PyPSA: Open Source Python Tool for Load Flow Study

Rishav Sharma* and Javed Dhillon
School of Electronics and Electrical Engineering,
Lovely Professional University, Punjab, India
Email ID: rishavsharma1099@gmail.com*, javeddah@gmail.com

Abstract- PyPSA is a Python-based open-source tool used for load flow analysis in the power system. PyPSA (Python for load flow analysis) is a leading tool, which is convenient for simulation and optimization of the power system network. It is a free software which regulates parametric valuation along with multi-temporal analysis in the power system. The Modern power system is the integration of various devices like Generators, Transformers, Bus bar, lines Protecting devices and load connected through it. Load flow analysis is used to determine the different types of parameters in the power system i.e. Voltage Magnitude, Phase angle, Power injected, power output and losses through the power system etc. Various kind of devices is utilised in modern power system analysis. Conventionally many techniques are adopted for power System analysis i.e. Mathematical modelling of the system and then calculation. These techniques consume very much time for the calculation. But, due to the advantage of Science and Technology, different types of new methods are developed. PyPSA is one of them. These methods are accurate, fast and reliable for power system analysis. In this study, PyPSA tool is used to analyse the IEEE 12 bus system for determining the voltage magnitude across each bus.

1. Introduction

The modern power system is an interconnection of different systems like generation system, transmission system and distribution system. For maintaining continuity of electric supply, it is required to resolve the power system losses and faults related problems as soon as possible [1]. Various problems in power systems easily resolved by using power system analysis toolbox. These toolboxes provide a variable application to analyse, modified and objectified the electrical power system. Lots of investigations are done by researchers in this field [1-4]. They are very keen to find the new methodology and various tools used to analyse the power system based problems in the fastest way at minimum cost[5 -8]. These methods and tools provide easy access to analyse the various shortcoming in the electrical power system.

1.1 Python tools for electrical power system analysis

PyPSA toolbox is such a kind of software which is based on Python programming language. This program is implemented for analysis of power system. PyPSA software provides dynamic response and it is a flexible toolbox for modelling modification. This software executes full loads flow calculations and it can also carry out optimal load flow linearisation. Since the electrical power system is quite convoluted, so there is a requirement of such a toolbox which provide easy access to the analysis of the electrical power system. PyPSA toolbox dispenses uncomplicated services for analysing the electrical power system. The main objective of this paper is to resolve power system problems with the help of PyPSA toolbox.

1.2 Other Open Source tools

For the power system calculation, There are various types of open source tools accessible today [9]. For optimal scheduling and optimal power flow, MATPOWER is used [10]. PSAT toolbox is another toolbox which is used for evaluation of dynamic network in an electrical power system. This important toolbox stands on MATLAB/ Simulink software [11]. PY-POWER toolbox is an open-source toolbox which is based on Python programming language [12]. There is another toolbox which stands on python programming language Dome[9]. This toolbox is procured from PSAT. GridCal [13] toolbox provide means for calculation of short circuit and power flow analysis. pandapower is one more toolbox which is based on Python programming language. Pandapower toolbox gives easy access to quasi-static and static analysis of the electrical power system. One of the advanced toolboxes which are standing on C/C++ language is GridLAB-D. This toolbox is also integrated with SCADA controls along with C++
language. This modern toolbox provides great access to static simulation. GridLAB-D toolbox uses an agent for simulation which is simulating the model. All these toolboxes furnish a variety of operations for an electrical power system.

2. **PyPSA**

PyPSA (Python for power system analysis) is a freely available toolbox for optimization, analysing and simulating the power system. It includes a lot of features like renewables i.e., solar, wind, conventional Generators, energy storage system work on AC as well as DC or in the mixed form[14]. It can calculate static power flow for both the linear and non-linear networks. PyPSA also provides access for calculation of optimal power flow. Security-constrained linear optimal power flow is also calculated by PyPSA toolbox. This toolbox has a model for AC and DC networks, transformer, generators, storage units etc. Non-linear power flow is also calculated by PyPSA toolbox for alternating current networks with a slack distributed bus. This toolbox is also helpful in the analysis of the behaviour of electrical power system ensuing contingencies. This is generally known as contingency analysis in a power system. PyPSA toolbox provides a wide range of applicability for the modification in modelling in the electrical power system.

PyPSA is an open-source software which also has inbuilt complementary libraries i.e., pandapower, PowerDynamics.jl, where pandapower is used for modelling of distribution type grids, short circuit calculation and unbalanced load flow. PowerDynamics.jl is used for power grid modelling which is dynamic in time scale when differential equations are relevant in a detailed and appropriate manner.

2.1 **Network Models**

Short circuit calculations, power flow etc. are the function which is analysed by the electrical power system. These analyses rely on mathematical modelling of the electrical power system. Various kinds of approaches are implemented to provide user access to describe this mathematical model by using power system tools. BBM (Bus Branch Model) is such a kind of approach which allow user for the specification of the model. Bus Branch Model is a compilation of different buses which are interconnected with each other by using generic branches. The equivalent circuit is a circuit to which the generic branches are modelled with. This is applicable for modelling multipole elements such as transformer and lines. In Bus Branch Model, buses are facing with shunt admittances and power injections for the modelling of a single-pole element. Example of single-pole elements are generators, load and capacitors banks. Bus Branch Model is used to derive the electric equation for the analysis of the electrical power system and it is highly précised mathematical presentation of a network in an electrical power system.[15]

PyPSA uses Distribution System Model in place of Bus Branch Model. This is used to modelled the network of distribution in an electrical power system. Distribution System Modelling provides easy access to remove errors which are faced during operating and designing phase of the electrical power system. Distribution System Model analyses the network which helps in resolving many problems in a power system. The distribution system consists of devices for short circuit and power flow study.

Some of the important commands used to defined the network in Python while using PyPSA toolbox are shown.

- **import** - this command is used for adding the toolbox in the python programming language.
- **network.add** – this command is used for adding buses, generators, Transformers, lines and loads etc in the programming.
- **network.pf()** – this command is used for performing load flow analysis in Python programming language.
- **network.buses_t.v_mag_pu** – this command is used to show the value of bus voltage magnitude in per unit.
• **network.lpf** - this command is used for performing linear power flow in Python programming language.

• **pypsa.opf** – this command is used for optimal power flow in the python programming language.

• **pypsa.lionpf** – this command is used for linearised optimal power flow in python programming language.

• **pypsa.contingency** - this command is used for contingency analysis of power system in a python programming language.

2.2 Generators: Generators are the starting point of the Electricity Generation. The electricity is generated at the 11KV/6.6KV and then, the 11KV/6.6KV is step up to 66KV/33KV through Power Transformers. Further this 66KV/33KV is transferred to the Transmission system of the Network. In my PyPSA tool, network.add command with generators attributes is used to define the generator in the given system and the command for adding the generator is:

```python
network.add("Generator","Generator Name", bus = "Bus No 0", control = "Slack")
```

where bus defined the location of the generator in the network and control defined the type of the generators. It works in any manner as PV generator, PQ Generator or Slack generator. To getting the output after applying the load flow is determined by the following command

```python
network.generators_t
```

For getting only real power across the generator output the command is

```python
network.generators_t.p
```

2.3 Buses: The buses basically represent the fundamental node of the electrical networks form where different branches can be taken out. It is basically used to provide the interconnection between the Generator/ Load with the transmission line. The following command is used for adding bus in the system in python programing

```python
for i in range(nbus): network.add("Bus","Bus No {}",v_nom=20, control = "Slack")
```

where `nbus` is the number of buses, `v_norm` is the nominal voltage in kV across the busses and `control` defined the type of the buses i.e. Slack, PQ or PV. and to get the output after applying the load flow is determined by the following command

```python
network.buses_t
```

For getting only voltage magnitude across the buses to output the command is

```python
network.buses_t.v_mag_pu
```

2.4 Lines: Lines are basically two types of transmission and distribution which carry the power from one bus to other. Various types of devices are connected with them like Generators, buses and loads. As an example, to connect the buses no 0 to the buses no 01 with the transmission lines having the resistance R1 and impedance X1 the command is given below

```python
network.add("Line","Line Name",bus0= "Bus No 0",bus1="Bus No 1",r = R1,x=X1)
```

For getting the output across the transmission line the command is

```python
network.lines_t
```
For getting the sending end real power flow, p0 attributes are used across the output command and for the receiving end p1 attribute is used, an example is given below:

```
network.lines_1.p0
```

2.5 Loads: Loads are the ending point of Electricity where it is consumed and utilized for different purposes. The loads are connected with distribution system i.e. Industrial, Commercial, Agriculture and Residential. The command for adding the Loads in the system

```
network.add("Load","Load Name", bus = "Bus No 1", p_set =P1 , q_set =Q1 )
```

where P1 and Q1 are the fixed or set real and reactive load connected across the Bus No 1

2.6 Storage: In the 21st century most of the work done on the electricity so it is important to have the backup of electricity available all the time. So, in the distribution system model, Energy Storage System is used for backup of electricity. The command for adding the Storage unit in the system

```
network.add("Storage","Storage Name", bus = "Bus No 1", p =Ps , q =Qs )
```

where Ps and Qs are the real and reactive power limit of storage devices connected across the Bus No 1

3. Distribution System Model

The modern electrical Power Distribution system is an interconnection of DG’s, buses, lines and loads providing electric power to all consumer. The voltage level in the distribution system is very low as compared to the transmission system because of the location of load centre nearest to the generation station and the incoming power from the transmission system is step down in various distribution substations. This power then supplied to a different primary consumer. Rest of power is supplied to distribution Transformer for step down the voltage up to 230V/440V. Afterwards, power is supply to different consumers as per the requirement. Distribution System Model is very imperative for the electrical power system. Many consumers are provided with electric power through only one transformer via secondary distribution lines. In the distribution system, there is again integrated circuit of DG’s, Bus system, distribution lines and different loads. The arrangement of the distribution system network is shown in figure 1.

![Figure 1. Distribution System Model](image-url)
4. Load Flow Analysis in the distribution system
Load flow analysis in the distribution system is used to calculate different parameters like the magnitude of voltage face angle for each bus connected to the system as well as the power flow through the systems injected power through different devices and the losses through the system.
Load flow methods for power system analysis plays a vital role in the distribution system, used to calculate the voltage and the quality of power inflow through the lines. In load flow analysis there are four important variables must be taken i.e., voltage angle, magnitude of the voltage, active and reactive power. Some of them are known while remaining are unknown [16-17] as shown in figure 2. For example in PQ bus the value of voltage magnitude and its angle are unknown on the other hand PV bus, the parameter of active and Reactive voltages are known in the study of load flow analysis the frequency is constant [18].
In my python program of load flow study for the distribution system using Newton Raphson method, is performed by modelling different devices of the distribution system like DG’s, Buses, lines and load connected through the system. PyPSA toolbox using python programming to perform load flow analysis by taking different parameters of the distribution system for calculating the various types of analysis i.e, static or dynamic. [19].

5. Simulation Results
For the sake of calculation, the Python programming toolbox i.e., PyPSA is selected for IEEE 12 bus distribution system. The single line diagram of the 12-bus distribution system is shown in figure 3. The analysis of the power system in this case study is static type and length of all transmission line is assumed to be equal to 1 km. Line and Bus data used for the case study is shown in the appendix. By applying the Load Flow analysis in PyPSA, voltage magnitude is calculated and compared with the reference result as shown in Table 1 and their visual comparison is shown in Fig. 4
Table 1. Comparison of Load Flow Solutions

| Bus No. | PyPSA Results | Reference Results [20] |
|--------|---------------|------------------------|
| 0      | 1.0           | 1.0                    |
| 1      | 0.99433       | 0.99433                |
| 2      | 0.98903       | 0.98903                |
| 3      | 0.98045       | 0.98057                |
| 4      | 0.96924       | 0.96982                |
| 5      | 0.96605       | 0.96653                |
| 6      | 0.96332       | 0.96374                |
| 7      | 0.95485       | 0.95530                |
| 8      | 0.94633       | 0.94727                |
| 9      | 0.94446       | 0.94446                |
| 10     | 0.94356       | 0.94356                |
| 11     | 0.94354       | 0.94335                |

Figure 4 provides a detailed comparison between the voltage magnitude across each bus provided by the PyPSA and the reference result shows in paper [20]. The results showed all voltage profile from PyPSA are slightly less than the reference case, meanwhile, the maximum difference percentage is 0.1% on Bus No. 8, and that difference is decreasing to zero as the voltage profile moves towards the slacks bus.

6. Conclusion

This paper presents Python programming using its PyPSA toolbox. This PyPSA is an open-source software which is used to solve various problems in the power system. Results of simulation project the voltage profile. The result of each bus is compared with the reference paper. It shows the maximum difference in the voltage magnitude is nearly 0.1% across the bus no. 8, whereas on the other bus is far below 0.05%. In term of the accuracy, the PyPSA may less than the classical load flow techniques i.e forward sweeping method but can use to compute the electrical power system as it's easy to use and having a wide variety of electrical power system components. So, we can say that PyPSA programming is useful in Resolving every condition and helpful in the case study in future.
7. References

[1] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education," IEEE Transactions on Power Systems, vol. 26, no. 1, pp. 12-19, 2011.

[2] F. Milano, L. Vanfretti, and J. C. Morataya, "An Open Source Power System Virtual Laboratory: The PSAT Case and Experience," IEEE Transactions on Education, vol. 51, no. 1, pp. 17-23, 2008.

[3] L. Thurner et al., "pandapower - an Open Source Python Tool for Convenient Modeling, Analysis and Optimization of Electric Power Systems," IEEE Transactions on Power Systems, pp. 1-1, 2018.

[4] K. J. Millman and M. Aivazis, "Python for Scientists and Engineers," Computing in Science & Engineering, vol. 13, no. 2, pp. 9-12, 2011.

[5] Y. Kongjeen and K. Bhunakitipich, "Modeling of electric vehicle loads for power flow analysis based on PSAT," in 2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2016, pp. 1-6.

[6] C. R. Myers and J. P. Sethna, "Python for Education: Computational Methods for Nonlinear Systems," Computing in Science & Engineering, vol. 9, no. 3, pp. 75-79, 2007.

[7] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," Computing in Science & Engineering, vol. 13, no. 2, pp. 22-30, 2011.

[8] A. Hariri, A. Newaz, and M. O. Faruque, "Open-source python-OpenDSS interface for hybrid simulation of PV impact studies," IET Generation, Transmission & Distribution, vol. 11, no. 12, pp. 3125-3133, 2017.

[9] F. Milano and L. Vanfretti, "State of the art and future of oss for power systems," in IEEE Power Energy Society General Meeting, July 2009.

[10] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "Mat- power: Steady-state operations, planning, and analysis tools for power systems research and education," IEEE Transactions on Power Systems, vol. 26, no. 1, pp. 12–19, Feb 2011.

[11] F. Milano, "An open-source power system analysis toolbox," IEEE Transactions on Power Systems, vol. 20, no. 3, pp. 1199–1206, Aug 2005.

[12] R. Lincoln, "Learning to trade power," PhD dissertation, University of Strathclyde, 2011.

[13] S. P. Vera, "Gridcal - research-oriented power systems software, https://github.com/SanPen/GridCal.

[14] T. Brown; J. Hörsch; and D. Schlachterberger, "PyPSA: Python for Power System Analysis," Journal of Open Research Software, vol. 6, no. 4, pp. 1-15, 2018.

[15] L. Thurner et al., "Pandapower—An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems," in IEEE Transactions on Power Systems, vol. 33, no. 6, pp. 6510-6521, Nov. 2018, DOI: 10.1109/TPWRS.2018.2829021.

[16] F. Katiraei, M.R. Iravani, Power management strategies for a microgrid with multiple distributed generation units, IEEE Trans. Power Syst. 21 (4) (Nov. 2006) 1821e1831.

[17] M. Marei, E.F. El-Saadany, M.M.A. Salama, Flexible distribution generation: FDG, in IEEE Power Engineering Society Summer Meeting, 2002, pp. 49e53.

[18] D.Q. Hung, N. Mithulananthan, Analytical expressions for DG allocation in primary distribution networks, IEEE Trans. Energy Convers. 25 (3) (Sep. 2010).

[19] T. Brown, "PyPSA Documentation Releases 0.13.1," 2018.

[20] D. Das, H. S. Nagi and D. P. Kothari, "Novel method for solving radial distribution networks," in IEE Proceedings - Generation, Transmission and Distribution, vol. 141, no. 4, pp. 291-298, July 1994, DOI: 10.1049/ip-gtd:19949966.
8. Appendix

Table 2. Line Data

| Branch No. | Sending end | Receiving end | R(ohms) | X(ohms) |
|------------|-------------|---------------|---------|---------|
| 1          | 1           | 2             | 1.093   | 0.455   |
| 2          | 2           | 3             | 1.184   | 0.494   |
| 3          | 3           | 4             | 2.095   | 0.873   |
| 4          | 4           | 5             | 3.188   | 1.329   |
| 5          | 5           | 6             | 1.093   | 0.455   |
| 6          | 6           | 7             | 1.002   | 0.417   |
| 7          | 7           | 8             | 4.403   | 1.215   |
| 8          | 8           | 9             | 5.642   | 1.597   |
| 9          | 9           | 10            | 2.89    | 0.818   |
| 10         | 10          | 11            | 1.514   | 0.428   |
| 11         | 11          | 12            | 1.238   | 0.351   |

Table 3. Load Data

| Node No. | PL (KW) | QL (KVAr) |
|----------|---------|-----------|
| 1        | 0       | 0         |
| 2        | 60      | 60        |
| 3        | 40      | 30        |
| 4        | 55      | 55        |
| 5        | 30      | 30        |
| 6        | 20      | 15        |
| 7        | 55      | 55        |
| 8        | 45      | 45        |
| 9        | 40      | 40        |
| 10       | 35      | 30        |
| 11       | 40      | 30        |
| 12       | 15      | 15        |