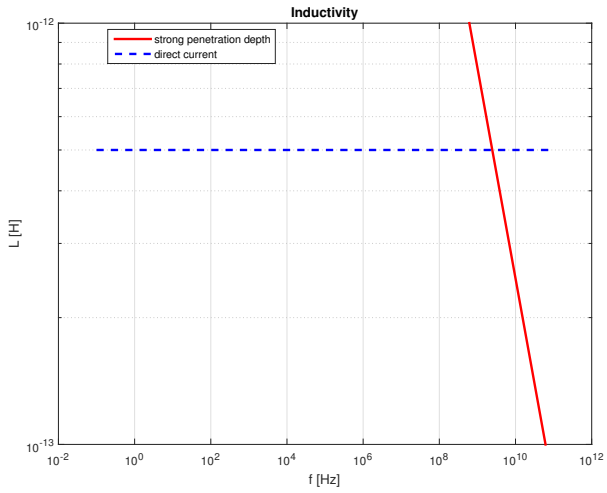


Figure 36: Internal inductance - extreme cases: direct current (blue) and strong skin depth (red).

The general case

We will show how the analytic formulas are obtained considering a FW regime, homogenous conductor, no internal sources. Te MQS regime can be obtained by setting $\varepsilon = 0$.

Assume the conductor is excited with a current $i(t) = I\sqrt{2}\sin(\omega t)$, the current excitation being the natural one in proving the analytic formulas.

The complex representation of the current is $\underline{I} = I$.

Do to the infinite extension of the conductor and the symmetry with respect to the Oz axis assumed to be the cylinder axis, the field quantities will depend only on the radial coordinate r of a cylindrical system of coordinates, as follows:

- the electric field is axial
$$\underline{\vec{E}} = \underline{E}(r)\hat{u}_z \tag{6}$$

- the magnetic field is transverse, only with an azimuth component
$$\underline{\vec{H}} = \underline{H}(r)\hat{u}_\varphi \tag{7}$$

We will focus only on the field inside the conductor, where:

$$\operatorname{curl}\underline{\vec{H}} = (\sigma + j\omega\varepsilon)\underline{\vec{E}} \tag{8}$$
$$\operatorname{curl}\underline{\vec{E}} = -j\omega\mu\underline{\vec{H}} \tag{9}$$

Since the material is homogenous, from (8) it follows that

$$\operatorname{div}\underline{\vec{E}} = 0 \tag{10}$$

We apply curl to equation (9) and we combine it with (8):

$$\operatorname{curl}\operatorname{curl}\underline{\vec{E}} = -j\omega\mu(\sigma + j\omega\varepsilon)\underline{\vec{E}} \tag{11}$$

It follows that

$$\operatorname{grad}\operatorname{div}\underline{\vec{E}} - \triangle\underline{\vec{E}} = -j\omega\mu(\sigma + j\omega\varepsilon)\underline{\vec{E}} \tag{12}$$

and we obtain the Helmholtz complex vector equatio

$$\triangle\underline{\vec{E}} - j\omega\mu(\sigma + j\omega\varepsilon)\underline{\vec{E}} = 0 \tag{13}$$

Let's denote the complex propagation constant γ defined by

$$\underline{\gamma}^2 = j\omega\mu(\sigma + j\omega\varepsilon) \tag{14}$$

In an algebraic form, the classical notation is

$$\underline{\gamma} = \alpha + j\beta \tag{15}$$

where α is called attenuation constant and β is the wave number. It can be proven that they have the expressions

$$\alpha = \omega\sqrt{\frac{\mu\varepsilon}{2}\left[\sqrt{\left(\frac{\sigma}{\omega\varepsilon}\right)^2 + 1} - 1\right]}, \tag{16}$$

$$\beta = \omega\sqrt{\frac{\mu\varepsilon}{2}\left[\sqrt{\left(\frac{\sigma}{\omega\varepsilon}\right)^2 + 1} + 1\right]} \tag{17}$$

We denote by $\delta = 1/\alpha$ the skin depth.

If $\sigma >> \omega\varepsilon$ (MQS regime) $\alpha = \beta = \sqrt{\omega\mu\sigma/2}$ and it follows that $\delta = \sqrt{2/(\omega\mu\sigma)}$.

Consequently, the Helmholtz vector equation is

$$\triangle\underline{\vec{E}} - \underline{\gamma}^2\underline{\vec{E}} = 0 \tag{18}$$

which is equivalent to three scalar Helmholtz equations, one for each direction of the coordinate system

In our case, since the electric field has only an axial component (relation (6)), the Helmholtz equation becomes

$$\triangle \underline{E} - \underline{\gamma}^2 \underline{E} = 0 \tag{19}$$

In cylindrical coordinates, the Laplace operator has the expression

$$\triangle \cdot = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial \cdot}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 \cdot}{\partial \varphi^2} + \frac{\partial^2 \cdot}{\partial z^2} \tag{20}$$

Since $\underline{E}$ depends only on r, the Helmholtz equation becomes

$$\frac{1}{r}\frac{\mathrm{d}}{\mathrm{d}r}\left(r\frac{\mathrm{d}\underline{E}}{\mathrm{d}r}\right) - \underline{\gamma}^2 \underline{E} = 0 \tag{21}$$

and hence

$$\frac{\mathrm{d}^2\underline{E}}{\mathrm{d}r^2} + \frac{1}{r}\frac{\mathrm{d}\underline{E}}{\mathrm{d}r} - \underline{\gamma}^2 \underline{E} = 0 \tag{22}$$

The solution of this equation is (see appendix C)

$$\underline{E}(r) = AJ_0(j\underline{\gamma}r) + BY_0(j\underline{\gamma}r) \tag{23}$$

but since the computational domain includes the axis $r = 0$ and the field cannot be unbounded on the axis, it means that $B = 0$ and the solution is

$$\underline{E}(r) = AJ_0(j\underline{\gamma}r) \tag{24}$$

The integration constant A can be obtained by computing the magnetic field and imposing its value on the cylinder surface, value that correspond to the imposed current.

The magnetic field $\underline{H}$ is obtained from the Faraday's law of induction (9)

$$\vec{\underline{H}} = -\frac{1}{j\omega\mu}\mathrm{curl}\,\vec{\underline{E}} \tag{25}$$

Expressing the curl in cylindrical coordinates and considering the symmetry of the field, it follows that

$$\underline{H}\hat{u}_\varphi = -\frac{1}{j\omega\mu}\left(-\frac{\mathrm{d}\underline{E}}{\mathrm{d}r}\right)\hat{u}_\varphi \tag{26}$$

and thus

$$\underline{H}(r) = \frac{1}{j\omega\mu}\frac{\mathrm{d}}{\mathrm{d}r}\left(AJ_0(j\underline{\gamma}r)\right) = \frac{A}{j\omega\mu}j\underline{\gamma}J_0'(j\underline{\gamma}r) = -A\frac{\underline{\gamma}}{\omega\mu}J_1(j\underline{\gamma}r) \tag{27}$$

At the cylinder surface, the tangential component of the magnetic field is exactly $\underline{H}(a)$, which has to be continuous (when passing from inside to outside), and consequently it has to be equal to

$$\underline{H}(a) = \frac{\underline{I}}{2\pi a} = \frac{I}{2\pi a} \tag{28}$$

It follows that

$$-A\frac{\underline{\gamma}}{\omega\mu}J_1(j\underline{\gamma}a) = \frac{I}{2\pi a} \tag{29}$$

and hence

$$A = -\frac{\omega\mu I}{2\pi a\underline{\gamma}J_1(j\underline{\gamma}a)} \tag{30}$$

Consequently, the complex representations of the electric and magnetic fields are:

$$\underline{E}(r) = -\frac{\omega\mu I}{2\pi a\underline{\gamma}J_1(j\underline{\gamma}a)}J_0(j\underline{\gamma}r) \tag{31}$$

$$\underline{H}(r) = \frac{\omega\mu I}{2\pi a\underline{\gamma}J_1(j\underline{\gamma}a)}\frac{\underline{\gamma}}{\omega\mu}J_1(j\underline{\gamma}r) \tag{32}$$

In order to compute the complex power transferred by the electromagnetic field to this domain, we have to compute the Poyinting vector on the surface

$$\vec{\underline{S}}(a) = \vec{\underline{E}}(a) \times \vec{\underline{H}}(a)^* = \underline{E}(a)\underline{H}(a)^*(\hat{u}_z \times \hat{u}_\varphi) = \underline{E}(a)\underline{H}(a)^*(-\hat{u}_r) = -\underline{E}(a)\underline{H}(a)^*\hat{u}_r \tag{33}$$

The complex power transferred from outside to inside, to a domain of length l is

$$\underline{P}_{ap} = \int_\Sigma \vec{\underline{S}}(a) \cdot \vec{n}_i \, \mathrm{d}A \tag{34}$$

where $\vec{n}_i$ is the normal oriented from outside to inside. Only the lateral surface of the cylinder contributes to this result, the final formulas are:

$$\begin{aligned}\underline{P}_{ap} &= l(2\pi a)\underline{E}(a)\underline{H}(a)^* = \\ &= l(2\pi a)\cdot(-1)\frac{\omega\mu I}{2\pi a\underline{\gamma}J_1(j\underline{\gamma}a)}J_0(j\underline{\gamma}a)\cdot\left(\frac{I}{2\pi a}\right)^* = \\ &= -l\frac{\omega\mu I^2}{(2\pi a)\underline{\gamma}}\frac{J_0(j\underline{\gamma}a)}{J_1(j\underline{\gamma}a)}\end{aligned} \tag{35}$$

From this expression, we can compute the resitance and the inductance, as frequency dependent quantities

$$\underline{P}_{ap} = P + jQ = RI^2 + j\omega LI^2 \tag{36}$$

and finally

$$R = -l\frac{\omega\mu}{2\pi a}\mathrm{Real}\left(\frac{J_0(j\underline{\gamma}a)}{\underline{\gamma}J_1(j\underline{\gamma}a)}\right) \tag{37}$$

$$L = -l\frac{\mu}{2\pi a}\mathrm{Imag}\left(\frac{J_0(j\underline{\gamma}a)}{\underline{\gamma}J_1(j\underline{\gamma}a)}\right) \tag{38}$$

Let's check the formulas, by computing the asymptotic limits at high frequencies. At high frequencies, the ration between J_0/J_1 goes to $-j$.

In MQS $\beta = \alpha$ and thus $\gamma = \alpha(1+j)$, $\alpha = 1/\delta$ unde $\delta = \sqrt{2/(\omega\mu\sigma)}$

The expression in the parentheses of (37) and (38) becomes at strong skin depth

$$\frac{J_0(j\underline{\gamma}a)}{\underline{\gamma}J_1(j\underline{\gamma}a)} = \frac{-j}{\alpha(1+j)} = \delta\frac{-j(1-j)}{2} = -\delta\frac{1+j}{2} \tag{39}$$

Consequently, what we obtain at strong skin depth from these general formulas is

$$R_{pp} = l\frac{\omega\mu}{2\pi a}\frac{\delta}{2} \tag{40}$$

and replacing $\omega\mu = 2/(\sigma\delta^2)$ we end at

$$R_{pp} = \frac{l\delta}{4\pi a}\frac{2}{\sigma\delta^2} = \frac{l}{\sigma 2\pi a\delta} \tag{41}$$

which is the same as (2).

For the imaginary part

$$\omega L_{pp} = l\frac{\omega\mu}{2\pi a}\frac{\delta}{2} \tag{42}$$

and thus $\omega L_{pp} = R_{pp}$. It follows that

$$L_{pp} = l\frac{\mu}{2\pi a}\frac{\delta}{2} = l\frac{\mu\delta}{4\pi a} \tag{43}$$

which is the same as (5).

The results are shown in figures 37 and 38.

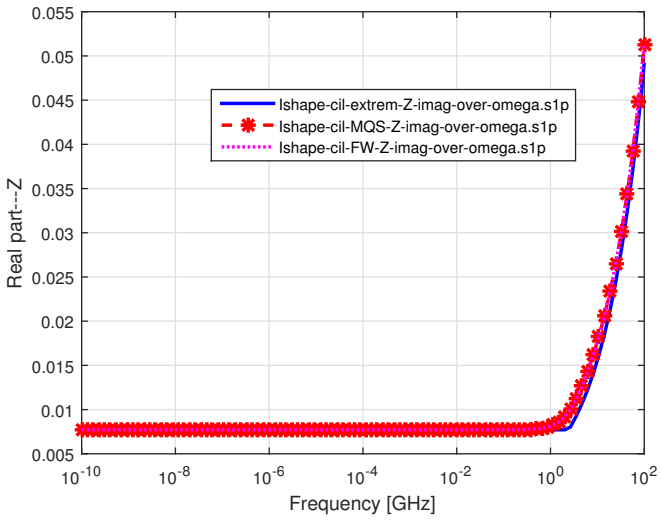


Figure 37: Resistance - analytic formula (red) vs. extreme formulas (blue).

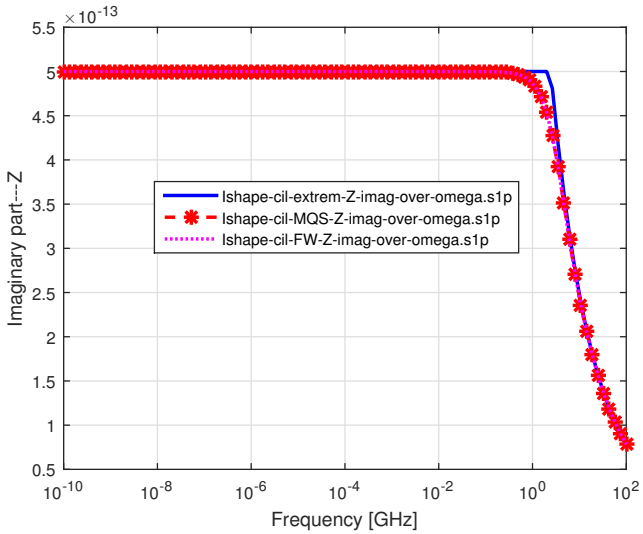


Figure 38: Internal inductance - analytic formula (red) vs. extreme formulas (blue).

And a zoom-in and loglog scales (so that to better see the asymptotic behavior) in the figures 39 si 40.

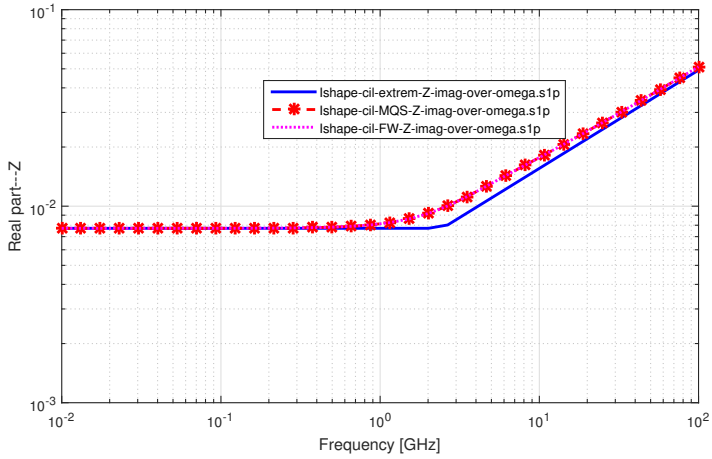


Figure 39: Resistance - analytic formula (red) vs. extreme formulas (blue)- zoom and loglog plot.

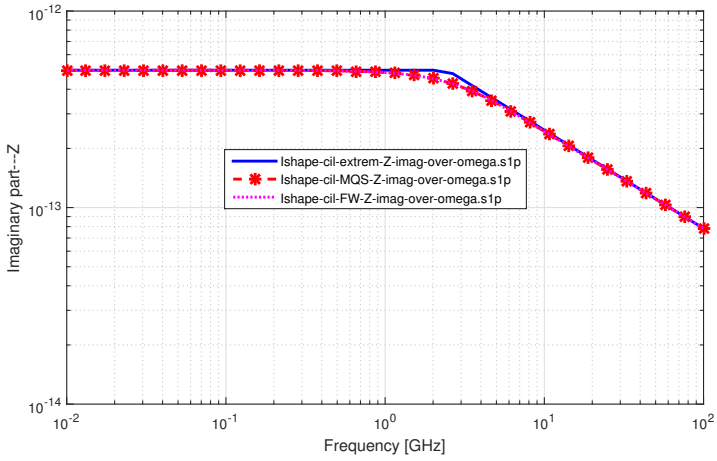


Figure 40: Internal inductance - analytic formula (red) vs. extreme formulas (blue)- zoom and loglog plot.

C Bessel equations and function Bessel - minimal!

The Bessel differential equation is

$$x^2\frac{\mathrm{d}^2y}{\mathrm{d}x^2}+x\frac{\mathrm{d}y}{\mathrm{d}x}+(x^2-\nu^2)y=0 \tag{44}$$

where $\nu\in\mathbb{R}$ si $x\in\mathbb{C}$, $y:\mathbb{C}\to\mathbb{C}$. If we divide by x^2 the equation is equivalent to

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2}+\frac{1}{x}\frac{\mathrm{d}y}{\mathrm{d}x}+\left(1-\frac{\nu^2}{x^2}\right)y=0 \tag{45}$$

The solutions of this equation are given by

- first kind Bessel functions of order ν denoted by $J_n(x)$
- second kind Bessel functions of order ν denoted by $Y_n(x)$

The solution of the equation is

$$y(x)=AJ_\nu(x)+BY_\nu(x) \tag{46}$$

where A and B are complex constants that have to be computed by imposing conditions that ensure the uniquness of the solution.

For details and representations of Bessel function, see for instance

- http://mathworld.wolfram.com/BesselFunctionoftheFirstKind.html
- http://mathworld.wolfram.com/BesselFunctionoftheSecondKind.html.

It is important to note that second kind Bessel functions are unbounded in the origin,

In order to link this to the equations obtained using EM field, let's consider the change of variable

$$x=j\underline{\gamma}r \tag{47}$$

and thus it follows that

$$\frac{\mathrm{d}y}{\mathrm{d}x}=\frac{1}{j\underline{\gamma}}\frac{\mathrm{d}y}{\mathrm{d}r} \tag{48}$$

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2}=\frac{1}{-\underline{\gamma}^2}\frac{\mathrm{d}^2y}{\mathrm{d}r^2} \tag{49}$$

and equation (45) is re-written as

$$\frac{\mathrm{d}^2y}{\mathrm{d}r^2}+\frac{1}{r}\frac{\mathrm{d}y}{\mathrm{d}r}-\left(\underline{\gamma}^2+\frac{\nu^2}{r^2}\right)y=0 \tag{50}$$

The solution of this equation is

$$y(r)=AJ_\nu(j\underline{\gamma}r)+BY_\nu(j\underline{\gamma}r) \tag{51}$$

In particular, the equation

$$\frac{\mathrm{d}^2y}{\mathrm{d}r^2}+\frac{1}{r}\frac{\mathrm{d}y}{\mathrm{d}r}-\underline{\gamma}^2y=0 \tag{52}$$

will have the solution

$$y(r)=AJ_0(j\underline{\gamma}r)+BY_0(j\underline{\gamma}r) \tag{53}$$

An property that is useful for us is the expression of the derivative of the first kind Bessel function and order 0:

$$J_0'(x)=-J_1(x) \tag{54}$$

D Understanding the GetDP file for FW, with ECE, formulation with E inside and V on the boundary

For details see [CIS22], here it is just the minimal information so that we can explain the matching with the GetDP description.

D.1 Weak formulation (continuous)

Find $\underline{\mathbf{E}} \in \mathcal{H}_E$ and $\underline{V} \in \mathcal{H}_V$, so that

$$a(\underline{\mathbf{E}}, \underline{\mathbf{E}}') = f(\underline{\mathbf{E}}'), \quad \forall \underline{\mathbf{E}}' \in \mathcal{H}_{E,0}; \tag{55}$$

$$\oint_{\partial S_k} \mathbf{H} \cdot \mathbf{dl} = \underline{I}_k, \quad k \in \mathcal{I}_c; \qquad \underline{\mathbf{E}}_t = -\nabla_2 V, \quad \text{on} \quad \partial\Omega,$$

where

$$a(\underline{\mathbf{E}}, \underline{\mathbf{E}}') = \int_\Omega \left[(\nu \nabla \times \underline{\mathbf{E}}) \cdot (\nabla \times \underline{\mathbf{E}}') + \mathrm{j}\omega(\sigma + \mathrm{j}\omega\varepsilon) \underline{\mathbf{E}} \cdot \underline{\mathbf{E}}' \right] \mathrm{d}x, \tag{56}$$

$$f(\underline{\mathbf{E}}') = \mathrm{j}\omega \sum_{k \in \mathcal{I}_c} \underline{V}'_k \underline{I}_k; \tag{57}$$

and $\underline{\mathbf{E}}'_t = -\nabla_2 V'$, where $\underline{V}' \in \mathcal{H}_{V,0}$,

$$\begin{aligned}
\mathcal{H}_E = \{ \mathbf{u} \in \mathcal{H}(\mathrm{curl}, \Omega) | \, & \mathbf{n} \times (\mathbf{u} \times \mathbf{n}) = -\nabla_2 \underline{V}' \text{ on } \partial\Omega, \quad \underline{V}' \in \mathcal{H}_V \\
& \mathbf{n} \times (\mathbf{u} \times \mathbf{n}) = \mathbf{0} \text{ on } \cup_{k=1}^m S_k \} \\
\mathcal{H}_{E,0} = \{ \mathbf{u} \in \mathcal{H}(\mathrm{curl}, \Omega) | \, & \mathbf{n} \times (\mathbf{u} \times \mathbf{n}) = -\nabla_2 \underline{V}' \text{ on } \partial\Omega, \quad \underline{V}' \in \mathcal{H}_{V,0} \\
& \mathbf{n} \times (\mathbf{u} \times \mathbf{n}) = \mathbf{0} \text{ on } \cup_{k=1}^m S_k \} \\
\mathcal{H}_V = \{ u \in \mathcal{H}(\mathrm{grad}, \partial\Omega) | \, & u = \underline{V}_k \text{ on } S_k, \ k \in \mathcal{I}_v, \\
& u = \text{constant(unknown, floating potentials) on } S_k, \ k \in \mathcal{I}_c \} \\
\mathcal{H}_{V,0} = \{ u \in \mathcal{H}(\mathrm{grad}, \partial\Omega) | \, & u = 0 \text{ on } S_k, \ k \in \mathcal{I}_v \\
& u = \text{constant(unknown, floating potentials) on } S_k, \ k \in \mathcal{I}_c \}.
\end{aligned}$$

D.2 Weak formulation (discrete, FEM)

In [CIPL21] we used a simplicial mesh (tetrahedrons in 3D, triangles in 2D), numerical test functions $\vec{N}_k$ that correspond to edge elements of order (0,1), and degrees of freedom that represent the complex representations of voltages $\underline{U}_k$ along the edges. In the case of using classical boundary conditions, the numerical trial function is approximated as

$$\underline{\vec{E}} = \sum_{j=1}^{\mathrm{Ne}} \underline{U}_j \vec{N}_j, \tag{58}$$

where Ne is the total number of edges in the domain, including its boundary.

In the case of using ECE boundary conditions, the function space where the trial function is searched for is curl free on the domain boundary, where nodal unknowns V_k and test functions φ_k are needed. The connection between the approximations inside and on the boundary can be done at the level at test functions. For instance, since for one element

$$\vec{N}_k^{(e)} = \varphi_i^{(e)} \nabla \varphi_j^{(e)} - \varphi_j^{(e)} \nabla \varphi_i^{(e)}, \tag{59}$$

it follows that the numerical trial function when using ECE boundary conditions is approximated as

$$\underline{\vec{E}} = \sum_{j=1}^{\text{NeInt}} \underline{U}_j \vec{N}_j - \sum_{j=1}^{\text{NnBnd}} \underline{V}_j \nabla \varphi_j \tag{60}$$

where NeInt is the total number of edges that are strictly inside the domain and NnBnd is the total number of nodes on the boundary. Some of the nodes that are on the boundary belong to the same terminal, which must be equipotential. The corresponding terms in (60) have to be grouped together, and the final expression of numerical solution with respect to the the trial functions is:

$$\underline{\vec{E}} = \sum_{j=1}^{\text{Neint}} \underline{U}_j \vec{N}_j - \sum_{j=1}^{\text{NnBndNotTerm}} \underline{V}_j \nabla \varphi_j - \sum_{k=1}^{m} \left(\underline{V}_k \sum_{j=1}^{\text{NnTermK}} \nabla \varphi_j \right), \tag{61}$$

where m is the total number of terminals, and NnTermK are the number of nodes that are covered by terminal k.

D.3 GetDP - function space and formulation

GetDP keywords are in red; comments are in green. Only the black words are user defined.

This is the essence of our contribution (hence the motto on the first page). The rest of the document was dedicated to its testing and validation.

```
FunctionSpace {

  { Name Hcurl_E; Type Form1;
    BasisFunction {
      { Name se; NameOfCoef ee; Function BF_Edge;
                Support Dom_FW ; Entity EdgesOf[All, Not Sur_FW]; }
      { Name sn; NameOfCoef vn; Function BF_GradNode;
                Support Dom_FW ; Entity NodesOf[Sur_FW, Not Sur_Terminals_FWece]; }
      { Name sf; NameOfCoef vf; Function BF_GradGroupOfNodes;
                Support Dom_FW ; Entity GroupsOfNodesOf[Sur_Terminals_FWece]; }
    }
    GlobalQuantity {
      { Name TerminalPotential; Type AliasOf        ; NameOfCoef vf; }
      { Name TerminalCurrent  ; Type AssociatedWith; NameOfCoef vf; }
    }

    SubSpace {
      { Name dv ; NameOfBasisFunction {sn}; } // Subspace, it maybe use in equations or post-pro
    }

    Constraint {
      { NameOfCoef TerminalPotential; EntityType GroupsOfNodesOf;
        NameOfConstraint SetTerminalPotential; }
      { NameOfCoef TerminalCurrent; EntityType GroupsOfNodesOf;
        NameOfConstraint SetTerminalCurrent; }
    }
  }

}

Formulation {

  { Name FullWave_E_ece; Type FemEquation;
    Quantity {
      { Name e;  Type Local; NameOfSpace Hcurl_E; }
      { Name dv; Type Local; NameOfSpace Hcurl_E[dv]; } // Just for post-processing issues
      { Name V;  Type Global; NameOfSpace Hcurl_E[TerminalPotential]; }
      { Name I;  Type Global; NameOfSpace Hcurl_E[TerminalCurrent]; }
    }
    Equation {
      // \int_D curl(\vec{E}) \cdot  curl(\vec{e}) dv
      Galerkin { [ nu[] * Dof{d e} , {d e} ]; In Vol_FW; Jacobian Vol; Integration Int; }

      // \int_D j*\omega*(\sigma + j*\omega*\epsilon) \vec{E} \cdot \vec{e} dv
      Galerkin { DtDof   [ sigma[]   * Dof{e} , {e} ]; In Vol_FW; Jacobian Vol; Integration Int; }
      Galerkin { DtDtDof [ epsilon[] * Dof{e} , {e} ]; In Vol_FW; Jacobian Vol; Integration Int; }

      //    alternative // j*\omega*sum (vk Ik);  for k - current excited terminals
      //    GlobalTerm {DtDof [ -Dof{I} , {V} ]; In SurBCec; }

      // j*\omega*sum (vk Ik);  for k - all terminals so that the currents through the terminals will
           be computed as well
      GlobalTerm {DtDof [ -Dof{I} , {V} ]; In Sur_Terminals_FWece; }
    }
  }

}
```

Relations (55), (56) and (57) are the key to understand the lines 42-52. Line 42 is the first term of (56). Line 45 is the second term of (56). Line 46 is the third term of (56). Line 52 corresponds to (57), with a minus, because in GetDP the relation is written as an equality with zero.

Relation (61) is the key to understand lines 5 - 10. Lines 5 and 6 represent the first sum in (61) (sum for all the edges inside the domain). Lines 7 and 8 represent the second sum in (61) (sum for all the nodes on the boundary but which are not on the terminals). Lines 9 and 10 represent the third sum in (61) (the basis functions for the group of nodes of each electrode are added and a global basis function is thus obtained). The minus signs in (61) are treating in the pre- and post-processing parts. The user imposes values for some terminal potentials, but inside the code, immediately, those value are multiplied by -1. Thet GetDP syntax does not allow you to use minus in lines 7 and 9.

Here it is a more detailed matching between math concepts and objects and GetDP notations.

- Hcurl_E = the name of the discrete space for the electric field strength, it was denoted by $\mathcal{H}_E$;
- Form1 = means that the unknown field is a vector quantity;
- ee = the coefficient in the numerical solution expansion. It is $\underline{U}_j$ in (61), i.e. a voltage along an edge inside the domain;
- se = the name of the basis function associated to the coefficient ee. It is $\vec{N}_j$ in(61);
- BF_Edge = specify the chosen basis function, in this case edge basis function of order 1;
- Dom_FW = a group of geometrical entities (volumes, surfaces) defined by the user, in this case it is the domain Ω and its boundary $\partial\Omega$;
- Sur_FW = a group of geometrical entities (surfaces) defined by the user, in this case it is the domain boundary $\partial\Omega$;
- vn = the coefficient in the numerical solution expansion. It is $-\underline{V}_j$ in (61), i.e. minus the potential in a node of the boundary;
- sn = the name of the basis function associated to the coefficient vn. It is $\nabla\varphi_j$ in(61);
- BF_GradNode = specify the chosen basis function, in this case the gradient of a nodal basis function of order 1;
- Sur_Terminals_FWece = a group of geometrical entities (surfaces) defined by the user, in this case it is the union of the terminal surfaces $\cup_{k=1}^{m} S_k$;
- vf = the coefficient in the numerical solution expansion. It is $-\underline{V}_k$ in (61), i.e. minus the potential of a terminal;
- sf = the name of the basis function associated to the coefficient vf. It is $\sum_{j=1}^{NnTermK}(\nabla\varphi_j)$ in(61);
- BF_GradGroupOfNodes = specify the chosen basis function, in this case the sum of gradient of nodal basis functions of order 1 for all the nodes of a terminal;
- TerminalPotential it is a constrained, defined elsewhere, which imposes values of voltages for the voltage excited terminals;
- TerminalCurrent it is a constrained, defined elsewhere, which imposes values of currents for the current excited terminals;

- unknown e (trial function) is Dof{ e};
- curl applied to the unknown e is Dof{d e};
- Vol_FW = a group of geometrical entities (volumes) defined by the user, in this case it is the domain Ω;
- the test function (denoted with prime in the mathematical formulas) is {e};
- curl applied to the test function is {d e};
- unknown current of voltage excited terminal (trial function) is Dof{ I};
- the test function associated to the unknown potential on the boundary (denoted with prime in the mathematical formulas) is {V}.

Line 13: TerminalPotential is an alias for vf, and thus it refers to the potential of electrodes.

Line 14: TerminalCurrent is an alias for the variable associated with vf, in this case the current entering the terminals.

Line 22: some vn values are set according to voltage excited terminals in the Constraints object in the pro file.

Line 24: some terminals are excited in current. To better understand this, we should look at the Constraint object, but this is in the .pro file which is problem dependent. For example, here it is how it looks for a SISO case (e.g. Ishape).

```
1  Constraint{
2          // ece BC
3      { Name SetTerminalPotential; Type Assign;  // voltage excited terminals
4            Case {
5             { Region Ground; Value 0.; }
6               If((Flag_AnalysisType==0))
7                     { Region Terminal; Value VTerminal1[]; }
8               EndIf
9             }
10     }
11     { Name SetTerminalCurrent; Type Assign;   // current excited terminals
12           Case {
13             If((Flag_AnalysisType==1))
14                    { Region Terminal; Value ITerminal1[]/h2Ddepth; }  // here the depth is needed
15             EndIf
16           }
17     }
18 }
```

D.4 3D, 2D, 2.5D (AXI)

The function space and formulation described above are valid for all the cases: 3D, 2D and 2D AXI (also known as 2.5 D).

The difference is solved by the Jacobian object:

```
1 Jacobian {
2   { Name Vol;
3     Case {
4       If(Flag_Axi && modelDim < 3)
5         { Region All; Jacobian VolAxi; }  //VolAxi  or VolAxiSqu ??? which one??
6       Else
7         { Region All; Jacobian Vol; }
8       EndIf
9     }
10  }
11  { Name Sur;
12    Case {
13      If(Flag_Axi && modelDim < 3)
14        { Region All; Jacobian SurAxi; }
15      Else
16        { Region All; Jacobian Sur; }
17      EndIf
18    }
19  }
20 }
```

and by the correct computation of currents in the postprocessing part:

```
PostProcessing {

  { Name FW_E_ece; NameOfFormulation FullWave_E_ece;
    Quantity {
      //......
      { Name I;
        Value {Term { [ -1*{I}*h2Ddepth ]; In Sur_Terminals_FWece; }}
      }
}
```

You just have to set the correct "h2Ddepth" depending on your problem. We did this in the *_data.pro file:

```

modelDim = 2; //
Flag_Axi = 1; // 1 for AXI - it makes sense only for modelDim = 2

If ((modelDim == 2)&&(Flag_Axi == 0)) // 2D
    h2Ddepth = h;
ElseIf ((modelDim == 2)&&(Flag_Axi == 1))    // 2D AXI
    h2Ddepth = 2*Pi;
Else // 3D
    h2Ddepth = 1;
EndIf
```