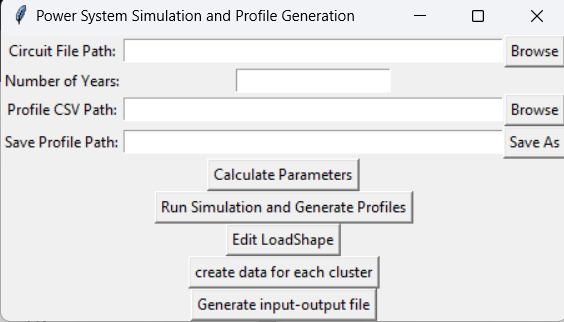
**Dataset creation:**

This file is intended to explain how to use the GUI in order to generate dataset for training and testing the neural network.

The main file that control the system is main\_gui.py.

When you run the file for the first time, a window named "Power System Simulation and Profile Generation" will show.



**Circuit File Path:** you need to enter the address of the circle you want to analyze and create data for.

**Number of Years:** you need to enter the number of years for which you want to create information. For example, if each dataframe has 8760 samples and you chose to create a dataset of 4 years, then we will get a profile with 8760\*4 samples.

**Profile CSV Path:** You need to enter the address of the dataframe on which you want to create new data.

**Save Profile Path:** You need to choose where you want to save the new profiles.

Now you can Run the simulation and Generate the profile. As you click the upper bottom, the GUI will call the "dss\_profile\_solver" file and "profile\_with\_differrent\_mu" file.

* Calculate Parameters button will calculate the parameters of the circuit that are critical to create data, such that number of loads.
* Run Simulation and Generate Profiles will make profiles for each load using the file "profile\_with\_differrent\_mu.py"
* Edit LoadShape button will create new loadshape.dss file that define the ne profiles we created.
* Generate input-output file will create csv file include the inputs of the circuit (P,Q for each load) using the profiles we created and the outputs of the circuit.
* Create data for each cluster button will divide the input-output file to clusters using the cluster dividing file.

**dss\_profile\_solver.py:**

This code includes 2 functions, the first function is "find\_number\_of\_loads", which get from the GUI the circuit path that you insert and finds the number of loads in the circuit. This is critical to find it because we need to know how many profiles to create.

The second function is "run". This function solve the circuit with openDSS commands. This function also return the input-output data for the system.

The input matrix will be with the same number of rows as the profiles we created (number\_of\_years\*examination\_period) and the number of columns will be as follows: first column will be the input voltage of the circuit, and then we will get 2 powers for each load (P and Q). (V\_in,P1,Q1,P2,Q2….)

The output matrix will be the magnitude and the angles of all the buses in the circuit.

Note that each row indicates a different time point and each column indicates a different load**.**

**profile\_with\_differrent\_mu:**

this file generates the profiles. It gets the LoadShape Path from the GUI and extracts the data from it. Then, he applies Mu law to the data and normalizes it. This method gives us an option to create big datasets from the data we got from openDSS (other methods can be checked). Depending on the number of years we entered in the GUI, the function concatenates the normalized data times the number of years, then we get a long vector with a length of 'number\_of\_years' times the length of the original LoadShape. We repeat this operation twice for each load. Finally we get 'number\_Of\_Loads' csv file named LS1,LS2,LS3… and 'number\_Of\_Loads' csv files named LS\_Q1,LS\_Q2,LS\_Q3….

Each load has an active power file and reactive power file.

After we solved the circuit, we need to update the dss load file. A new window will open:

You need to insert the load file.dss Path to it and then the GUI will call the "edit\_loadshape.py" file.

**Edit\_loadshape:**

This function create a new updated loadshape file according to the new dataset we created.

The file receives the address where we would like to save the new file and writes into it the contents of the original load file with changes according to the new data.

Actually The function will take care of assigning to each load the new data file we made for it.

The function is general for every circuit (IEE123,ckt5, etc).

**Input\_output\_for\_clusters:**

This function gets the input\_output file generated from the solver file (the big file) and also gets the cluster divide file. The function returns input and output files for each cluster. For example, if the circuit divided to 20 clusters, then the function will output 20 files for inputs and 20 files for outputs. Attention that thee are more buses than loads, so the inputs file will be smaller than the output files (because we calculate P and Q for loads only and v\_mag and v\_ang for every bus)