# GUI for Sensitivity Analysis for Multi MicroGrid Networks

**Daniel Gros, Gurupraanesh, Dr Jimmy Peng**
Department of Electrical and Computer Engineering, College of Engineering, University of Illinois at Urbana-Champaign

## INTRODUCTION

GUIs, short for Graphical User Interfaces, are used daily by most people to facilitate interaction with computers. As opposed to command line interfaces that are entirely text based, GUIs include windows, buttons, and other graphics that make it more user friendly. Most software uses GUIs for ease of use. For example the famous word processor, Microsoft Word, utilizes a GUI for most of its features including font type selection and font color selection.

Multi MicroGrid Networks are systems that integrate multiple sources of power generation (solar PV, wind) with storage solutions (battery banks, fuel cells) such that several "microgrids" are formed that are each capable of operating independently in case of emergencies. These systems are the future of distribution grids.

In order to provide power balance in the system certain types of controls must be used that share the power amongst all its sources. When operated as a multi-source network this sharing leads to an oscillating response, similar to tug of war. The oscillation can cause line or source tripping, resulting in blackouts. To restore normal operation damping must be increased; in other words the amplitude of the oscillations must be decreased by draining energy from the system.

## AIM

Develop a GUI that will display graphs based on parameters that were input by the user or based on pre-existing MATLAB scripts.

The GUI will be used as a verification tool by researchers to reproduce the research done at the NUS Power Systems Research Laboratory.

This GUI is also planned to be made open source at a later date, ensuring a transparent and verifiable research tool.

This GUI will also be able to identify which sources are most critical towards their effect on improving damping, aka minimizing the instability of the system.

## METHODS

MATLAB's Standalone Application creator failed for heavy restrictions and licensing issues (during distribution).

Python and many of its GUI frameworks offered very few restrictions and no licensing problems. The first two tried, Kivy and PySide, had issues with a small online community and non-native looking GUI and compatibility issues. The GUI framework that worked best was PyQt5.

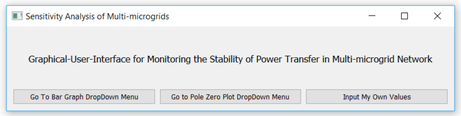
In order to upload the MATLAB scripts into the python code, the MATLAB scripts were converted into python packages which were then imported into the python code and used.

The libraries used for statistical graphics were seaborn and matplotlib, which offered aesthetic designs and lots of functionality.

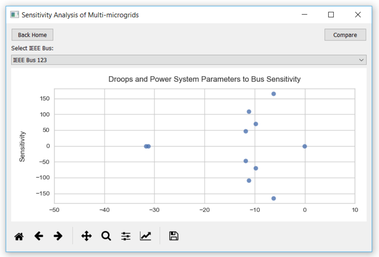
All common methods for creating the standalone application from the python code (PyInstaller, Py2exe, cx_Freeze) failed because of the plotting libraries used. Thus Pynsist was used, which doesn't freeze the python code into an executable file, but instead downloads all the python libraries necessary onto the computer and then runs the application as if it were a python file (This is all done automatically once the user double clicks on the exe file).

## RESULTS

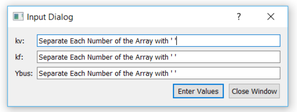
*Main Menu. Three different options offered*




*Display data through pole-zero plots and other graph types*



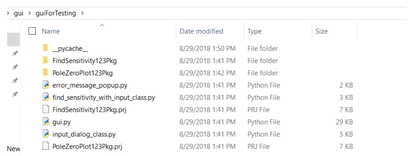

*Compare data graphically*

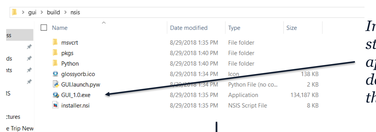




*Input your own parameters*

*Source A is the most sensitive in these graphs*

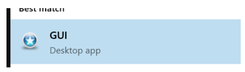
*All files and packages used to make GUI*




*Install a standalone application by double clicking on the executable*

*Standalone Application*




## CONCLUSIONS

The project was successful and there currently is a working prototype of the graphical user interface. However there are a few bugs and improvements that need to be made for future version of the GUI.

Bugs:

- Within the compare window under the "Go to Pole Zero Plot Dropdown Menu" tab, the first plot doesn't display the data points. The cause for this problem is unknown.
- Under the "Input My Own Values" tab. If numbers are put into the appropriate fields but the quantity of numbers in each field is incorrect the application will crash instead of displaying an "Invalid Input" warning as it does with other types of errors.

Improvements:

- Setting a fixed size for the main menu window without restricting the resizing of other windows
- displaying a loading bar while the GUI is starting to let the user know that the application is loading and he/she has to wait.
- Organizing the code in a way that makes it easy to add additional functionality like different types of graphs and the ability to compare three or four different plots instead of just two. This improvement was attempted but soon abandoned because of time constraints.
- Create executable in a different manner. The current one uses up too much space as it downloads the entirety of many python libraries.

## ACKNOWLEDGEMENTS

This was possible with the help of Gurupraanesh and Dr. Jimmy Peng of the Power Systems Research Laboratory at the National University of Singapore.

NUS National University of Singapore

ILLINOIS