pyelmer Example: 3D Electrostatic Capacitance

Werner Simbürger
werner.simbuerger@hppi.de

Arved Enders-Seidlitz
arved.enders-seidlitz@ikz-berlin.de

August 2, 2021

Contents

1 Problem Description

The 3D electrostatic capacitance including fringing between top metal and bottom metal (Fig. 1) shall be calculated with Elmer FEM [1], Gmsh [2] and Python [3] including the pyelmer [4] package. ParaView [5] is used to review the calculated FEM results, such as vector fields.

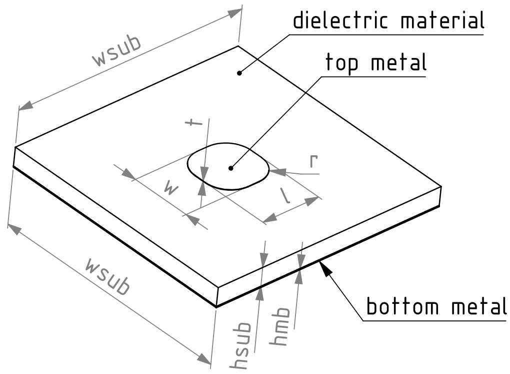

Figure 1: 3D electrostatics capacitance

The top metal has a width w, length l and metal layer thickness t. The corners of the top metal patch have a fillet r. The dielectric material layer has a relative permittivity ϵ_r, thickness $hsub$, width and length $wsub$. The bottom metal layer has a layer thickness hmb. The entire stack is surrounded by air.

2 Solution

2.1 Analytical Calculation

To verify the Elmer FEM computation of the capacitance, we can use a special geometrical case $w = l$, $r \to w/2 = l/2$ and $t = hmb \ll hsub$, which results in a circular microstrip disk, as shown in Fig. 2 and presented in the paper [6].

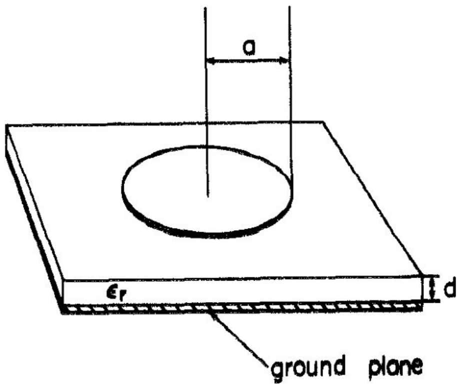

Figure 2: Effects of fringing fields on the capacitance of a circular microstrip disk [6]

The authors of the paper [6] present a quite accurate approximation formula for the capacitance within the parameter range of $\epsilon_r = 1$ to 8.5 and $d/a = 0.1$ to 0.5 as follows:

$$C \approx \frac{a^2 \pi \epsilon_r \epsilon_0}{d} \left\{ 1 + \frac{2d}{\pi \epsilon_r a} \left[\ln\left(\frac{a}{2d}\right) + (1.41\epsilon_r + 1.77) + \frac{d}{a}(0.268\epsilon_r + 1.65) \right] \right\} \tag{1}$$

where the dielectric permittivity of vacuum $\epsilon_0 = 8.854 \times 10^{-12}\,\mathrm{C\,V^{-1}\,m^{-1}}$.

2.2 Elmer FEM Calculation

Prerequisite is the installation of the following software packages[1]:

1. Elmer FEM solver [1]
2. Python ≥ 3.7 [3] including the pyelmer package [4]
3. ParaView [5] (optional)

The installation of the Elmer FEM solver and the Python/pyelmer package is mandatory. Gmsh is part of the Python pyelmer package. However, for debugging purposes the independent Gmsh software can be installed in parallel. ParaView is optional, but it is extremely helpful to review the Elmer FEM calculated results (`*.vtu`), such as vector fields.

Copy the files `3d_electrostatic_capacitance.py`, `my_simulations.yml`, `my_solvers.yml` and `my_materials.yml` in a local folder and run `3d_electrostatic_capacitance.py`.

The entire geometry definition, mesh creation, Elmer FEM calculation and results evaluation process is executed by a single Python/pyelmer top level script `3d_electrostatic_capacitance.py`, which is listed in Sect. 4.1.

3 Test Case and Results

3.1 Test Case Parameter

The structure shown in Fig. 1 is simulated with the following parameters:

Parameter	**Value**	**Unit**	**Description**
w	6	mm	top metal patch width
l	6	mm	top metal patch length
t	0.035	mm	top metal layer thickness
r	2.95	mm	top metal patch corner radius (fillet)
$hsub$	1.524	mm	dielectric material layer thickness
$wsub$	24	mm	dielectric material layer width
ϵ_r	3.55		dielectric material layer relative permittivity (Rogers RO4003C material [7])
hmb	0.1	mm	bottom metal layer thickness

Table 1: Test case parameter

3.2 Capacitance Calculation Results

Analytical solution (Eqn. 1)	$C = 1.012\,\mathrm{pF}$
Elmer FEM computation	$C = 1.062\,\mathrm{pF}$

Table 2: Analytical solution and Elmer FEM computation result of the capacitance

The Elmer FEM computation result listing, CPU-time of the Elmer solver in [second], excluding meshing:

```
Errors: []
Warnings: []
Statistics: {'CPU-time': 86.6, 'real-time': 86.6}
Relative Change: 5.10E-12
##############################################
w: 6E-3
l: 6E-3
r: 2.95E-3
Capacitance: 1.062E-12
##############################################
```

[1] available for Windows, Linux, Mac. References shown mainly for Windows.

The agreement of the Elmer FEM computation result with the approximation formula Eqn. 1 is very good. However, it is very important to have small mesh size in the regions of high field gradients (Fig. 3). Otherwise the Elmer FEM computation result may be significant wrong despite fast FEM algorithm convergence.

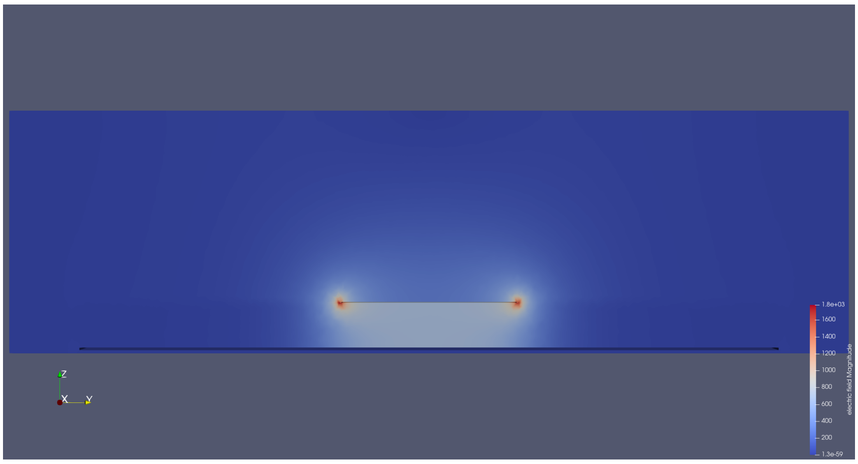

Figure 3: Calculated electric field magnitude (cross section of Fig. 4)

3.3 Discussion of the Elmer FEM Solution and Boundary Conditions

1. The surrounding air of the structure is modeled by a proper air box, as shown in Fig. 4. The size of the air box is calculated in line 123 and 124 (see Sect. 4.1). The top height of the air box is designed properly in order to cover the fringing field in the air.

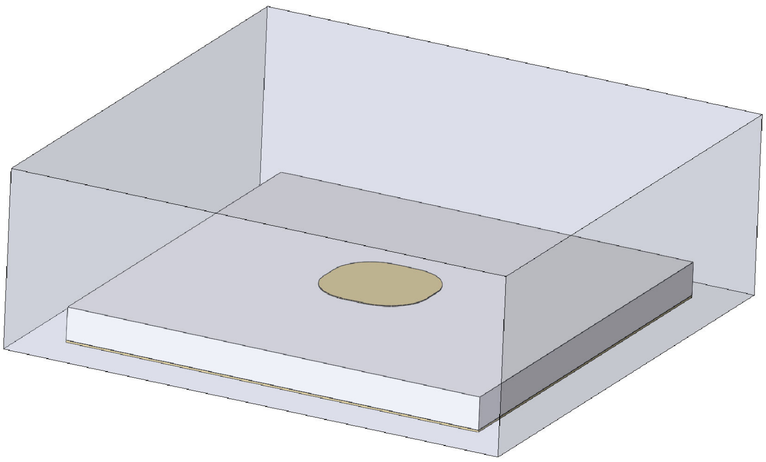

Figure 4: 3D model including air box

2. The top metal and bottom metal layer are considered as perfect conductor. Therefore, these two bodies are not meshed in order to keep the required FEM meshing time and FEM computation time as low as possible. As a consequence, the structure consists of only two bodies (see line 128) which are required to be meshed:

   (a) dielectric material layer

   (b) air box

3. To calculate the electrostatic capacitance between bottom metal and top metal, all surfaces need to be assigned to a potential value. All bottom metal body surfaces (there are 6 surfaces) are assigned to a potential of 0 V. All top metal body surfaces (there are 10, because of the corner radius) are assigned to a potential of 1 V. Further, it is very important to assign the correct value of the potential difference (line 269). Otherwise a wrong capacitance value will be calculated.

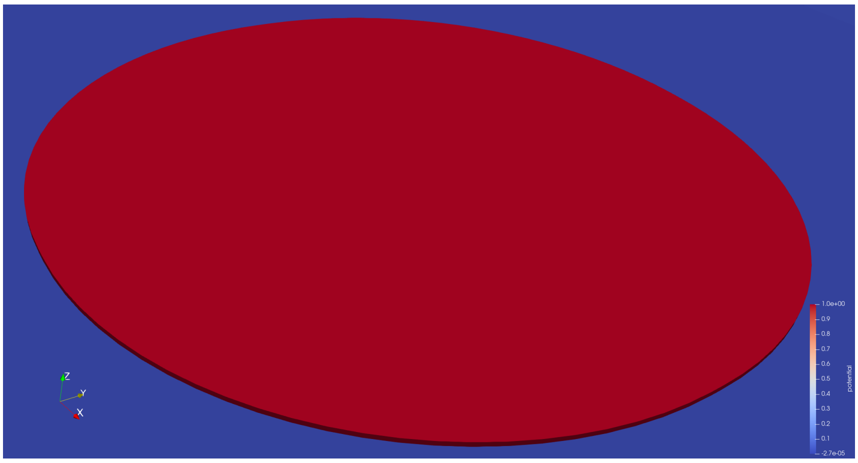

Figure 5: Potential distribution (top metal zoom view)

4. The correct assignment of physical bodies and boundary conditions are essential. The assignment of physical groups is semi-automated in this example, e.g. line 193, 194. The entity numbers of surfaces and physical group assignment is done automatically (e.g. line 205, 206). The general step approach of 3D modeling using pyelmer and Gmsh can be summarized as follows:
   (a) 3D modeling of the structure using Gmsh API functions
   (b) Assign (only one time!) physical groups of required bodies
   (c) Assign boundaries using the style `id_for_elmer = add_physical_group(...)`. This can be done with automatic extraction using functions such as `getMyEntitiesInBoundingBox` or by manual assignment, e.g. `ph_sub, ph_ab` at line 193, 194.
   (d) Meshing using Gmsh API functions
   (e) Use Gmsh GUI to check if the assignment (step 2 above) is correct: press Ctrl+V, afterwards click through the physical groups
   (f) For the definition of bodies use the `id_for_elmer` from step 2 above

5. Meshing notes: The set up of the mesh including optimization is done using the Gmsh application programming interface (API) [8]. The geometric meshing parameters could be further optimized. However, it works well at this point of time, but the FEM computation time and meshing time could be further reduced by an optimized mesh.

4 Source File Listings

4.1 Python (pyelmer) Script

The following Python script is based on pyelmer [4] and is used to set up and run the Elmer FEM simulation from Python including model geometric parameter definition, meshing, definition of boundary conditions, Elmer FEM simulation and extraction of results.

Listing 1: pyelmer code «3d_electrostatic_capacitance.py»

```python
import os
import gmsh
from pyelmer import elmer, post
from pyelmer import execute
from pyelmer.gmsh import add_physical_group
from math import floor, log10


###########################################################################
### Settings
###########################################################################

# The following switches defined are for debugging purposes
# run Elmer solver. Default: True
run_solver = True
# run Gmsh and create mesh. Default: True
run_mesher = True
# run Gmsh GUI in order to view and verify the mesh manually before Elmer simulation. Default: False
run_gmsh_gui = False

# Parameter Definition
w = 6e-3  # pad width
l = 6e-3  # pad length
r = 2.95e-3  # pad fillet radius
t = 0.035e-3  # metal thickness
tol = 1e-6  # search tol
hsub = 1.524e-3  # RO4003C substrate height
hmb = 0.1e-3  # bottom metal height
wsub = 4 * max([w, l])  # substrate width
h_ab = 5 * (hsub + hmb + t)  # air box height
w_ab = 1.2 * wsub  # air box width

# set up working subdirectory
sim_dir = "./simdata/"

if not os.path.exists(sim_dir):
    os.mkdir(sim_dir)

###########################################################################
### Some useful functions
###########################################################################


def getMyEntitiesInBoundingBox(xmin, ymin, zmin, xmax, ymax, zmax, dim):
    entities = gmsh.model.getEntitiesInBoundingBox(
        xmin, ymin, zmin, xmax, ymax, zmax, dim
    )
    all_entities = []
    for item in entities:
        all_entities.append(item[1])
    return all_entities


# scan err, war, stats and results
def extract_results_logfile(sim_dir):
    """Scan log file for errors and warnings.

    Args:
        sim_dir (str): Simulation directory

    Returns:
        list[str], list[str], dict: error messages, warnings, statistics
    """
    with open(sim_dir + "/elmersolver.log", "r") as f:
        log = f.readlines()
    for i in range(len(log)):
        log[i] = log[i][:-1]
    for line in log:
```

```python
        if "StatElecSolve:  Capacitance" in line:  # extract capacitance from log file
            s = " ".join(line.split()).split(" ")
            capacitance = float(s[3])
        if (
            "StatElecSolve:  Relative Change" in line
        ):  # extract relative change from log file
            s = " ".join(line.split()).split(" ")
            rel_change = float(s[4])
    return capacitance, rel_change


# engineering format
def powerise10(x):
    """Returns x as a*10**b with 0 <= a < 10"""
    if x == 0:
        return 0, 0
    Neg = x < 0
    if Neg:
        x = -x
    a = 1.0 * x / 10 ** (floor(log10(x)))
    b = int(floor(log10(x)))
    if Neg:
        a = -a
    return a, b


def eng(x):
    """Return a string representing x in an engineer friendly notation"""
    a, b = powerise10(x)
    if -3 < b < 3:
        return "%.4g" % x
    a = a * 10 ** (b % 3)
    b = b - b % 3
    return "%.4gE%s" % (a, b)


############################################################################
### Geometry modeling using gmsh
############################################################################

gmsh.initialize()

gmsh.model.add("3d_electrostatic_capacitance")
geom = gmsh.model.occ

# top metal
m1 = geom.addBox(-w / 2, -l / 2, 0, w, l, t)
geom.fillet([m1], [1, 3, 5, 7], [r], True)

# substrate
sub = geom.addBox(-wsub / 2, -wsub / 2, -hsub, wsub, wsub, hsub)

# bottom metal
m2 = geom.addBox(-wsub / 2, -wsub / 2, -(hsub + hmb), wsub, wsub, hmb)

# airbox: we mesh only air and substrate, but not metal
# design proper height of air box to cover fringing fields
ab = geom.addBox(-w_ab / 2, -w_ab / 2, -(hsub + 2 * hmb), w_ab, w_ab, h_ab)

# remove metal volumes
geom.cut([(3, ab)], [(3, m1), (3, m2)], removeObject=True, removeTool=True)

geom.synchronize()
geom.fragment([(3, sub)], [(3, ab)])

############################################################
# structured meshing would be advantageous but needs further optimization

# ## M1
# NN = 4
```

```
# tf_lines  = [2,13,21,23,24,22,15,3]
# for k in tf_lines:
#     gmsh.model.mesh.setTransfiniteCurve(k, NN)

# ## Sub
# NN = int((w-2*r)*10*1e3)
# tf_lines = [7,16,8,17]
# for k in tf_lines:
#     gmsh.model.mesh.setTransfiniteCurve(k, NN)

# ## MB
# NN = int((l-2*r)*10*1e3)
# tf_lines = [1,4,11,20]
# for k in tf_lines:
#     gmsh.model.mesh.setTransfiniteCurve(k, NN)

# NN = 5
# tf_lines = [42,45,47,41,50,53,55,49]
# for k in tf_lines:
#     gmsh.model.mesh.setTransfiniteCurve(k, NN)

# NN = int(wsub*50*1e3)
# tf_lines = [38,44,52,37,43,51,39,46,54,40,48,56]
# for k in tf_lines:
#     gmsh.model.mesh.setTransfiniteCurve(k, NN)
##########################################################

############################################################################
### Physical Groups and Boundary Conditions
############################################################################
geom.synchronize()

# these two were identified manually in Gmsh GUI
# volumes = gmsh.model.getEntities(dim=3) # check volume numbers after fragment
ph_sub = add_physical_group(3, [2], "substrate")
ph_ab = add_physical_group(3, [3], "airbox")

# the others are identified using own function 'getMyEntitiesInBoundingBox
m1_sfs = getMyEntitiesInBoundingBox(
    -w / 2 - tol, -l / 2 - tol, 0 - tol, w / 2 + tol, l / 2 + tol, t + tol, 2
)
ph_m1_sfs = add_physical_group(2, m1_sfs, "metal1")

m2_sfs = getMyEntitiesInBoundingBox(
    -wsub / 2 - tol,
    -wsub / 2 - tol,
    -(hsub + hmb) - tol,
    wsub / 2 + tol,
    wsub / 2 + tol,
    -hsub + tol,
    2,
)
ph_m2_sfs = add_physical_group(2, m2_sfs, "metal2")

sub_sfs = getMyEntitiesInBoundingBox(
    -wsub / 2 - tol,
    -wsub / 2 - tol,
    -hsub - tol,
    wsub / 2 + tol,
    wsub / 2 + tol,
    tol,
    2,
)

temp_sfs = getMyEntitiesInBoundingBox(
    -w_ab / 2 - tol,
    -w_ab / 2 - tol,
    -h_ab / 2 - tol,
    w_ab / 2 + tol,
    w_ab / 2 + tol,
```

```python
    h_ab / 2 + tol,
    2,
)
ab_sfs = [x for x in temp_sfs if x not in (m1_sfs + sub_sfs + m2_sfs)]
ph_ab_sfs = add_physical_group(2, ab_sfs, "ab_sfs")

###########################################################################
### Meshing
###########################################################################

# We can activate the calculation of mesh element sizes based on curvature
# (here with a target of 90 elements per 2*Pi radians):
gmsh.option.setNumber("Mesh.MeshSizeFromCurvature", 90)

# Finally we apply an elliptic smoother to the grid to have a more regular
# mesh:
gmsh.option.setNumber("Mesh.Smoothing", 10)
gmsh.option.setNumber("Mesh.Algorithm3D", 10)  # faster
# gmsh.option.setNumber('General.NumThreads', 8)
# gmsh.option.setNumber("Mesh.MeshSizeMin", 0.1)
gmsh.option.setNumber("Mesh.MeshSizeMax", 0.2e-3)

if run_mesher:
    geom.synchronize()
    gmsh.model.mesh.generate(dim=3)
    gmsh.write(sim_dir + "/3d_electrostatic_capacitance.msh")
    # Preview mesh.
if run_gmsh_gui:
    gmsh.fltk.run()

# Clear mesh and close gmsh API.
gmsh.clear()
gmsh.finalize()

###########################################################################
### Elmer Setup
###########################################################################

sim = elmer.load_simulation("3D_steady", "my_simulations.yml")
# adding constants is very important, otherwise the solver calculates wrong results!
sim.constants.update({"Permittivity of Vacuum": "8.8542e-12"})
sim.constants.update({"Gravity(4)": "0 -1 0 9.82"})
sim.constants.update({"Boltzmann Constant": "1.3807e-23"})
sim.constants.update({"Unit Charge": "1.602e-19"})

# materials
air = elmer.load_material("air", sim, "my_materials.yml")
ro4003c = elmer.load_material("ro4003c", sim, "my_materials.yml")

# solver
solver_electrostatic = elmer.load_solver("Electrostatics", sim, r"my_solvers.yml")
# very important, the value must match the boundary condition abs(potential difference)
    !!!
# otherwise the capacitance will be calculated wrong !
solver_electrostatic.data.update({"Potential Difference": "1.0"})

# equation
eqn = elmer.Equation(sim, "main", [solver_electrostatic])

# bodies
bdy_sub = elmer.Body(sim, "substrate", [ph_sub])
bdy_sub.material = ro4003c
bdy_sub.equation = eqn

bdy_ab = elmer.Body(sim, "airbox", [ph_ab])
bdy_ab.material = air
bdy_ab.equation = eqn

# boundaries
bndry_m1 = elmer.Boundary(sim, "top metal", [ph_m1_sfs])
```

```python
bndry_m1.data.update({"Potential": "1.0"})

bndry_m2 = elmer.Boundary(sim, "bottom metal", [ph_m2_sfs])
bndry_m2.data.update({"Potential": "0.0"})

bndry_airbox = elmer.Boundary(sim, "FarField", [ph_ab_sfs])
bndry_airbox.data.update({"Electric Infinity BC": "True"})

# export
sim.write_startinfo(sim_dir)
sim.write_sif(sim_dir)

if run_mesher:
    execute.run_elmer_grid(sim_dir, "3d_electrostatic_capacitance.msh")

##############
# execute ElmerGrid & ElmerSolver
if run_solver:
    execute.run_elmer_solver(sim_dir)
    ###############
    # scan log for errors and warnings
    err, warn, stats = post.scan_logfile(sim_dir)
    capacitance, rel_change = extract_results_logfile(sim_dir)
    print("## RESULTS BEGIN ##############################")
    print("Errors:", err)
    print("Warnings:", warn)
    print("Statistics:", stats)
    print("Relative Change:", f"{rel_change:.2E}")
    print("##############################################")
    print("w:", eng(w))
    print("l:", eng(l))
    print("r:", eng(r))
    print("Capacitance:", eng(capacitance))
    print("##############################################")
```

4.2 Elmer FEM Solver Input File (SIF)

This Elmer FEM solver input file (SIF) is created automatically and listed here just for review purpose:

Listing 2: Elmer FEM solver input file (SIF)

```
Header
  CHECK KEYWORDS "Warn"
  Mesh DB "." "."
End

Simulation
  Max Output Level = 5
  Coordinate System = Cartesian
  Coordinate Mapping(3) = 1 2 3
  Simulation Type = Steady state
  Steady State Max Iterations = 1
  Output Intervals = 1
  Timestepping Method = BDF
  BDF Order = 1
  Solver Input File = case.sif
  Post File = case.vtu
  Output File = case.result
End

Constants
  Stefan Boltzmann = 5.6704e-08
  Permittivity of Vacuum = 8.8542e-12
  Gravity(4) = 0 -1 0 9.82
  Boltzmann Constant = 1.3807e-23
  Unit Charge = 1.602e-19
End

! main
Equation 1
  Active Solvers(1) = 1   ! Electrostatics,
End


! Electrostatics
Solver 1
  Equation = Electrostatics
  Calculate Electric Field = True
  Procedure = "StatElecSolve" "StatElecSolver"
  Variable = Potential
  Calculate Electric Energy = True
  Exec Solver = Always
  Stabilize = True
  Bubbles = False
  Lumped Mass Matrix = False
  Optimize Bandwidth = True
  Steady State Convergence Tolerance = 1e-05
  Nonlinear System Convergence Tolerance = 1e-07
  Nonlinear System Max Iterations = 20
  Nonlinear System Newton After Iterations = 3
  Nonlinear System Newton After Tolerance = 0.001
  Nonlinear System Relaxation Factor = 1
  Linear System Solver = Iterative
  Linear System Iterative Method = BiCGStab
  Linear System Max Iterations = 500
  Linear System Convergence Tolerance = 1e-10
  BiCGstabl polynomial degree = 2
  Linear System Preconditioning = ILU0
  Linear System ILUT Tolerance = 0.001
  Linear System Abort Not Converged = False
  Linear System Residual Output = 10
  Linear System Precondition Recompute = 1
  Potential Difference = 1.0
End
```

```
! air
Material 1
  Density = 1.1885
  Electric Conductivity = 0.0
  Heat Capacity = 1006.4
  Heat Conductivity = 0.025873
  Relative Permeability = 1
  Relative Permittivity = 1
End

! ro4003c
Material 2
  Density = 1790
  Relative Permeability = 1
  Relative Permittivity = 3.55
End


! substrate
Body 1
  Target Bodies(1) = 1
  Equation = 1  ! main
  Material = 2  ! ro4003c
End

! airbox
Body 2
  Target Bodies(1) = 2
  Equation = 1  ! main
  Material = 1  ! air
End


! top metal
Boundary Condition 1
  Target Boundaries(1) = 3
  Potential = 1.0
End

! bottom metal
Boundary Condition 2
  Target Boundaries(1) = 4
  Potential = 0.0
End

! FarField
Boundary Condition 3
  Target Boundaries(1) = 5
  Electric Infinity BC = True
End
```

5 Conclusions

This example may help the reader to set up a Elmer FEM computation of a 3D electrostatic capacitance problem using pyelmer. The Elmer FEM computation result agrees very well with published data in the literature. However, the calculated result depend on the mesh structure and mesh element size. Mesh generation is by far not perfect and not solved yet in this example. Gmsh API functions can be used to ensure small mesh size at regions of high field gradients (edges, corners) and to optimize the mesh for lower FEM computation time. The authors welcome any comments for further improvements.

References

[1] "Elmer FEM open source multiphysical simulation software." (2021), [Online]. Available: http://www.elmerfem.org/blog/.

[2] C. Geuzaine and J.-F. Remacle. "Gmsh – A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities." (2021), [Online]. Available: https://gmsh.info/.

[3] "WinPython – Portable Scientific Python 2/3 32/64bit Distribution for Windows." (2021), [Online]. Available: https://sourceforge.net/projects/winpython/files/.

[4] A. Enders-Seidlitz, A. Kunwar, and K. Dadzis, *Pyelmer - a python interface to elmer fem*, Apr. 2021. DOI: 10.5281/zenodo.4431440. [Online]. Available: https://github.com/nemocrys/pyelmer.

[5] "ParaView – An open-source, multi-platform data analysis and visualization application." (2021), [Online]. Available: https://www.paraview.org/.

[6] W. C. Chew and J. A. Kong, "Effects of fringing fields on the capacitance of circular microstrip disk," *IEEE Transactions on Microwave Theory and Techniques*, vol. 28, no. 2, pp. 98–104, 1980. DOI: 10.1109/TMTT.1980.1130017.

[7] "RO4000 Series High Frequency Circuit Materials." (2021), [Online]. Available: https://rogerscorp.com/-/media/project/rogerscorp/documents/advanced-electronics-solutions/english/data-sheets/ro4000-laminates-ro4003c-and-ro4350b---data-sheet.pdf.

[8] C. Geuzaine. "The Gmsh Application Programming Interface (API)." (2021), [Online]. Available: https://gitlab.onelab.info/gmsh/gmsh/blob/gmsh_4_8_4/api/gmsh.py.