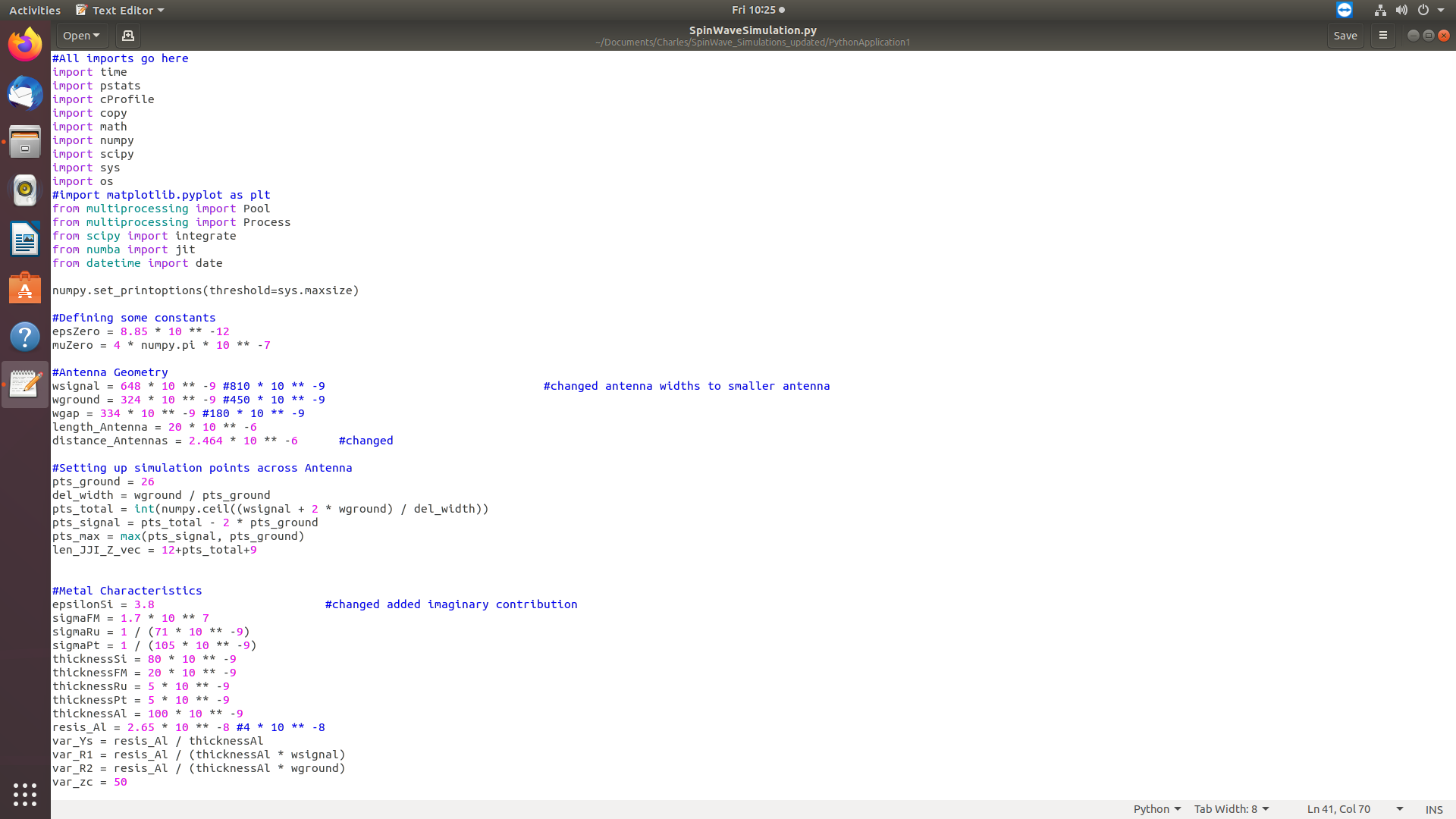
**SpinWaveSimulation.py**

The code is very long, but when running the simulation only two things need to be changed and the rest should be ignored.

* 1. The system and magnetic parameters which are the global constants at the top of the file
  2. The main function at the bottom of the file where the save path will be determined as well as name of the file to be saved and how many simulations should be run. This will be explained below.



1. The first thing to do is set the system parameters of the simulation that one wishes to run. The system parameters start with the value of the **epsZero** and end with **Hubz.** Only values between these two lines should be changed. Anything after **Hubz** should not be changed with the exception of the main function at the bottom (explained later).

All parameters should be in SI units and I will now explain what each of the parameters are.

**epsZero**: this is the permittivity of free space

**muZero**: permeability of free space

**#Antenna Geometry**

**wsignal:** width of the signal line of the CPL antenna

**wground:** width of the ground line of the CPL antenna

**wgap:** width of the signal-ground line separation

**length\_Antenna:** length of the antenna to be simulated (in other words this is the width of the Ferromagnetic stripe)

**distance\_Antennas:** this is the center-center separation of the input and output CPL antennas.

**#Setting up simulation points across the Antenna**

**pts\_ground:** number of mesh points across the width of the ground line of the CPL antenna

**del\_width:** the width of each mesh step. This is automatically calculated and should not be changed

**pts\_total:** the total number of mesh points across the entire input antenna. Again this is automatically calculated from the chosen CPL strip sizes and number of mesh points on the ground line. Should not be changed

**pts\_signal:** number of mesh points across the width of the signal line. Automatically calculated, do not change.

**pts\_max:** maximum number of mesh pts on one of the strips. Automatically calculated.

**len\_JJI\_Z\_vec:** number of columns used during the saving process. **DO NOT CHANGE!**

**#Metal Characteristics**

**epsilonSi:** relative permittivity of the SiO2 dielectric spacer layer separating the Ferromagnetic stripe and the CPL antennas

**sigmaFM:** electrical conductivity of the ferromagnetic layer

**sigmaRu:** electrical conductivity of the ruthenium layer. Note: the Ru layer is the bottom layer in the trilayer stack. If a different material should be used then simply change the conductivity accordingly.

**sigmaPt:** electrical conductivity of the platinum layer. Top layer in the trilayer stack.

**thicknessAl:** thickness of the aluminium CPL antennas.

**resis\_Al:** electrical resistivity of the aluminium CPL antennas.

**var\_Ys:** sheet resistance of the Al antennas. Automatically calculated

**var\_R1:** linear resistance of the signal CPL line. Automatically calculated

**var\_R2:** linear resistance of the ground signal lines. Automatically calculated

**#Frequency range to test with Simulation**

**freq\_lower:** starting frequency of the simulation

**freq\_upper:** end frequency of the simulation

**plot\_pts\_num:** the number of points to simulate. The frequency range, [**freq\_lower**, **freq\_upper**] will be broken up into **x** equal frequency steps where **x =** **plot\_pts\_num**.

The if-else statement that follows should not be changed.

**#Independent variables**

**centralFreq:** central frequency of the checked frequency band. This value is actually not used in the simulation and can be ignored.

**appliedH:** externally applied magnetic field strength in the plane

**linewidthSlope:** frequency dependent magnetic losses. These correspond to the slope of the HWHM (half-width at half-maximum) vs frequency plots. **linewdithSlope** = , where is the Gilbert damping constant, and is the gyromagnetic ratio.

**broadening: I**nhomogeneous linewidth broadening. Note this is the full-width broadening () and the simulation later divides this by 2 in order to get the half-width broadening ( which is added as a loss term.

**gamma:** Inhomogeneous linewidth broadening. Note this is the true inhomogeneous linewidth broadening ( and not the reduced inhomogeneous linewidth broadening (.

**ampMs:** Saturation magnetisation

**exchangeA:** exchange stiffness constant

**surface\_Ks1:** surface anisotropy constant of the bottom non-magnetic/ferromagnetic interface (Ru/Co)

**surface\_Ks2:** surface anisotropy constant of the top ferromagnetic/non-magnetic interface (Co/Pt)

**surface\_Ds1:** surface DMI constant of the bottom non-magnetic/ferromagnetic interface (Ru/Co)

**surface\_Ds2:** surface DMI constant of the top ferromagnetic/non-magnetic interface (Co/Pt)

**#Aharoni demag factors**

**NAxx:** demag factor in the x-direction corresponding to the direction along the ferromagnetic stripe. The stripe has longest length in this direction typically and thus this demag factor should be the smallest.

**NAzz:** demag factor in the y-direction corresponding to direction along the stripe width.

**NAyy:** demag factor in the z-direction corresponding to the direction along the strip thickness. This is typically the most confined geometrical direction and thus has the largest demag factor. This factor is automatically calculated from the other two.

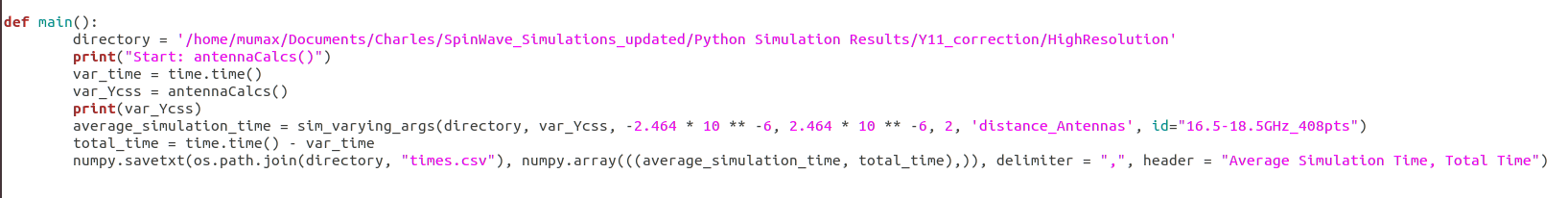
**#Bulk easy axis anisotropy**

**Hubx:** bulk easy axis anisotropy along the x-direction (along the ferromagnetic stripe length)

**Hubz:** bulk easy axis anisotropy along the z-direction (along the ferromagnetic stripe width)

These are all the system parameters that should be changed by the user and the remainder of the code (with the exception of the **main()** function discussed in point **7**) should be left the same.

1. After setting the system parameters one needs to define the save location, file name, and the number of simulations to run in the main function, as explained below. There are only two lines of code that should be changed in the main function.



* 1. The first thing that needs to be addressed is name of the directory variable: “**directory”** is the variable that stores the directory to where the simulation data should be saved. This directory needs to be supplied as a string and if it does not exist then an error will be thrown at the end of the simulation. Thus, it is a good idea to double check that the directory is indeed a valid file path. Note, the name of this directory may vary between computers and it is up to the user to supply a valid file path.
  2. The second and last point to address in the **main** function is the function call to **sim\_varying\_args.** This function is the main entry point into the simulation and does all of the calculations as well as saving. This function takes in **7** arguments, which can be changed by the user as explained below.

**sim\_varying\_args(dir, Ycss, lowB, upB, pts, arg, id)**

* + 1. **dir:** the first argument is the file path to which the simulation data should be saved. Currentlythe **directory** variable defined at the start of **main** is passed in.
    2. **Ycss**: the second argument is the linear capacitance of the CPL antenna and should not be changed. Currently the **vary\_Ycss** is passed in which is calculated by the **antennaCalcs()** function in the code.
    3. The 3rd , 4th , 5th, and 6th arguments are all connected. Using these four arguments it is possible to run several simulations in a row without manually starting new simulations.
    4. **lowB:** this is the value of the **arg** parameter for the first simulation
    5. **upB:** this is the value of the **arg** parameter for the last simulation
    6. **pts:** this is the number of simulations to run. There are 3 cases to consider:
       1. **pts=1:** Only 1 simulation will be run and the value of the **arg** parameter will be equal to **lowB.**
       2. **pts=2:** 2 simulations will be run where the first simulation will have **arg=lowB,** and the 2nd simulation will have **arg=upB.**
       3. **pts>3: pts** number of simulations will be run where the range of **arg** values will be [**lowB, upB]** inclusive and the range is broken up into **pts** even steps.
    7. **arg:** This is the system parameter which is changed between the **pts** number of simulations. This is a string and needs to be equal to the name of the system parameter as written in point 6. For example, the most typical simulation I run is where I run two simulations back to back where I change the sign of the antenna separation in order to simulate the forward and backward travelling waves. In such a case **arg= distance\_Antennas,** since **distance\_Antennas** is the parameter which defines the antenna separation.
    8. **id: (optional)** This is a string which is added to the name of the file which will be saved. This is an optional parameter and is useful for adding information to the name of the file for easier categorization later.

For the case where we wish to simulate the forward and backward travelling waves back to back we may have a call to **sim\_varying\_args()** as follows:  
**sim\_varying\_args(directory, var\_Ycss, -2.464\*10\*\*-6, 2.464\*10\*\*-6, 2, ‘distance\_Antennas’, id=”some\_test\_simulation”)**

Here **directory** and **var\_Ycss** are as explained above. **lowB= -2.464\*10 \*\*-6** and **upB= 2.464\*10\*\*-6** with **pts=2** and **arg=**’**distance\_Antennas**’. This means that 2 simulations will be run, the first one will have an antenna separation (**distance\_Antennas)** of -2.464 m and the second simulation will have an antenna separation of +2.464 m. All other system parameters will be those that are defined at the start of the file. Note, the values of **arg** supplied this way will overwrite the values supplied at the start of the python file.

Another typical example may be simulating at several different magnetic fields. In this case the call to **sim\_varying\_args()** may look something like this:

**sim\_varying\_args(directory, var\_Ycss, 100, 1000, 10, ‘appliedH’, id=”sweeping\_fields”)**

In this case a total of 10 simulations will be run back to back and for each simulation the value of the applied magnetic field (**appliedH**) will be changed. The first simulation will be for H=100 Oe and the last simulation will be for H=1000 Oe. The other **8** simulations will be evenly spread out in the range between [100,1000].

1. Once the system parameters have been set (point **6** above) and the main function has been configure (point **7** above) then the simulation is ready to be run. Now one must simply run the python script.

**File Saving**

The saving of the simulation files is done automatically to the file path supplied by the user via the **directory** variable in the **main()** function. In total, each simulation will save **3 .csv** data files and **1 .txt** file. All files saved will have the following naming format.

**“date\_arg\_arg-value\_id”**

The underscores separate the different identifiers.

**date:** The current date the simulation finished at. The format of **date** is: **year-month-day** (e.g. 2021-09-13)

**arg**: the name of the **arg** parameter used in **main()** and this corresponds to the system parameter which was changed during the various simulations (e.g. distance\_Antennas) (see point **7** above).

**arg-value:** the value of the **arg** parameter for the saved simulation. For instance if **arg=’distance\_Antennas’** then  **arg-value** may be: -2.464e-06 (see point **7** above).

**id:** This is the **id** string from **main()** and is appended as defined by the user in **main**. (e.g. “some\_text\_supplied\_by\_the\_user”).

Thus the main simulation file of interest containing the S parameters may have a name as follows:

**2021-09-13\_distance\_Antennas\_-2.464e-06\_some\_text\_supplied\_by\_the\_user.csv**

In addition to the main data file containing the S parameters there are **2** additional **.csv** files and **1** **.txt** which are saved. These are identified by the additional identifier automatically appended at the end of the file name:

**\_current.csv**

**\_impedance.csv**

**\_vars.txt**

Thus for the example above the four files saved would be:

**2021-09-13\_distance\_Antennas\_-2.464e-06\_some\_text\_supplied\_by\_the\_user.csv**

**2021-09-13\_distance\_Antennas\_-2.464e-06\_some\_text\_supplied\_by\_the\_user\_current.csv**

**2021-09-13\_distance\_Antennas\_-2.464e-06\_some\_text\_supplied\_by\_the\_user\_impedance.csv**

**2021-09-13\_distance\_Antennas\_-2.464e-06\_some\_text\_supplied\_by\_the\_user\_vars.txt**

**Format of the data files**

All **.csv** have column headers which help to identify what is saved in the column. However, let us be explicit here.

**.csv**

Let us first consider the main file, which does not have an extra identifier and is simply ending in **.csv.** This file contains **50** columns.

1. The frequencies for which the numerical calculations were performed

**2-5.**  parameter (Real, Imaginary, Amplitude, Phase). The reflection S coefficient

**6-9**. “…”. The averaged electric field across the signal strip of the CPL antenna

**10-13.**  “…”. The averaged electric field across the first ground strip of the CPL antenna

**14-17.**  “…” The averaged electric field across the second ground strip of the CPL antenna

**18-21.**  “…“. The averaged current across the signal strip of the CPL antenna??

**22-25.**  “…“. The averaged current across the first ground strip of the CPL antenna?

**26-29.**  “…”. The averaged current across the second ground strip of the CPL antenna?

**30-33.**  “…”. The transmission S parameter

**34-37.** “…”. The complex impedance of the first current loop consisting of the signal strip and the first ground strip.

**38-41.**  “…”. The complex impedance of the second current loop consisting of the signal strip and the second ground strip.

**42-45.**  “…”. ???

**46-49.**  “…”

**50.**

**\_current.csv**

Let us now discuss the format of the file ending with \_current.csv. This file saves the current distribution across the three CPL strips as a function of the microwave driving frequency. The first column of the file contains the frequencies that were simulated. All other columns store the current density for each mesh point. Note, we store the real, imaginary, amplitude, and phase of the current distribution for each mesh point across our strips. Thus, the total number of columns will be **4,** where ***pts\_total*** is the total number of mesh points as defined in point **6** above. Thus, the first number of columns after the first (frequency column) correspond to the real part of the current distribution. The ***pts\_total*** number of columns after this correspond to the imaginary part, followed by the amplitude and finally the phase. Note, the gaps between the signal strip and ground strips is not included in the mesh steps, and would need to be added manually if desired.

**\_impedance.csv**

The **\_impedance.csv** stores the values of the impedance matrix of the CPL antennas. In total there should be **37** columns. The first column stores the frequencies of the simulation. All other columns store mutual and self impedances and should be labelled accordingly, but we will discuss them again here.

1. Frequencies

**2-5.** (real, imaginary, amplitude, phase)

**6-9.**  “…“

**10-13**. “…”

**14-17.** “…”

**18-21**. “…”

**22-25**. “…”

**26-29**. “…”

**30-33**. “…”

**34-37**. “…”

Here 1 refers to the signal strip, 2 corresponds to the first ground strip, and 3 corresponds to the second ground strip. is a self-inductance of strip whilst is a mutual inductance between strips and .

**\_vars.txt**

The **\_vars.txt** stores all of the system parameters for which the simulation was run. The name of the parameters is the same as the name used in the Python file, as described in point **6** above. It should be mentioned, that for ease of reading, not all values are saved in SI units in the **\_vars.txt** file. The following values have been simplified prior to the saving of the **\_vars.txt** file to ease the reading of these units.

1. **freq\_lower:** divided by 2 to get into linear frequency *f*
2. **freq\_upper: “…”**
3. **freq\_step: “…”**
4. **centralFreq: “…”**
5. **appliedH:** divided by 79.57747 in order to get this into units of Oe (equivalent to multiplication by )
6. **gamma:** divided by to turn this into: with units of
7. **ampMs:** divided by 79.57747 in order to get this into units of Oe. (equivalent to multiplication by )

Note, all the simulations were run for the SI units and the conversions were done after the simulation was completed and are only represented in the **\_varx.txt** file for ease of reading.