Main Simulator

Jake Kolevas, Aidan Gresko

May 1, 2025

1 Overview

This project is to integrate a base power systems modeling package in Python, now with an added graphical user interface (GUI) for interactive simulation. The code establishes basic classes for power system components—buses, generators, conductors, geometries, and a top-level Circuit class to link them together. The Circuit class offers network building, equivalent distance calculation, and building of the network admittance matrix (Y-bus), which are crucial for load flow analysis.

A primary new addition in this version is the MainWindow class, a PyQt6 graphical user interface (GUI) that allows users to define electrical circuits, execute powerflow simulations, and display results. The GUI has dynamic component addition (e.g., transformer, bus, loads), circuit configuration checking, and real-time feedback through text prints and matplotlib plots. It is combined with backend modules for circuit modeling, fault studies, and error handling, and the tool is more user-friendly for academic as well as real-world power system analysis.

The modular architecture of the code allows for simultaneous development and future expansion, e.g., additional advanced modeling functionality and compatibility with other power system analysis software. The ultimate aim is to have a versatile, customized platform for debating power system concepts and elementary analysis, now with enhanced user interaction using the GUI.
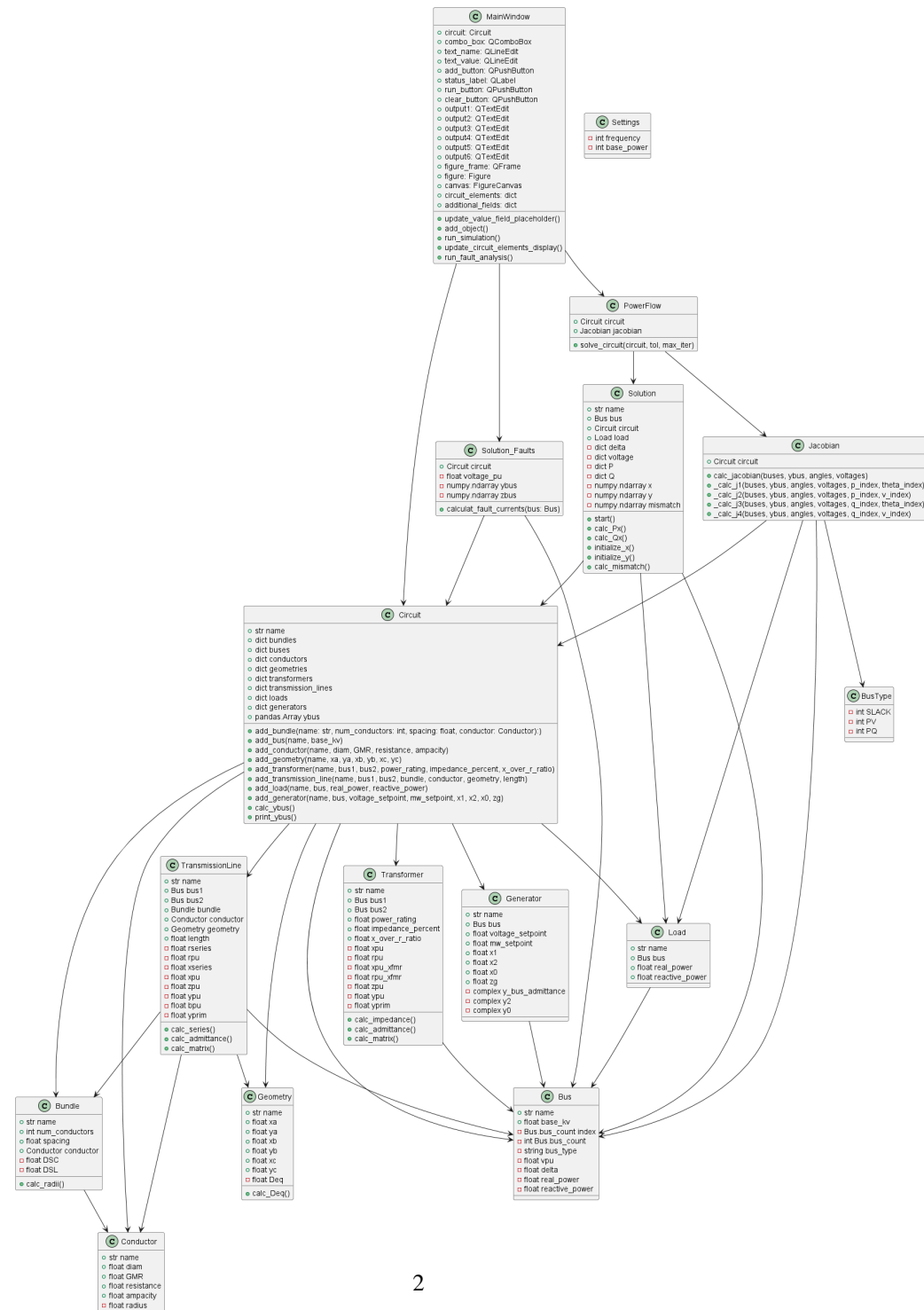
2 Class Diagram

MainWindow
- circuit: Circuit
- combo_box: QComboBox
- text_name: QLineEdit
- text_value: QLineEdit
- add_button: QPushButton
- status_label: QLabel
- run_button: QPushButton
- clear_button: QPushButton
- output1: QTextEdit
- output2: QTextEdit
- output3: QTextEdit
- output4: QTextEdit
- output5: QTextEdit
- output6: QTextEdit
- figure_frame: QFrame
- figure: Figure
- canvas: FigureCanvas
- circuit_elements: dict
- additional_fields: dict
- update_value_field_placeholder()
- add_object()
- run_simulation()
- update_circuit_elements_display()
- run_fault_analysis()

Settings
- int frequency
- int base_power

PowerFlow
- Circuit circuit
- Jacobian jacobian
- solve_circuit(circuit, tol, max_iter)

Solution
- str name
- Bus bus
- Circuit circuit
- Load load
- dict delta
- dict voltage
- dict P
- dict Q
- numpy.ndarray x
- numpy.ndarray y
- numpy.ndarray mismatch
- start()
- calc_Px()
- calc_Qx()
- initialize_x()
- initialize_y()
- calc_mismatch()

Jacobian
- Circuit circuit
- calc_jacobian(buses, ybus, angles, voltages)
- _calc_j1(buses, ybus, angles, voltages, p_index, theta_index)
- _calc_j2(buses, ybus, angles, voltages, p_index, v_index)
- _calc_j3(buses, ybus, angles, voltages, q_index, theta_index)
- _calc_j4(buses, ybus, angles, voltages, q_index, v_index)

Solution_Faults
- Circuit circuit
- float voltage_pu
- numpy.ndarray ybus
- numpy.ndarray zbus
- calculat_fault_currents(bus: Bus)

Circuit
- str name
- dict bundles
- dict buses
- dict conductors
- dict geometries
- dict transformers
- dict transmission_lines
- dict loads
- dict generators
- pandas Array ybus
- add_bundle(name: str, num_conductors: int, spacing: float, conductor: Conductor;)
- add_bus(name, base_kv)
- add_conductor(name, diam, GMR, resistance, ampacity)
- add_geometry(name, xa, ya, xb, yb, xc, yc)
- add_transformer(name, bus1, bus2, power_rating, impedance_percent, x_over_r_ratio)
- add_transmission_line(name, bus1, bus2, bundle, conductor, geometry, length)
- add_load(name, bus, real_power, reactive_power)
- add_generator(name, bus, voltage_setpoint, mw_setpoint, x1, x2, x0, zg)
- calc_ybus()
- print_ybus()

BusType
- int SLACK
- int PV
- int PQ

TransmissionLine
- str name
- Bus bus1
- Bus bus2
- Bundle bundle
- Conductor conductor
- Geometry geometry
- float length
- float rseries
- float rpu
- float xseries
- float xpu
- float zpu
- float ypu
- float bpu
- float yprim
- calc_series()
- calc_admittance()
- calc_matrix()

Transformer
- str name
- Bus bus1
- Bus bus2
- float power_rating
- float impedance_percent
- float x_over_r_ratio
- float xpu
- float rpu
- float ypu_xfmr
- float rpu_xfmr
- float zpu
- float ypu
- float yprim
- calc_impedance()
- calc_admittance()
- calc_matrix()

Generator
- str name
- Bus bus
- float voltage_setpoint
- float mw_setpoint
- float x1
- float x2
- float x0
- float zg
- complex y_bus_admittance
- complex y2
- complex y0

Load
- str name
- Bus bus
- float real_power
- float reactive_power

Bundle
- str name
- int num_conductors
- float spacing
- Conductor conductor
- float DSC
- float DSL
- calc_radii()

Geometry
- str name
- float xa
- float ya
- float xb
- float yb
- float xc
- float yc
- float Deq
- calc_Deq()

Bus
- str name
- float base_kv
- Bus bus_count index
- int Bus bus_count
- string bus_type
- float vpu
- float delta
- float real_power
- float reactive_power

Conductor
- str name
- float diam
- float GMR
- float resistance
- float ampacity
- float radius

2

Figure 1: Class Diagram for Project

3 Classes

3.1 Bundle

The Bundle class models a collection of conductors used in this simulation and calculates effective radii for resistance and inductance computations. This class is essential for modeling transmission lines, as it helps accurately compute effective radii for conductors arranged in bundles, affecting overall line behavior in analyses.

3.2 Bus

This Bus class is designed to model various types of buses within the power system, such as Slack, PV, and PQ buses. Each Bus instance is initialized with a name and base voltage (base_kv). The class maintains a count of all Bus instances created using the class-level variable bus_count. Key attributes include the bus type, voltage per unit (vpu), phase angle (delta), and power values (real_power and reactive_power). This class serves as a foundational component for simulating and analyzing power systems, enabling tracking of essential electrical characteristics for each bus.

3.3 Circuit

The Circuit class serves as a foundational component for modeling this power systems. It encapsulates the various elements that comprise a circuit—including buses, conductors, bundles, transformers, transmission lines, loads, and generators—providing a structured framework for representing and analyzing power flow. The class allows users to define and connect these components, storing their properties and relationships. A key functionality is the calc_ybus method, which computes the network admittance matrix (Y-bus) which is crucial for power system analysis. This class forms the basis for more complex power system studies, enabling simulations of load flow, fault analysis, and system stability.

3.4 Conductor

The Conductor class is designed to model electrical conductors by initializing them with key properties such as name, diameter (diam), geometric mean radius (GMR), resistance, and ampacity. Upon instantiation, these parameters are assigned to instance variables, and the radius is calculated by converting the diameter from a given unit to inches using a divisor of 24.

3.5 Generator

The Generator class models a generator component within this power system. Each generator is initialized with a name for identification, an associated bus reference (Bus), a target voltage level (voltage_setpoint), and a target power output in megawatts (mw_setpoint).

3.6 Geometry

The Geometry class models a geometric figure defined by three points (A, B, and C) with their respective coordinates. Upon initialization, it calculates an attribute Deq, which is the average distance between each pair of these points. This is achieved using the distance formula to compute the lengths of sides AB, BC, and AC, then averaging them. The class is useful for scenarios requiring a measure of the average side length of a triangle formed by three points.

3.7 Jacobian

The Jacobian class performs the Jacobian matrix calculation for power flow analysis in power systems. It provides methods for calculating each of the four submatrices (J1, J2, J3, J4) that constitute the total Jacobian. These include the real power (P) and reactive power (Q) sensitivities due to variations in voltage angles (δ) and voltage magnitudes (V). The class crosses the network buses, computing these sensitivities from the admittance matrix (Ybus) and voltage angle and magnitude current estimates. The calculated Jacobian is a critical input to iterative power flow solvers, including the Newton-Raphson solution, to determine power system operating conditions. The functions within this class calculate efficiently the partial derivatives utilized for power flow convergence.

3.8 Load

The Load class is an electrical load that is connected to some bus in the power system. Each load is described by a name, a pointer to the 'Bus' that it is connected to, and its real_power and reactive_power demands (in Watts and VARs, respectively). This class simulates the action of power usage at a node in the network and is a key object to be utilized within power system simulation and analysis. It allows demand modeling of the electricity grid and must be done in order to conduct load flow, contingency, and other power calculations.

3.9 Powerflow

The PowerFlow class computes the voltage magnitudes and angles of every bus in a power system circuit through the Newton-Raphson power flow algorithm. It is initialized with a circuit object that defines the network topology and parameters, and the power flow solution is determined iteratively. The solve_circuit method performs the primary power flow calculation with a tolerance (tol) and maximum iterations (max_iter). It employs a Jacobian object to calculate the Jacobian matrix and then proceeds to use it to update bus voltage angles and magnitudes iteratively until a converged solution is reached. The function returns a dictionary containing the convergence status, number of iterations, final mismatch, calculated voltage magnitudes and angles, calculated real and reactive power injections, and a history of the mismatch values during the iteration process, providing a complete overview of the power flow solution.

3.10 Settings

The Settings class is a store of global parameters used throughout the power system analysis. It currently sets the system frequency (in Hertz) and base_power (in MVA), base values used for per-unit calculation and conversion. This class provides simple modification and consistent application of these essential system parameters, with the benefit of code maintainability and the flexibility to represent different power system configurations. While presently constrained, it is designed to be extensible with more global settings as needed.

3.11 Seven Bus Power System

This Seven Bus Power System illustrates the development and solution of a power flow study on a typical electrical power system. It outlines the creation of a network model, including bus data, line data, and load data, in a Python setting. The code calculates the Jacobian matrix, a critical component of the power flow equations' solution, and utilizes a Newton-Raphson method to determine bus voltages and power injections. Additionally, the Seven Bus incorporates fault analysis capability, calculating fault currents and determining bus voltages under fault conditions. The code forms a foundation for power system behavior simulation, network stability verification, and contingency analysis, ultimately demonstrating an integrated approach to power system modeling and analysis.

3.12 Solution

This Solution class implements a Newton-Raphson power flow solver for power system network analysis. It solves bus voltages by iterative minimization of the mismatch between calculated and specified power injections, utilizing an external 'circuit' object to define network data and topology. The solver accommodates various bus types (Slack, PV, PQ) and features base MVA scaling for normalized calculations. By converging to a solution, it provides valuable information on steady-state operation, including voltage profiles and power flows, and is therefore a key building block for power system analysis.

3.13 Solution Symmetric

...

3.14 Transformer

The Transformer class models transformers in a power system, connecting two buses and calculating their electrical characteristics like impedance and admittance. It's essential for simulating power flow and analyzing the system's behavior in this simulation.

3.15 Transmission Line

The TransmissionLine class encapsulates the electrical characteristics of a power transmission line. It initializes with necessary components such as buses, conductors, bundles, geometry, and length. The class computes series impedance (rpu, xpu) and shunt admittance (bpu) using given formulas. These values are then used to construct an admittance matrix (yprim), essential for power flow analysis. This model allows detailed simulation of how electrical power is transmitted between two points in a power system.

3.16 MainWindow

The MainWindow class (PyQt6 UI Class) is the core GUI component of a powerflow simulation tool using PyQt6, where users can define electrical circuits, execute power-flow simulations, and visualize results. It features a dynamic user interface for adding components (e.g., buses, transformers, loads), circuit configuration checks, and simulation runs through Newton-Raphson algorithms. Outputs are bus voltages, power injections, convergence indicators, and fault analysis, displayed through text outputs and matplotlib plots. The course is coupled with backend modules for circuit modeling and fault studies with error handling and real-time feedback for user input.

4 Equations Used for Power Calculations

Bundle

1 conductor:

$$DSC = r$$
$$DSL = GMR(\text{geometric mean radius})$$

2 conductors:

$$DSC = \sqrt{r*s} - \text{r is radius and s is spacing}$$
$$DSL = \sqrt{GMR*s}$$

3 conductors:

$$DSC = \sqrt[n]{r*s^2}$$
$$DSL = \sqrt[n]{GMR*s^2}$$

4 conductors:

$$DSC = 1.091*(r*s^4)^{1/4}$$
$$DSL = 1.091*(GMR*s^4)^{1/4}$$

Where:
- DSC is the equivalent diameter of a single circular conductor
- DSL is the geometric mean radius of the bundle
- r is the radius of individual conductors
- s is the spacing between conductors

Circuit

$$\text{Y-Bus}[i, i] = \text{sum of all self-admittances at bus i}$$

$$\text{Y-Bus}[i, j] = \text{-sum of mutual admittances between buses i and j}$$

$$Z_{\text{transformer}} = \frac{\text{base\_voltage}^2}{\text{power\_rating}} * \frac{\text{impedance\_percent}}{100}$$

$$G_{\text{line}} = \text{conductance per unit length} * \text{length}$$

$$B_{\text{line}} = \text{susceptance per unit length} * \text{length}$$

$$Z_{\text{line}} = R + jX$$

Generator

$$Y_1 = 1/jX_1$$
$$Y_2 = 1/jX_2$$
$$Y_0 = 1/(jX_0 + 3Zg)$$

Where:
- Y represents admittance (per unit or Siemens)
- X represents reactance (per unit or Ohms)
- Zg represents ground resistance (Ohms)
- j is the imaginary unit ($\sqrt{-1}$)

Geometry

Euclidean Distance Formula: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Jacobian

Active Power:

$$P_i = V_i \sum_{j=1}^{n} V_j[G_{ij} \cos(\delta_i - \delta_j) + B_{ij} \sin(\delta_i - \delta_j)]$$

Reactive Power:

$$Q_i = V_i \sum_{j=1}^{n} V_j[G_{ij} \sin(\delta_i - \delta_j) - \cos(\delta_i - \delta_j)]$$

Jacobian Sub-matrices:

J1:

$$J1_{ij} = \begin{cases} -V_i^2 \sum_{k\neq i} Y_{ik} \sin(\delta_i - \delta_k - \theta_{ik}) \text{ if } i = j \\ V_i V_j Y_i \sin(\delta_i - \delta_j - \theta_{ik}) \text{ if } i \neq j \end{cases}$$

J2:

$$J2_{ij} = \begin{cases} V_i Y_{ii} \cos(\delta_i - \delta_i - \theta_{ii}) + V_i \sum_{k\neq i} Y_{ik} \cos(\delta_i - \delta_k - \theta_{ik}) \text{ if } i = j \\ V_i Y_{ij} \cos(\delta_i - \delta_j - \theta_{ij}) \text{ if } i \neq j \end{cases}$$

J3:

$$J3_{ij} = \begin{cases} V_i^2 \sum_{k\neq i} Y_{ik} \cos(\delta_i - \delta_k - \theta_{ik}) \text{ if } i = j \\ -V_i V_j Y_{ij} \cos(\delta_i - \delta_j - \theta_{ij}) \text{ if } i \neq j \end{cases}$$

J4:

$$J4_{ij} = \begin{cases} -V_i Y_{ii} \sin(\delta_i - \delta_i - \theta_{ii}) + V_i \sum_{k\neq i} Y_{ik} \sin(\delta_i - \delta_k - \theta_{ik}) \text{ if } i = j \\ V_i Y_{ij} \sin(\delta_i - \delta_j - \theta_{ij}) \text{ if } i \neq j \end{cases}$$

Where:
- V_i and V_j: Voltage magnitudes at buses i and j
- G_{ij} and B_{ij}: Real and imaginary parts of the admittance between buses i and j
- Y_{ii}: Admittance between buses i and j, where $Y_{ij} = G_{ij} + jB_{ij}$
- δ_i and δ_j: Phase angles at buses i and j
- θ_{ij}: Phase angle of the admittance Y_{ij}

PowerFlow

Newton-Raphson Method:

$$J \cdot dx = -\text{mismatch}$$

Angle and Voltage Corrections:

$$\Delta\theta_i^{k+1} = \Delta\theta_i^k + dx_i$$

Solution

Active Power Mismatch:

$$\Delta P_i = P_{spec,i} - P_i$$

Reactive Power Mistmatch:

$$\Delta Q_i = Q_{spec,i} - Q_i$$

Solution Faults

Inverse Matrix:
$$Z_{bus} = (Y_{bus})^{-1}$$

Fault Current at Bus:
$$I_f(n) = \frac{V_{pu}}{Z_{nn}}$$

Bus Voltage Post-Fault:
$$V_k = (1 - \frac{Z_{kn}}{Z_{nn}}) \times V_{pu}$$

Transformer

Transformer Impedance:
$$Z_{pu} = \frac{Z_\%}{100} \cdot \frac{S_{\text{base}}}{S_{\text{transformer}}} \angle\theta$$

where $\theta = \tan^{-1}(X/R)$.

Transformer Admittance:
$$Y_{pu} = \frac{1}{Z_{pu}}$$

Transformer Y-Bus Matrix:
$$[Y_{\text{Bus}}] = \begin{bmatrix} Y_{11} & -Y_{12} \\ -Y_{21} & Y_{22} \end{bmatrix}$$

where $-Y_{12} = -Y_{21}$.

Transmission Line

Series Resistance Calculation:
$$R_{series} = (\frac{p}{n}) \times L$$

Series Reactance Calculation:
$$X_{series} = 2\pi f \times \mu_0 \times \log(\frac{D_{eq}}{D_{SL}}) \times L$$

Per Unit Impedance:
$$Z_{pu} = R_{pu} + jX_{pu}$$

Shunt Admittance Calculation:
$$B_{shunt} = 2\pi f \times \frac{\epsilon_0}{\log(\frac{D_{eq}}{D_{SC}})} \times L$$

Per Unit Shunt Admittance Calculation:

$$B_{pu} = \frac{B_{shunt}}{Y_{base}}$$

Prim Matrix Calculation:

$$Y' = \begin{bmatrix} Y_p + B/2 & -Y_p \\ -Y_p & Y_p + B/2 \end{bmatrix}$$

Where:
- Y_p is the per unit admittance due to series resistance and reactance.
- B is the shunt conductance (real part of admittance).

5 Example Problem with Solution

This section presents a practical example of power flow analysis using the developed graphical user interface. We will analyze a 7-bus power system, calculate power flows, and interpret the results.

5.1 Problem Statement

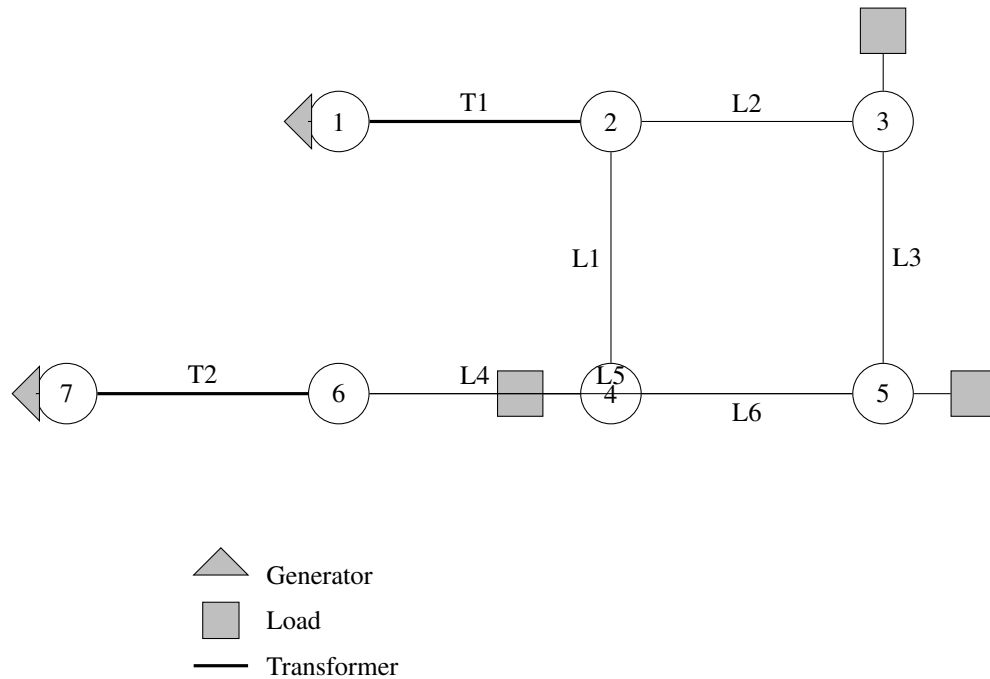Consider the 7-bus power system shown in Figure 2 with the following parameters:

Figure 2: 7-Bus Power System

Bus Data:

- Bus 1: Slack bus, voltage $V_1 = 1.0\angle 0°$ pu, base voltage 20 kV

- Bus 2: Load bus (PQ), base voltage 230 kV

- Bus 3: Load bus (PQ), base voltage 230 kV

- Bus 4: Load bus (PQ), base voltage 230 kV

- Bus 5: Load bus (PQ), base voltage 230 kV

- Bus 6: Load bus (PQ), base voltage 230 kV

- Bus 7: Generator bus (PV), voltage magnitude $V_7 = 1.0$ pu, $P_{G7} = 200$ MW, base voltage 18 kV

Load Data:

- Bus 3: $P_{L3} = 110$ MW, $Q_{L3} = 50$ MVAR

- Bus 4: $P_{L4} = 100$ MW, $Q_{L4} = 70$ MVAR

- Bus 5: $P_{L5} = 100$ MW, $Q_{L5} = 65$ MVAR

Transformer Data:

Name	From Bus	To Bus	MVA Rating	Impedance (%)
T1	1	2	125	8.5
T2	6	7	200	10.5

Table 1: Transformer Parameters

Transmission Line Data:

Line	From Bus	To Bus	Conductor	Length (mi)
L1	2	4	C1	10
L2	2	3	C1	25
L3	3	5	C1	20
L4	4	6	C1	20
L5	5	6	C1	10
L6	4	5	C1	35

Table 2: Transmission Line Parameters

Conductor Data:

- Type: C1

- Resistance: 0.642 Ω/mile

- Reactance: 0.0217 Ω/mile

- Diameter: 0.385 inches

- Ampacity: 460 A

Bundle Configuration:

- Bundle B1: 2 conductors of type C1 with 1.5 ft spacing

Geometry Configuration:

- Geometry G1: Horizontal configuration with phase positions at (0,0), (18.5,0), and (37,0) ft

The objective is to determine:

1. The voltage magnitude and angle at each bus

2. Power flow in all transmission lines and transformers

3. Generator power output at the slack bus

4. System losses

5.2   Solution Approach

To solve this power flow problem using our GUI application, we follow these steps:

1. Input all bus data, including voltage levels and bus types

2. Add conductor data and bundle configurations

3. Add geometry configurations for transmission lines

4. Input transformer parameters

5. Add transmission lines with proper connections

6. Specify the load values

7. Configure the generators

8. Run the Newton-Raphson power flow algorithm

9. Analyze the results

The following sections detail this process and show the resulting outputs.

5.3   Implementation in the GUI

Using the GUI application, we enter the components as shown in the sequence below:

Component Type	Name	Value
Bus	Bus1	20,Slack Bus
Bus	Bus2	230,PQ Bus
Bus	Bus3	230,PQ Bus
Bus	Bus4	230,PQ Bus
Bus	Bus5	230,PQ Bus
Bus	Bus6	230,PQ Bus
Bus	Bus7	18,PV Bus
Transformer	T1	Bus1,Bus2,125,8.5
Transformer	T2	Bus6,Bus7,200,10.5
Conductor	C1	0.642,0.0217,0.385,460
Bundle	B1	2,1.5,C1
Geometry	G1	0,0,18.5,0,37,0
Transmission Line	L1	Bus2,Bus4,B1,C1,G1,10
Transmission Line	L2	Bus2,Bus3,B1,C1,G1,25
Transmission Line	L3	Bus3,Bus5,B1,C1,G1,20
Transmission Line	L4	Bus4,Bus6,B1,C1,G1,20
Transmission Line	L5	Bus5,Bus6,B1,C1,G1,10
Transmission Line	L6	Bus4,Bus5,B1,C1,G1,35
Load	Load3	Bus3,110,50
Load	Load4	Bus4,100,70
Load	Load5	Bus5,100,65
Generator	G1	Bus1,1.0,0.0,0.12,0.14,0.05,0
Generator	G7	Bus7,1.0,200,0.12,0.14,0.05,0

Table 3: Component Input Sequence for GUI

5.4   Power Flow Results

After running the simulation, the GUI generates the following results:

5.4.1 Bus Voltages

Bus	Voltage Magnitude (pu)	Voltage Angle (degrees)	Type
Bus1	1.0000	0.00	Slack
Bus2	0.9360	-4.46	PQ
Bus3	0.9194	-5.49	PQ
Bus4	0.9288	-4.72	PQ
Bus5	0.9256	-4.86	PQ
Bus6	0.9385	-3.97	PQ
Bus7	1.0000	2.08	PV

Table 4: Bus Voltage Results

5.4.2 Power Flow in Lines and Transformers

Bus	Active Power (MW)	Reactive Power (MVAR)
Bus1	116.36	87.13
Bus2	0.00	0.00
Bus3	-110.00	-50.00
Bus4	-100.00	-70.00
Bus5	-100.00	-65.00
Bus6	0.00	0.01
Bus7	200.00	107.66

Table 5: Bus Power Injections

5.4.3 Generator Outputs

Bus	Generator	Active Power (MW)	Reactive Power (MVAR)
Bus1	G1	116.36	87.13
Bus7	G7	200.00	107.66

Table 6: Generator Power Outputs

5.4.4 System Losses

Type	Value
Active Power Losses	6.36 MW
Reactive Power Losses	9.80 MVAR

Table 7: System Losses

5.4.5 Convergence

The power flow solution converged in 2 iterations with a maximum power mismatch of 5.3×10^{-5} pu.

5.5 Analysis of Results

1. **Voltage Profile**: All bus voltages are within acceptable limits (0.92-1.05 pu), with the lowest voltage at Bus 3 (0.9194 pu).

2. **Power Flow Distribution**: The power flows from the two generators (at Bus 1 and Bus 7) to the three load centers (at Bus 3, Bus 4, and Bus 5). The generators at Bus 1 and Bus 7 supply 116.36 MW and 200 MW respectively.

3. **Generator Outputs**: The slack bus (Bus 1) generates 116.36 MW and 87.13 MVAR to balance the system, while the PV bus (Bus 7) maintains its scheduled 200 MW output with 107.66 MVAR reactive power support.

4. **System Losses**: The total system losses are 6.36 MW (2.0

5. **Load Balance**: Total generation (316.36 MW) equals the sum of total load (310 MW) and system losses (6.36 MW), confirming power balance.

6. **Fast Convergence**: The solution converged in only 2 iterations, indicating good numerical stability of the model.

5.6 Visualization

Figure 3 shows the voltage profile across the system, providing a visual representation of the voltage variations.
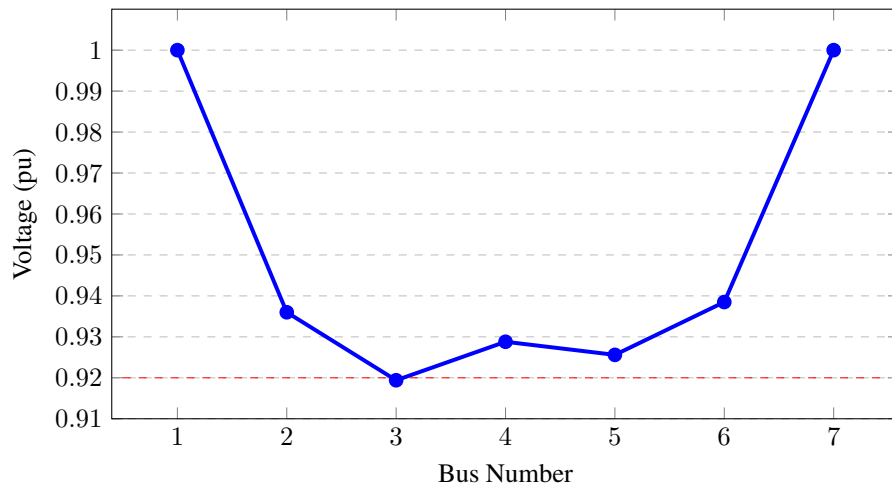

Figure 3: Voltage Profile Across the System

5.7 Fault Analysis

To demonstrate the fault analysis capabilities of the GUI, we simulate a three-phase fault at Bus 1. The resulting fault currents and post-fault bus voltages are shown below:

Bus	Fault Current (pu)
Bus1	11.94

Table 8: Fault Current at Bus 1

Bus	Voltage During Fault (pu)
Bus1	0.0000
Bus2	0.2458
Bus3	0.2764
Bus4	0.2760
Bus5	0.3024
Bus6	0.3239
Bus7	0.5302

Table 9: Bus Voltages During Fault at Bus 1

The fault analysis shows that a three-phase fault at Bus 1 causes significant voltage drops throughout the system. The voltage at the fault location drops to zero, while other buses experience voltage drops proportional to their electrical distance from the fault location. Bus 7, which has the largest electrical distance from the fault, maintains the highest voltage during the fault at 0.5302 pu.

5.8 Conclusion

This example demonstrates how the GUI application can be used to model, solve, and analyze a realistic 7-bus power system. The Newton-Raphson method successfully converged to an accurate solution that satisfies power flow equations and system constraints. The visualization tools help identify potential issues like voltage violations, while the fault analysis provides insights into system behavior under contingency conditions.

The example shows that the application can effectively handle the fundamental tasks of power system analysis:

- Modeling complex transmission networks with transformers and multiple voltage levels

- Calculating steady-state power flow for systems with multiple generators and loads

- Determining bus voltages and branch power flows

- Analyzing system losses

- Performing basic fault studies

These capabilities make the application a valuable educational tool for understanding power system concepts and performing preliminary system studies.