Simple Circuit Simulator Final Report

Joseph Rivera

February 3, 2026

Dr. Kerestes
Computer Analysis of Power Systems

1 Project Overview

The purpose of this project is to develop a Python-based simulator capable of modeling and solving a simple DC circuit. The objective is to obtain foundational knowledge required to build a power system simulator from scratch using a voltage source, a series resistor, and a load resistor connected between Bus A and Bus B.

The program calculates and displays the voltage at each bus as well as the current flowing through the circuit. This project emphasizes real-world applications of circuit analysis and power system modeling, while reinforcing object-oriented programming concepts and numerical problem solving.

2 Class Diagrams

The simulator is implemented using multiple object-oriented classes that represent the fundamental components of the circuit, including buses, resistive elements, and the overall circuit structure. Each class contains attributes and methods that define the electrical properties and behaviors of the corresponding component, allowing the circuit to be constructed and analyzed programmatically.

Figure 1 shows the class structure used to model the circuit simulator. The diagram illustrates the relationships between buses, circuit elements, and supporting classes used in the implementation.

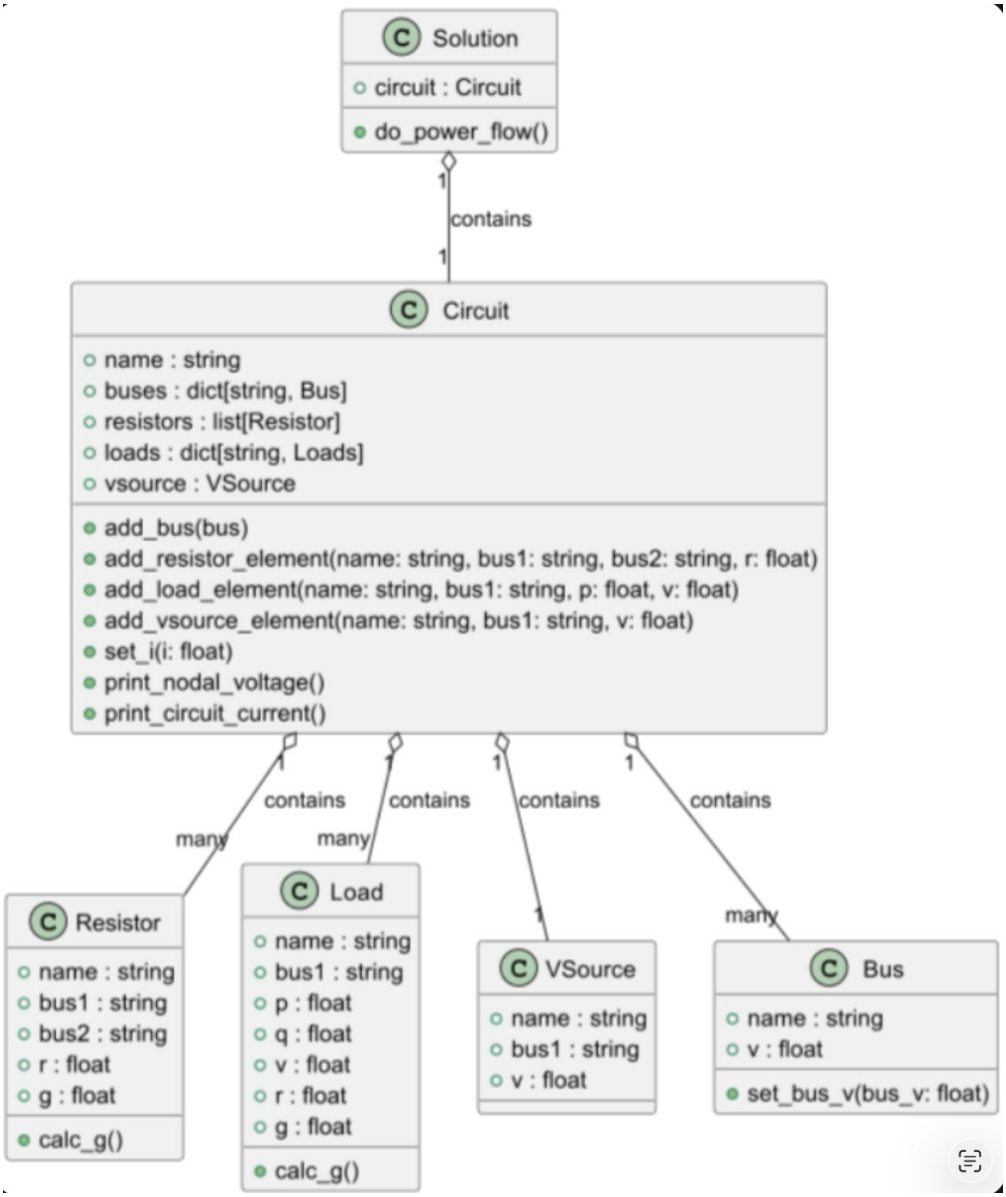

Figure 1: Class diagram for the simple circuit simulator

3 Class Description

3.1 Bus Class

The `Bus` class represents an electrical node in the circuit where components such as voltage sources, resistors, and loads are connected. In the context of the simple circuit simulator, the bus serves as a reference point for storing and updating voltage values during circuit analysis.

3.1.1 Attributes

- `name`: A string identifier used to label the bus (e.g., Bus A or Bus B).
- `v`: A floating-point variable representing the voltage magnitude at the bus. This value is initialized to 0.0 and updated during the solution process.

3.1.2 Methods

- `set_bus_v(bus_v)`: Assigns a new voltage value to the bus. This method is used after solving the circuit equations to update the bus voltage based on calculated results.

3.1.3 Role in the Simulator

The `Bus` class provides a structured way to store and manage node voltage information. By encapsulating voltage data within a dedicated class, the simulator maintains modularity and allows for scalable expansion to larger power system models with additional buses.

3.2 Resistor Class

The `Resistor` class models a resistive element connected between two buses in the circuit. This class represents the physical resistance that limits current flow and causes voltage drops between connected buses. In the simulator, resistors are used to model transmission elements and series resistances within the system.

3.2.1 Attributes

- `name`: A string identifier used to label the resistor.
- `bus1`: A reference to the first `Bus` object connected to the resistor.
- `bus2`: A reference to the second `Bus` object connected to the resistor.
- `r`: A floating-point value representing the resistance in ohms.
- `g`: The conductance of the resistor, computed as the inverse of the resistance.

3.2.2 Methods

- `calc_g()`: Calculates and returns the conductance of the resistor using the relationship $G = \frac{1}{R}$. This method is called during object initialization to ensure the conductance value is available for circuit calculations.

3.2.3 Role in the Simulator

The `Resistor` class provides a bridge between two buses and contributes to the overall conductance of the circuit. By storing both resistance and conductance values, the simulator can efficiently perform current and voltage calculations while maintaining a clear mapping between physical circuit elements and their software representations.

3.3 Load Class

The `Load` class models a constant impedance electrical load connected to a bus in the circuit. This class represents power-consuming elements within the system and is used to simulate how loads draw current based on specified power and nominal voltage ratings.

3.3.1 Attributes

- `name`: A string identifier used to label the load.
- `bus1`: A reference to the `Bus` object to which the load is connected.
- `p`: A floating-point value representing the rated power consumption of the load.
- `vnom`: The nominal voltage at which the load is specified.
- `g`: The conductance of the load, calculated based on its power and nominal voltage.

3.3.2 Methods

- `calc_g()`: Computes and returns the load conductance using the relationship

$$G = \frac{P}{V_{\text{nom}}^2}.$$

  This formulation allows the load to be treated as a constant impedance element during circuit analysis.

3.3.3 Role in the Simulator

The `Load` class enables realistic modeling of power consumption at a bus by converting power and voltage specifications into an equivalent conductance. This approach simplifies circuit calculations while preserving accurate load behavior, allowing the simulator to scale naturally to more complex power system models.

3.4 Voltage Source Class

The `Vsource` class represents an ideal DC voltage source connected to a bus in the circuit. This class is responsible for defining the reference voltage at the connected bus and serves as the driving source for the entire circuit.

3.4.1 Attributes

- `name`: A string identifier used to label the voltage source.
- `bus1`: A reference to the `Bus` object to which the voltage source is connected.
- `v`: A floating-point value representing the voltage magnitude of the source.

3.4.2 Methods

The `Vsource` class does not implement additional computational methods. Instead, its functionality is realized during initialization by assigning the specified voltage value to the connected bus.

3.4.3 Role in the Simulator

Upon creation, the `Vsource` class sets the voltage of the associated bus using the `set_bus_v()` method. This establishes the reference voltage for the circuit and allows subsequent calculations of current and downstream bus voltages to be performed consistently. By modeling the source as ideal, the simulator focuses on resistive and load behavior without introducing source impedance effects.

3.5 Circuit Class

The `Circuit` class acts as the main container and control structure for the simple circuit simulator. It is responsible for managing all circuit components, including buses, resistors, loads, and the voltage source, and for storing the computed circuit current. This class enables systematic construction, analysis, and reporting of circuit behavior.

3.5.1 Attributes

- `name`: A string identifier used to label the circuit.
- `buses`: A dictionary that maps bus names to `Bus` objects, allowing dynamic creation and retrieval of circuit nodes.
- `resistors`: A dictionary containing all `Resistor` objects in the circuit, indexed by their names.
- `loads`: A dictionary containing all `Load` objects connected within the circuit.
- `vsource`: A reference to the `Vsource` object supplying the circuit.
- `i`: A floating-point variable representing the total current flowing through the circuit.

3.5.2 Methods

- `add_bus(bus)`: Creates a new `Bus` object and adds it to the circuit.
- `add_resistor_element(name, bus1_name, bus2_name, r)`: Adds a resistive element between two existing buses and stores it within the circuit.
- `add_load_element(name, bus1_name, p, v)`: Creates a load element connected to a specified bus and stores it in the circuit.
- `add_vsource_element(name, bus1, v)`: Assigns a voltage source to the circuit and applies the source voltage to the associated bus.

- `set_i(i)`: Stores the calculated circuit current.
- `print_nodal_voltage()`: Displays the voltage at each bus in the circuit.
- `print_circuit_current()`: Outputs the total circuit current.

3.5.3 Role in the Simulator

The `Circuit` class integrates all individual component classes into a cohesive system model. By centralizing element creation, storage, and result reporting, this class enables modular circuit construction and straightforward expansion to more complex power system simulations. It serves as the backbone of the simulator, coordinating data flow between components and supporting validation and analysis of circuit behavior.

3.6 Solution Class

The `Solution` class is responsible for performing the circuit analysis and computing the electrical state variables for the simulator. It acts as the solver layer by extracting parameters from the `Circuit` object and applying the required circuit equations to compute bus voltages and the total circuit current.

3.6.1 Attributes

- `Circuit`: A reference to the `Circuit` object being solved. This allows the solver to access buses, resistors, loads, and the voltage source.

3.6.2 Methods

- `do_power_flow()`: Executes the circuit solution procedure. This method:
  - Reads the source voltage V_a from the circuit voltage source and assigns it to Bus A.
  - Extracts the resistance and conductance values from the resistor element connecting buses.
  - Extracts the load conductance value from the load element.
  - Computes the total circuit current using the complete circuit current equation:

$$I = \frac{V_a}{\left(\frac{1}{G_{ab}} + \frac{1}{G_{load}}\right)}.$$

  - Computes Bus B voltage using the load impedance relationship:

$$V_b = I\left(\frac{1}{G_{load}}\right).$$

  - Updates the circuit current and stores the solved bus voltages back into the corresponding `Bus` objects.

3.6.3 Role in the Simulator

The `Solution` class separates the solution logic from the component definitions, which improves modularity and keeps the simulator organized. By implementing a dedicated solver method, the simulator can be extended in the future to support additional elements, multiple loads, or more advanced solution techniques while maintaining a clear structure.

3.7 Main Program Execution

The main program serves as the entry point for the simple circuit simulator. It is responsible for instantiating the circuit, defining all system components, executing the power flow solution, and displaying the calculated results.

3.7.1 Functionality

The execution process follows a structured sequence to ensure proper circuit construction and analysis:

- A `Circuit` object is created to represent the overall system.
- Two buses, Bus A and Bus B, are added to the circuit.
- A voltage source is connected to Bus A and initialized with a specified voltage magnitude.
- A resistor is added between Bus A and Bus B to model the series resistance.
- A constant impedance load is connected to Bus B using rated power and nominal voltage values.
- A `Solution` object is created and used to execute the power flow calculation.
- The computed nodal voltages and total circuit current are printed to the console.

3.7.2 Role in the Simulator

The main program demonstrates how the simulator components interact to form a complete power system model. By clearly defining the circuit structure and invoking the solver, this script validates the modular design of the simulator and provides a reproducible workflow for testing and verification. The structure also allows users to easily modify system parameters for additional validation cases or future extensions.

4 Relevant Equations

The following equations are used throughout the simulator to analyze circuit behavior:

$$V = IR \tag{1}$$

(Ohm's Law)

$$G = \frac{P}{V^2} \tag{2}$$

(Conductance–Power Relationship)

$$G = \frac{1}{R} \tag{3}$$

(Conductance–Resistance Relationship)

$$I = \frac{V_a}{\left(\frac{1}{G_{ab}} + \frac{1}{G_{load}}\right)} \tag{4}$$

(Complete Circuit Current Equation)

$$V_b = I\left(\frac{1}{G_{load}}\right) \tag{5}$$

(Bus B Voltage Equation)

5 Example Case

5.1 Problem Definition

The goal of this program is to calculate the voltages at Bus A and Bus B, as well as the current flowing through the entire circuit. This allows for the modeling of a simplified power system circuit and provides a foundation for more advanced power system simulations.

5.2 Solution Process

The solution approach involved developing multiple Python classes that represent all components required to construct the circuit. Each class includes attributes and methods used to define electrical characteristics such as resistance, conductance, voltage, and current.

Using object-oriented programming principles, the circuit is assembled and analyzed by applying Kirchhoff's Voltage Law, conductance relationships, and basic circuit analysis techniques. This modular structure allows for clarity, extensibility, and easier debugging.

5.3 Expected Output and Validation

To validate the simulator, standard values with known analytical solutions were used. In the first validation case, the system consisted of two buses (Bus A and Bus B), a voltage source of 100 V, a series resistor of 5 Ω, and a constant impedance load rated at 2000 W with a nominal voltage of 100 V.

The validated output for this case was:

- Bus A Voltage: 100 V

- Bus B Voltage: 50.0 V
- Circuit Current: 10.0 A

To further verify correctness, a second case was evaluated using doubled system values. This case included a voltage source of 200 V, a resistor of 10 Ω, and a constant impedance load of 4000 W with a nominal voltage of 200 V.

The resulting output was:

- Bus A Voltage: 200 V
- Bus B Voltage: 100 V
- Circuit Current: 10.0 A

These results are consistent with theoretical expectations, confirming that the simulator correctly scales with system parameters and produces accurate solutions.