SPICE Circuit Simulator

Krutarth Patel

7th September 2024

1 Introduction

This report presents a circuit simulator that performs nodal analysis to evaluate the behavior of electrical circuits. The simulator supports various basic circuit components, including resistors, current sources, and voltage sources. The tool is designed to be easily extensible, allowing for additional circuit elements and features in future versions.

2 Nodal Analysis

Nodal analysis is a systematic method for determining the voltage at each node of an electrical circuit. It involves solving a system of equations derived from Kirchhoff's Current Law (KCL), which states that the algebraic sum of currents entering a node is zero. The analysis typically requires the construction of an admittance matrix and a vector of current values, leading to a linear system of the form:

$$A \cdot C = B$$

Where:

- A is the admittance matrix (formed by the elements of the circuit),
- C is the vector of unknown node voltages,
- B is the vector of current values.

3 Methodology

3.1 Circuit Elements

The circuit simulator supports the following elements:

- **Resistors (R)**: Represented by the equation $V = I \cdot R$, resistors are placed in the admittance matrix with conductance values $G = 1/R$. [1]
- **Current Sources (I)**: These sources inject a constant current into the nodes.
- **Voltage Sources (V)**: Voltage sources impose a constant voltage difference between two nodes, introducing additional equations into the system.

3.2 Circuit Representation

The circuit is internally represented using an adjacency list, where each node maintains a list of elements connected to it. The simulator parses a SPICE-like circuit description file to construct this list.

3.3 Admittance Matrix and Solution

The simulator constructs the admittance matrix and solves the system of equations using the following steps:

1. Parse the circuit description file.
2. Construct the adjacency list for the circuit.
3. Populate the admittance matrix (A) and the vector (B) based on the components.
4. Solve the linear system $A \cdot C = B$ using NumPy's linear solver.

[1] This poses a problem, a resistance with value 0 will not give a desired solution. As a workaround one can specify the short-circuit as a zero voltage source.

4 Implementation

4.1 Overview

The code uses Object Oriented Programming to write modular code. It represents elements like Resistance, Current, Voltage as objects inheriting a BasicElement class. This approach results in more readable code.

Listing 1: BasicElement

```
class BasicElement:
  def __init__(self, filtered_lst) -> None:
    try:
      self.value = float(filtered_lst[-1])
      self.node1 = filtered_lst[1]
      self.node2 = filtered_lst[2]
      self.name = filtered_lst[0]
    except IndexError:
      print("Expected Format : NAME Ni Nj (TYPE OF SOURCE) VALUE")
      print(f"Found: {filtered_lst}")
      raise ValueError("Malformed circuit file") from None

  def place_admittance(self, Amatrix, Bmatrix, node, no_of_nodes):
    pass
```

4.2 Matrix Construction

The admittance matrix (A) is populated by iterating over the nodes and elements connected to each node:

Listing 2: Matrix Construction

```
def construct_matrix(Amatrix, Bmatrix, adj_list):
  for node in adj_list.keys():
    if node == "GND":
      continue
  for element in adj_list[node]:
    element.place_admittance(Amatrix, Bmatrix, node, len(adj_list) - 1)
```

4.3 Circuit Solution

The linear system is solved using NumPy's linear solver:

Listing 3: Solving the Circuit Equations

```
Cmatrix = np.linalg.solve(Amatrix, Bvector)
```

5 Results

Upon solving the circuit, the node voltages and currents through the voltage sources are displayed as the output. The simulator returns a tuple of two lists:

- **Node Voltages:** The potential at each node relative to the ground (GND).
- **Current through Voltage Sources:** The current flowing through each voltage source.

5.1 Sample Output

For a given circuit, the simulator may output the following:

Listing 4: Sample Output

```
# Node Voltages
{'n1': 5.0, 'n2': 0.0, 'GND': 0.0}

# Current through Voltage Sources
{'V1': 2.5}
```

6 Conclusion

This report presented a Python-based circuit simulator that uses nodal analysis to solve for node voltages and currents in a circuit. The modular design allows easy extension for additional circuit components in future iterations.