

pyelmer Example: 3D Electrostatic Capacitance

Werner Simbürger
werner.simbuenger@hppi.de

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

July 29, 2021

Contents

1	Problem Description	2
2	Solution	2
2.1	Analytical Calculation	2
2.2	Elmer FEM Calculation	3
3	Test Case and Results	3
3.1	Test Case Parameter	3
3.2	Capacitance Calculation Results	3
3.3	Discussion of the Elmer FEM Solution and Boundary Conditions	4
4	Source File Listings	5
4.1	Python (pyelmer) Script	5
4.2	Elmer FEM Solver Input File (SIF)	11
5	Conclusions	12
	References	13

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.

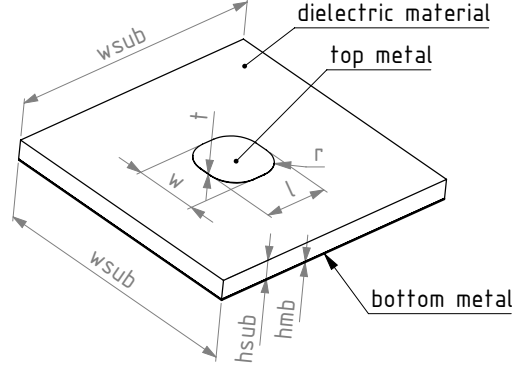


Figure 1: 3D electrostatics capacitance

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 h_{sub} , width and length w_{sub} . The bottom metal layer has a layer thickness h_{mb} . 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 $w = l$, $r \rightarrow w/2 = l/2$ and $t = h_{mb} \ll h_{sub}$, which results in a circular microstrip disk, as shown in Fig. 2 and presented in the paper [6].

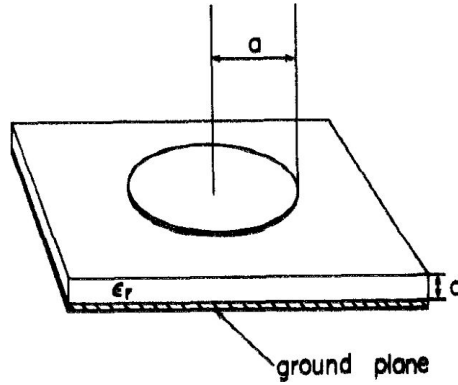


Figure 2: Effects of fringing fields on the capacitance of a circular microstrip disk [6]

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:

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

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

2.2 Elmer FEM Calculation

Prerequisite is the installation of the following software packages¹:

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)
h_{sub}	1.524	mm	dielectric material layer thickness
w_{sub}	24	mm	dielectric material layer width
ϵ_r	3.55		dielectric material layer relative permittivity (Rogers RO4003C material [7])
h_{mb}	0.1	mm	bottom metal layer thickness

Table 1: Test case parameter

3.2 Capacitance Calculation Results

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

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
#####
```

¹available 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.

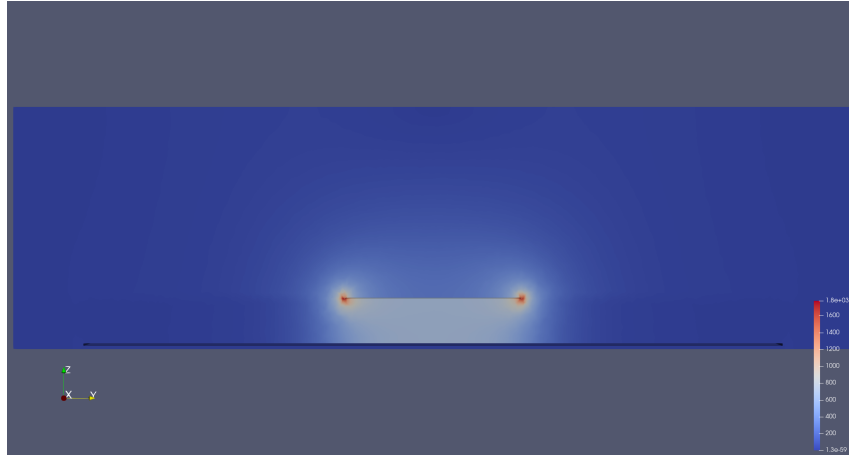


Figure 3: Calculated electric field magnitude (cross section of Fig. 4)

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.

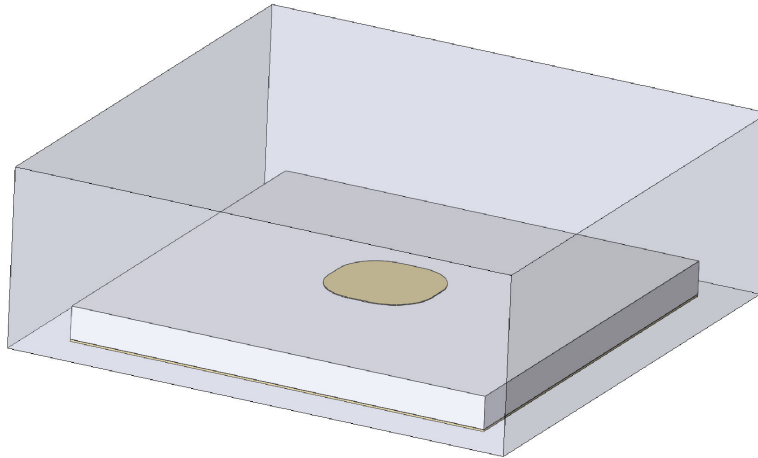


Figure 4: 3D model including air box

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.

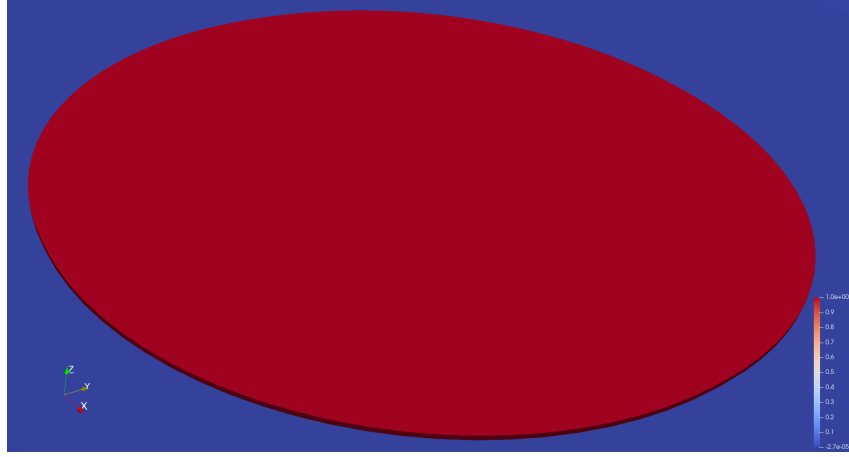


Figure 5: Potential distribution (top metal zoom view)

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»

```

1  # define options which module to execute
2
3
4  # The following switches defined are for debugging purposes
5  # run Elmer solver. Default: True
6  run_solver = True
7  # run Gmsh and create mesh. Default: True
8  run_mesher = True
9  # run Gmsh GUI in order to view and verify the mesh manually before Elmer simulation.
   # Default: False
10 run_gmsh_gui = False
11
12 import os
13 import gmsh
14 from pyelmer import elmer, post
15 from pyelmer import execute
16 from pyelmer.gmsh import add_physical_group, get_boundaries_in_box
17 import subprocess
18 import shutil
19 from math import floor, log10
20
21
22 def getMyEntitiesInBoundingBox(xmin, ymin, zmin, xmax, ymax, zmax, dim):
23     entities = gmsh.model.getEntitiesInBoundingBox(xmin, ymin, zmin, xmax, ymax, zmax,
24     dim)
25     all_entities = []
26     for item in entities:
27         all_entities.append(item[1])
28     return all_entities
29
30 # scan err, war, sats and results
31 def extract_results_logfile(sim_dir):
32     """Scan log file for errors and warnings.
33
34     Args:
35         sim_dir (str): Simulation directory
36
37     Returns:
38         list[str], list[str], dict: error messages, warnings, statistics
39     """
40     with open(sim_dir + "/elmersolver.log", "r") as f:
41         log = f.readlines()
42         for i in range(len(log)):
43             log[i] = log[i][: -1]
44         for line in log:
45             if "StatElecSolve: Capacitance" in line: # extract capacitance from log file
46                 s = " ".join(line.split()).split(" ")
47                 capacitance = float(s[3])
48             if "StatElecSolve: Relative Change" in line: # extract relative change from log
49                 file
50                 s = " ".join(line.split()).split(" ")
51                 rel_change = float(s[4])
52         return capacitance, rel_change
53
54 # engineering format
55 def powerise10(x):
56     """ Returns x as a*10**b with 0 <= a < 10
57
58     """
59     if x == 0: return 0,0
60     Neg = x < 0
61     if Neg: x = -x
62     a = 1.0 * x / 10**(floor(log10(x)))
63     b = int(floor(log10(x)))
64     if Neg: a = -a
65     return a,b
66
67 def eng(x):
68     """Return a string representing x in an engineer friendly notation"""

```

```

67     a,b = powerise10(x)
68     if -3 < b < 3: return "%.4g" % x
69     a = a * 10**(b % 3)
70     b = b - b % 3
71     return "%.4gE%s" % (a,b)
72 #####
73
74 # set up working subdirectory
75 sim_dir = '3d_electrostatic_capacitance_update'
76
77 if not os.path.exists(sim_dir):
78     os.mkdir(sim_dir)
79
80 #####
81 # geometry modeling using gmsh
82
83 gmsh.initialize()
84
85 gmsh.model.add("3d_electrostatic_capacitance")
86 geom = gmsh.model.occc
87
88 # Parameter Definition
89 # pad width
90 w = 6e-3
91 # pad length
92 l = 6e-3
93 # pad fillet radius
94 r = 2.95e-3
95 # metal thickness
96 t = 0.035e-3
97 # search tol
98 tol = 1e-6
99
100
101 # R04003C substrate height
102 hsub = 1.524e-3
103 hmb = 0.1e-3
104
105 # FEM: define substrate width 5*max(w, l)
106 wsub = 4 * max([w, l])
107
108 # top metal
109 m1 = geom.addBox(-w/2, -l/2, 0, w, l, t)
110 geom.fillet([m1], [1, 3, 5, 7], [r], True)
111
112
113 # substrate
114 sub = geom.addBox(-wsub/2, -wsub/2, -hsub, wsub, wsub, hsub)
115
116
117 # bottom metal
118 m2 = geom.addBox(-wsub/2, -wsub/2, -(hsub+hmb), wsub, wsub, hmb)
119
120
121 # airbox: we mesh only air and substrate, but not metal
122 # design proper height of air box to cover fringing fields
123 h_ab = 5 * (hsub + hmb + t)
124 w_ab = 1.2 * wsub
125 ab = geom.addBox(-w_ab/2, -w_ab/2, -(hsub+2*hmb), w_ab, w_ab, h_ab)
126
127 # remove metal volumes
128 geom.cut([(3, ab)], [(3, m1), (3, m2)], removeObject=True, removeTool=True)
129
130
131 geom.synchronize()
132 # https://gitlab.onelab.info/gmsh/gmsh/blob/gmsh_4_8_4/demos/api/bspline_bezier_patches.py#L70
133 # Method 3
134 #geom.removeAllDuplicates()
135

```



```

136 geom.fragment([(3,sub)],[(3,ab)])
137
138
139 # transfinite = True
140
141 # ## M1
142 # NN = 4
143 # tf_lines = [2,13,21,23,24,22,15,3]
144 # for k in tf_lines:
145 #     gmsh.model.mesh.setTransfiniteCurve(k, NN)
146
147 # ## Sub
148 # NN = int((w-2*r)*10*1e3)
149 # tf_lines = [7,16,8,17]
150 # for k in tf_lines:
151 #     gmsh.model.mesh.setTransfiniteCurve(k, NN)
152
153
154 # ## MB
155 # NN = int((l-2*r)*10*1e3)
156 # tf_lines = [1,4,11,20]
157 # for k in tf_lines:
158 #     gmsh.model.mesh.setTransfiniteCurve(k, NN)
159
160 # NN = 5
161 # tf_lines = [42,45,47,41,50,53,55,49]
162 # for k in tf_lines:
163 #     gmsh.model.mesh.setTransfiniteCurve(k, NN)
164
165 # NN = int(wsub*50*1e3)
166 # tf_lines = [38,44,52,37,43,51,39,46,54,40,48,56]
167 # for k in tf_lines:
168 #     gmsh.model.mesh.setTransfiniteCurve(k, NN)
169
170
171 #geom.synchronize()
172
173
174 # https://gitlab.onelab.info/gmsh/gmsh/-/issues/1219
175 # avoid non conformal gemetry
176 #geom.fragment([(3,2), (3,4)], [])
177 #geom.fragment(volumes, [])
178 #geom.fragment([(3,4)],[(3,2)])
179
180
181
182 #####
183 ### Physical Groups and Boundary Conditions
184 #####
185 geom.synchronize()
186
187 # serach volumes for defining physical groups
188 #volumes = gmsh.model.getEntities(dim=3)
189 # for v in volumes:
190 #     gmsh.model.addPhysicalGroup(3, [v[1]])
191
192 #volumes = gmsh.model.getEntities(dim=3) # check volume numbers after fragment
193 ph_sub = add_physical_group(3,[2], 'substrate')
194 ph_ab = add_physical_group(3,[3], 'airbox')
195
196
197 # https://gitlab.onelab.info/gmsh/gmsh/-/issues/1219
198 # avoid non conformal gemetry
199 #geom.fragment([(3,2), (3,4)], [])
200 #geom.fragment(volumes, [])
201 #geom.fragment([(3,4)],[(3,2)])
202 #geom.removeAllDuplicates()
203 #geom.synchronize()
204

```

```

205 m1_sfs = getMyEntitiesInBoundingBox(-w/2-tol, -l/2-tol, 0-tol, w/2+tol, l/2+tol, t+tol,
2)
206 ph_m1_sfs = add_physical_group(2, m1_sfs, "metal1")
207
208 m2_sfs = getMyEntitiesInBoundingBox(-wsub/2-tol, -wsub/2-tol, -(hsub+hmb)-tol, wsub/2+
tol, wsub/2+tol, -hsub+tol, 2)
209 ph_m2_sfs = add_physical_group(2, m2_sfs, "metal2")
210
211 sub_sfs = getMyEntitiesInBoundingBox(-wsub/2-tol, -wsub/2-tol, -hsub-tol, wsub/2+tol,
wsub/2+tol, tol, 2)
212
213 m2_sfs = getMyEntitiesInBoundingBox(-wsub/2-tol, -wsub/2-tol, -(hsub+hmb)-tol, wsub/2+
tol, wsub/2+tol, -hsub+tol, 2)
214
215 temp_sfs = getMyEntitiesInBoundingBox(-w_ab/2-tol, -w_ab/2-tol, -h_ab/2-tol, w_ab/2+tol,
w_ab/2+tol, h_ab/2+tol, 2)
216 ab_sfs = [x for x in temp_sfs if x not in (m1_sfs + sub_sfs + m2_sfs)]
217 ph_ab_sfs = add_physical_group(2, ab_sfs, "ab_sfs")
218
219
220 #geom.synchronize()
221
222 # We can activate the calculation of mesh element sizes based on curvature
223 # (here with a target of 90 elements per 2*Pi radians):
224 gmsh.option.setNumber("Mesh.MeshSizeFromCurvature", 90)
225
226
227 # Finally we apply an elliptic smoother to the grid to have a more regular
228 # mesh:
229 gmsh.option.setNumber("Mesh.Smoothing", 10)
230
231 gmsh.option.setNumber('Mesh.Algorithm3D', 10) # faster
232 #gmsh.option.setNumber('General.NumThreads', 8)
233
234 #gmsh.option.setNumber("Mesh.MeshSizeMin", 0.1)
235 gmsh.option.setNumber("Mesh.MeshSizeMax", 0.2e-3)
236
237
238 if run_mesher:
239     geom.synchronize()
240     gmsh.model.mesh.generate(dim=3)
241     gmsh.write(sim_dir + "/3d_electrostatic_capacitance.msh")
242     # Preview mesh.
243 if run_gmsh_gui:
244     gmsh.fltk.run()
245
246 # Clear mesh and close gmsh API.
247 gmsh.clear()
248 gmsh.finalize()
249
250 #####
251 # elmer setup
252
253 sim = elmer.load_simulation('3D_steady', 'my_simulations.yml')
254 # adding constants is very important, otherwise the solver calculates wrong results!
255 sim.constants.update({"Permittivity of Vacuum": "8.8542e-12"})
256 sim.constants.update({"Gravity(4)": "0 -1 0 9.82"})
257 sim.constants.update({"Boltzmann Constant": "1.3807e-23"})
258 sim.constants.update({"Unit Charge": "1.602e-19"})
259
260 # materials
261 air = elmer.load_material('air', sim, 'my_materials.yml')
262 ro4003c = elmer.load_material('ro4003c', sim, 'my_materials.yml')
263 #ideal = elmer.load_material('ideal', sim, r'E:\PROJECT\TLP_3010C_R01\PYTHON\Elmer_FEM\
own_definitions\materials.yml')
264
265 # solver
266 solver_electrostatic = elmer.load_solver('Electrostatics', sim, r'my_solvers.yml')
267 # very important, the value must match the boundary condition abs(potential difference)
!!!

```

```

268 # otherwise the capacitance will be calculated wrong !
269 solver_electrostatic.data.update({"Potential Difference": "1.0"})
270
271 # equation
272 eqn = elmer.Equation(sim, "main", [solver_electrostatic])
273 #eqn.data.update({"Electric Field": "Computed"})
274
275
276 # bodies
277
278 # bdy_m1 = elmer.Body(sim, "top metal", [m1])
279 # bdy_m1.material = ideal
280 # bdy_m1.equation = eqn
281
282 bdy_sub = elmer.Body(sim, "substrate", [ph_sub])
283 bdy_sub.material = ro4003c
284 bdy_sub.equation = eqn
285
286 # bdy_m2 = elmer.Body(sim, "bottom metal", [m2])
287 # bdy_m2.material = ideal
288 # bdy_m2.equation = eqn
289
290 bdy_ab = elmer.Body(sim, "airbox", [ph_ab])
291 bdy_ab.material = air
292 bdy_ab.equation = eqn
293
294 # boundaries
295 bndry_m1 = elmer.Boundary(sim, "top metal", [ph_m1_sfs])
296 bndry_m1.data.update({"Potential": "1.0"})
297
298 bndry_m2 = elmer.Boundary(sim, "bottom metal", [ph_m2_sfs])
299 bndry_m2.data.update({"Potential": "0.0"})
300
301 bndry_airbox = elmer.Boundary(sim, "FarField", [ph_ab_sfs])
302 bndry_airbox.data.update({"Electric Infinity BC": "True"})
303
304 # export
305 sim.write_startinfo(sim_dir)
306 sim.write_sif(sim_dir)
307
308 if run_mesher:
309     execute.run_elmer_grid(sim_dir, '3d_electrostatic_capacitance.msh')
310
311 #####
312 # execute ElmerGrid & ElmerSolver
313 if run_solver:
314     execute.run_elmer_solver(sim_dir)
315     #####
316     # scan log for errors and warnings
317     err, warn, stats = post.scan_logfile(sim_dir)
318     capacitance, rel_change = extract_results_logfile(sim_dir)
319     print('## RESULTS BEGIN #####')
320     print('Errors:', err)
321     print('Warnings:', warn)
322     print('Statistics:', stats)
323     print('Relative Change:', f'{rel_change:.2E}')
324     print('#####')
325     print('w:', eng(w))
326     print('l:', eng(l))
327     print('r:', eng(r))
328     print('Capacitance:', eng(capacitance))
329     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)

```
1 Header
2   CHECK KEYWORDS "Warn"
3   Mesh DB "." "."
4 End
5
6 Simulation
7   Max Output Level = 5
8   Coordinate System = Cartesian
9   Coordinate Mapping(3) = 1 2 3
10  Simulation Type = Steady state
11  Steady State Max Iterations = 1
12  Output Intervals = 1
13  Timestepping Method = BDF
14  BDF Order = 1
15  Solver Input File = case.sif
16  Post File = case.vtu
17  Output File = case.result
18 End
19
20 Constants
21   Stefan Boltzmann = 5.6704e-08
22   Permittivity of Vacuum = 8.8542e-12
23   Gravity(4) = 0 -1 0 9.82
24   Boltzmann Constant = 1.3807e-23
25   Unit Charge = 1.602e-19
26 End
27
28 ! main
29 Equation 1
30   Active Solvers(1) = 1   ! Electrostatics,
31 End
32
33
34 ! Electrostatics
35 Solver 1
36   Equation = Electrostatics
37   Calculate Electric Field = True
38   Procedure = "StatElecSolve" "StatElecSolver"
39   Variable = Potential
40   Calculate Electric Energy = True
41   Exec Solver = Always
42   Stabilize = True
43   Bubbles = False
44   Lumped Mass Matrix = False
45   Optimize Bandwidth = True
46   Steady State Convergence Tolerance = 1e-05
47   Nonlinear System Convergence Tolerance = 1e-07
48   Nonlinear System Max Iterations = 20
49   Nonlinear System Newton After Iterations = 3
50   Nonlinear System Newton After Tolerance = 0.001
51   Nonlinear System Relaxation Factor = 1
52   Linear System Solver = Iterative
53   Linear System Iterative Method = BiCGStab
54   Linear System Max Iterations = 500
55   Linear System Convergence Tolerance = 1e-10
56   BiCGStabl polynomial degree = 2
57   Linear System Preconditioning = ILU0
58   Linear System ILUT Tolerance = 0.001
59   Linear System Abort Not Converged = False
60   Linear System Residual Output = 10
61   Linear System Precondition Recompute = 1
62   Potential Difference = 1.0
63 End
64
65
```

```

66 ! air
67 Material 1
68   Density = 1.1885
69   Electric Conductivity = 0.0
70   Heat Capacity = 1006.4
71   Heat Conductivity = 0.025873
72   Relative Permeability = 1
73   Relative Permittivity = 1
74 End
75
76 ! ro4003c
77 Material 2
78   Density = 1790
79   Relative Permeability = 1
80   Relative Permittivity = 3.55
81 End
82
83
84 ! substrate
85 Body 1
86   Target Bodies(1) = 1
87   Equation = 1 ! main
88   Material = 2 ! ro4003c
89 End
90
91 ! airbox
92 Body 2
93   Target Bodies(1) = 2
94   Equation = 1 ! main
95   Material = 1 ! air
96 End
97
98
99 ! top metal
100 Boundary Condition 1
101   Target Boundaries(1) = 3
102   Potential = 1.0
103 End
104
105 ! bottom metal
106 Boundary Condition 2
107   Target Boundaries(1) = 4
108   Potential = 0.0
109 End
110
111 ! FarField
112 Boundary Condition 3
113   Target Boundaries(1) = 5
114   Electric Infinity BC = True
115 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 and A. Kunwar. “pyelmer – a simple object-oriented way to set up Elmer FEM simulations from python.” (2021), [Online]. Available: <https://pypi.org/project/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](https://doi.org/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.