FDTR Fitting with Python File:

This program opens up two separate data files collected from a Zurich lock-in amplifier (although any lock-in amplifier can be used for this system). The file should be recording frequency and phase lag and the user should be downloading the data as a .txt file. Note that all of the data needs to be in the viewing window for it to be included in the data file. The file, by default, should be tab delimited.

How to use this software

I personally use PyCharm to run this file. Note that Python will need to be installed to run it, in addition to all the import files at the top of the program. Unless otherwise noted, you should not change any of the code not shown in the images below.

Step 1 – Change points used in analysis and change beam sizes

```
12  deleteEnd = 1  # Deletes defined number of initial points from data
13  deleteBeg = 2  # Deletes defined number of end points from data
14
15  w0 = 8.8 * (10 ** (-6))  # Pump Diameter in m
16  w1 = 5.0 * (10 ** (-6))  # Probe Diameter in m
```

- The user can choose to delete some number of points at both the beginning and end of the data set, which is particularly useful if noise cannot be avoided in these regimes.
- The user should measure the beam diameters and input them into the initial w0 and w1 terms

Step 2 – Change guess values in guess vector (x0)

```
22  x0 = np.array([35, 1 * (10 ** 7)], dtype=float)  # Initial guesses for unknown parameters
```

- The values in the guess vector should be changed in the order that the user is fitting for (i.e., if the first parameter you are fitting for is thermal conductivity – more on that below – you should list its guess value first within the guess vector).

Step 3 – Change values in layer vectors

```
31      kr = [220, 0, cos[0]]  # Vector of in-plane thermal conductivities for each layer; even layers a
32      # because they are interfaces.
33      kz = [220, cos[1], cos[0]]  # Vector of through-plane thermal conductivities (odd layers) and th
34      # conductances (even layers)
35      cv = [2.48 * (10 ** 6), 0, 3.03 * (10 ** 6)]  # Vector of volumetric heat capacities for each la
36      t = [81 * (10 ** (-9)), 0, 1]  # Thickness of each layer; even layers are always 0 because they
```

- kr, kz, cv, and t are the in-plane thermal conductivity, through-plane thermal conductivity, volumetric heat capacity ($c_v = c_p \cdot \rho$), and thickness of each layer, respectively.
- This code is only built for samples with transducers, so your first value in each vector (layer 1) should be the transducer layer.
- All even layers are *interfaces*, thus:
  - $kr_{int} = 0$
  - $cv_{int} = 0$
  - $t_{int} = 0$

- kz_{int} = G (thermal boundary conductance); kz will never have a value of 0 in its vector

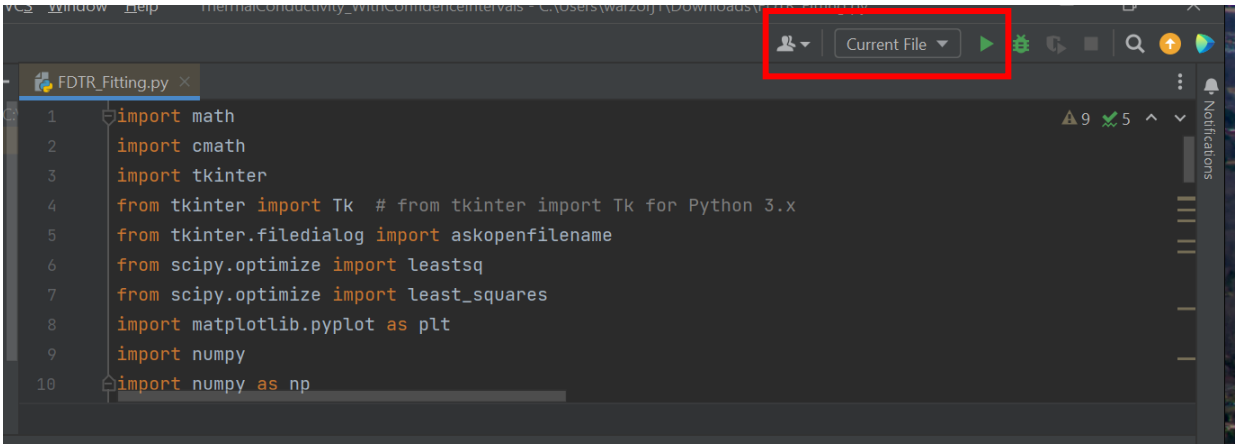
- The values listed in the previous image represent the case for an 81 nm Au layer on a sapphire substrate. The substrate thickness is set to 1 m to represent that it is semi-infinite. *It should be noted that this code works well for cases where there is a finite thickness*.
- For unknown parameters, the term "cos" is used (this is **not** a cosine function). Thus, cos[0] is the first unknown parameter, cos[1] is the second unknown parameter, etc. **These order of these values corresponds directly to the order of the values listed in the guess vector.**

Step 4 – Phase correction

```
95    if phi_exp.any() < -180:
96        phi_exp += 180
97    elif phi_exp.any() > 0:
98        phi_exp -= 180
99
```

- If the phase lag is not corrected properly from the above code, you can manually correct by replacing the lines above with your own offset (typically still 180 degrees). This may happen if some of the data you've included results in a phase lag that is above 0 degrees, for instance.

Step 5 – Run program



- Run the current file