Assignment 2

Aswitha Sriee, EE23B092

September 8, 2024

References

The resources used include (hyperlink attached below):

- Fundamentals of SPICE programming
- pandoc manual

Description of the Code

Libraries

- **numpy** and **numpy.linalg.solve**: Used for matrix operations and solving linear systems of equations.
- **os**: Used to check if the specified file exists.

Function: `evalSpice(filename)`

This is the main function that evaluates the circuit from the SPICE file provided.

1. File Parsing

The function first checks if the specified file exists. If not, it raises a `FileNotFoundError`. Inside the `parse_circuit_file()` function, the file is read line by line, ignoring comments (anything after #). It extracts lines between the `.circuit` and `.end` markers, which describe the circuit. If the file is malformed (missing `.circuit` or `.end`), an error is raised.

2. Circuit Parsing

The circuit components are stored in a dictionary `dct`, where the key is the component name (e.g., `R1`, `V1`, `I1`), and the value is a list containing the component value and the connected nodes. A utility function `is_convertible_to_float()` checks if a value can be converted to a float, ensuring that only valid numeric values are processed.

3. Component Validation

The program ensures only voltage sources (`V`), current sources (`I`), and resistors (`R`) are allowed. If any other components are found, an error is raised.

4. Node Identification

The set of nodes is determined, including the ground (`GND`).

5. Matrix Generation

The `matrix_generation()` function constructs the nodal admittance matrix (A) and the source vector (B) based on the circuit description:

- For voltage sources, it updates the B matrix with the voltage values and handles polarity.
- For resistors, it updates the admittance matrix using conductance values (1/Resistance).
- It also handles the case where current sources are present, updating the source vector accordingly.

The function ensures voltage sources aren't in a loop, which would result in an unsolvable circuit.

6. Solving the System

Using `numpy.linalg.solve`, the nodal equations (Ax = B) are solved to obtain the nodal voltages. The ground node is always assigned a voltage of 0.

7. Current Calculation

For each resistor, the current is computed using Ohm's law (V = IR), and the current through voltage sources is determined using Kirchhoff's Current Law (KCL).

8. Result Construction

The final nodal voltages and currents through voltage sources are stored in dictionaries and returned:

- `result`: Contains the nodal voltages, including ground (`GND` set to 0).
- `voltage_current`: Contains the current through each voltage source.

Custom Test Cases

- `test1.ckt`: parallel connection of voltage sources (raises error)
- `test2.ckt`: has a missing voltage value (raises error)
- `test3.ckt`: has resistor values (raises error)
- `test4.ckt`: missing voltage sources (raises error)
- `test5.ckt`: contains loops of voltage sources (raises error)
- `test6.ckt`: contains inductors (invalid elements) (raises error)
- `test7.ckt`: valid circuit (reference 1) (gives the expected output)
- `test8.ckt`: open circuit/1 voltage source (gives expected output)
- `test9.ckt`: single current source (raises error)
- `test10.ckt`: multiple voltage sources in the same branch (gives the expected output)
- `test11.ckt`: same as the former except the polarity of one voltage source has been reversed (gives the expected output)
- `test12.ckt`: voltage sources with resistors in each branch. testing KCL implementation (gives the expected output)