Working Notes

July 6, 2020

MEMS:testingclass.py

- two ways to get fields: solver.getselfe(), solver.getphi() and wp.getselfe(), wp.getselfe(). So far seem identical.

- when plotting in python, getting fields will give the values at endpoints on grid giving odd zero-contours. Be aware of this since gist graphics won’t show the same.

- so far python graphics haven’t matched with warp graphics. This is concerning since I’m pulling fields straight from warp. Example graphics below. Setup is as follows:

Two washers were created with 10 KV and -10 KV settings. Dimensions:

rmin=1mm

rmax=1.5mm

length= 2mm

z-center for washer 1 = 3mm

z-center for washer 2 = 8mm

Mesh details:

- x,y-extent = [-4mm, 4mm] with nx,ny = 100, 110. The different settings helps when slicing potential and electric fields arrays to be sure indexing is right.

- z-extent = [0m, 12mm] with nz =250

- solvergeom = XYZgeom

- boundary conditions = Dirichlet

- solver = MRBlock3D

Plots:

A close up of a logo

Description automatically generated

A close up of a logo

Description automatically generatedA close up of a logo

Description automatically generated

for the warp plot, green lines are Ex and blue correspond to the potential.

The red lines on the python plots are (left Ex, right potential) are the zero contours. How to do this is useful to right down.

As an example

fig, ax = plt.subplots()

cp = ax.contour()

cp is an object with a lot in it. The levels can be found by using

cp.levels

To find the zero contour np.where can be used. This returns a list of arrays and will require a double index for grabbing one index. For example

#--Find zero index

zeroindex = np.where(cp.levels == 0) #will be a list of arrays where this is true

zeroindex = zeroindex[0][0] #gives integer value of where the contour is zero

#--Set zero contour color and linestyle using collections

cp.collections[zeroindex].set\_linestyle(‘dashed’)

cp.collections[zeroindex].set\_color(‘r’)

Keep in mind this doesn’t change the color bar label. I’ll have to figure this out.

The warp plot is able to fill the conductor object with a color for visualization. Maybe there is a way to pull this data from warp and integrate with python.

--Update

Indexing!!! I was working in r-z geometry (a side effect of indexing with ORISS arrays for so long). The previous code was using wp.w3d.ymesh[1] thinking this was one grid cell above 0. This was actually one grid cell above -4mm. Thanks for Steve to pointing this out in about 2 minutes.

The code has been updated to find the 0 index as follows

#--Find index where y=0

#check existence

if (0 in wp.w3d.ymesh):

yindex = np.where(wp.w3d.ymesh==0)[0][0]

#if not find closest positive grid point

else:

yindex = np.where(wp.w3d.ymesh>0)[0][0]

With this addition the following python plots are produced: A close up of a logo

Description automatically generatedA close up of a logo

Description automatically generated

July 7, 2020

- Drawxy() is not specified for conductors resulting in a “Draw not implemented for \_\_\_” error message. A workout around will be to plot zero contours.

I attempted this first by finding where the electric field in x and y was zero. This made odd plots.

A screenshot of a computer

Description automatically generated

The code to accomplish this:

#--Plot conductors in xy using zero contours

# Find z-index where first conductor lives

x = wp.w3d.xmesh

y = wp.w3d.ymesh

z = wp.w3d.zmesh

zindexrange = np.where((z>2.5\*mm) & (z<3.5\*mm))

zindex = zindexrange[0][int(len(zindexrange)/2)] #take center point of range

Ex = wp.getselfe(comp='x')[1:-1, 1:-1, zindex]

Ey = wp.getselfe(comp='y')[1:-1, 1:-1, zindex]

E = Ex + Ey

X,Y = np.meshgrid(y[1:-1], x[1:-1])

fig, ax = plt.subplots()

ax.set\_xlabel('x [mm]')

ax.set\_ylabel('y [mm]')

ax.set\_title(r'$E\_x + E\_y$')

ax.set\_xlim(-5, 5)

ax.set\_ylim(-5, 5)

contx = ax.contourf(X/mm, Y/mm, E, levels=50)

#--Find 0 contour

if (0 in contx.levels):

zeroindex = np.where(contx.levels==0)[0][0]

else:

print("Zero contour doesn't exist")

contx.collections[zeroindex].set\_linestyle('dashed')

contx.collections[zeroindex].set\_color('r')

plt.tight\_layout()

plt.show()

Plotting the potential gives a better result:

A close up of a logo

Description automatically generated

The code to produce this:

V = 10\*kV

#--Plot conductors in xy using zero contours

# Find z-index where first conductor lives

x = wp.w3d.xmesh

y = wp.w3d.ymesh

z = wp.w3d.zmesh

zindexrange = np.where((z>2.5\*mm) & (z<3.5\*mm))

zindex = zindexrange[0][int(len(zindexrange)/2)] #take center point of range

potential = wp.getphi()[1:-1, 1:-1, zindex]

fig, ax = plt.subplots()

ax.set\_xlim(-5,5)

ax.set\_ylim(-5,5)

ax.set\_xlabel('x [mm]')

ax.set\_ylabel('y [mm]')

ax.set\_title(r'$\Phi(x,y)$')

X,Y = np.meshgrid(y[1:-1], x[1:-1])

cont = ax.contourf(X/mm, Y/mm, potential, levels=50)

if (V in cont.levels ):

index = np.where(cont.levels == V)[0][0]

else:

print("V not in levels")

cont.collections[-1].set\_linestyle('dashed')

cont.collections[-1].set\_color('r')

plt.tight\_layout()

plt.show()

Oddly, the index routine gives index=50. When running cont.collections[50] this index is out of range which is why [-1] index is used.

July 10, 2020

Now visualizing the ESQ geometry. The mesh created is as follows:

x, y extents: [-4mm, 4mm]

z extent: [0, 2mm]

ESQ placed at zcenter = 1mm

The ESQ was created from the geometry file (renamed conductor.py in /valverde). The arguments are (position=1mm, InvertPolarity=True, voltage=2\*kV).

Potential fields shown below:

A close up of a logo

Description automatically generatedA close up of a device

Description automatically generated

-First plot code is

X,Y = np.meshgrid(x,y)

phi = wp.getphi()[:, :, zcenterindex]

#--Plot potential in xy at z=center of esq

fig, ax = plt.subplots()

ax.set\_xlabel('x [mm]')

ax.set\_ylabel('y [mm]')

ax.set\_title('Potential of ESQ in x-y')

cont = ax.contour(X/mm, Y/mm, phi, levels=50 )

contcb = fig.colorbar(cont, extend='both', shrink=0.8)

#Set zero contour to red -- line

zerocontour = np.where(cont.levels==0)[0][0]

zerocontour = cont.collections[zerocontour]

zerocontour.set\_color('r')

zerocontour.set\_linestyle('dashed')

zerocontour.set\_linewidth(0.5)

plt.show()

The second plot is basically the same.

The Electric fields Ex and Ez are given below:

A close up of a map

Description automatically generatedA close up of a piece of paper

Description automatically generated

A close up of a map

Description automatically generated

July 13, 2020

Above plots using xy boundary conditions as periodic instead of Neumann.

A close up of a logo

Description automatically generatedA picture containing light

Description automatically generated

A close up of a map

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA close up of a map

Description automatically generated

July 15, 2020

-Beginning to test RFstack setup.

-I’m not set on how these are created. The wafer function within the function uses a zip to iterate through center positions and voltages. However, voltage isnt actually indexed.

for s, v in zip(stackPositions, voltages):

stack = wafer(s[0], 0) + wafer(s[1], v) + wafer(s[2], v) + wafer(s[3], 0)

stacks.append(stack)

return wp.sum(stacks)

This is a bit confusing as to what the inputs for the RFstack should be since it takes in RFstack(positions, voltages). I think this can be altered so that in the class functionality there is a simple time dependent voltage function. This will also eliminate the superfluous voltage listing in the single species code and will be cleaner.

-Another detail. I should check if the stack sperations are constant. For example, if this || represents a gnd and a biased plate, then ||--- || is the --- constant seperation always? This would make the stack routine much easier since all one needs to do is specific two center positions (assuming the gnd and bias separation are also biased).

--Only condition is the calculation of the drift space that needs to be .

July 16, 2020

-Field plots for the wafer stack. Geometry:

x,y extents = [-4mm, 4mm]

z extent = [0, 15mm]

stack is placed with wafer positions [1mm, 4mm, 9mm, 12mm]

bias is set to 3kV on 2nd and 3rd wafer.

-Potential in zx

A picture containing comb, light

Description automatically generated

-Ex and Ez in zx

A close up of a logo

Description automatically generatedA close up of text on a white background

Description automatically generated

-The plots agree with Warp’s output. In order to get an opposite bias on wafers 2 and 3 I had to do a little hack in the conductors.py file. In the last for-loop in RFstack, I made it so that the voltages (‘v’) is indexed. I’m not sure if this inversion is done later in the single species code. I will note it here for future reference.