README

Dhruv Prasad (EE23B130)

September 8, 2024

Introduction

This is a report containing explanation of the working of my submission evalSpice.py that solves circuits with resistances, dc current sources, and dc voltage sources.
The code parses a .ckt file containing a list of components and the nodes between which they are connected, and then solves the circuit using nodal analysis to find unknown node voltages and current through voltage sources.

Code Overview

The code consists of several functions that collectively perform the following tasks:

- Parse a .ckt file to extract components and their connections.
- Build an admittance matrix for the circuit. (Basically a system of linear equations)
- Solve the linear system to find node voltages and currents.

Code Explanation

The Python code is broken down into several functions that handle different components of the circuit:

- `increaseDimensions()`: This function increases the size of a square matrix by adding a new row and column initialized to 0.
- `checkMalformed()`: This function goes through the entire file to see if the circuit is 'valid'. I have elaborated in the error handling section.

- `handleNewNodes()`: Checks if new nodes are encountered and updates the unknowns dictionary. It also adjusts the size of the admittance and constants matrices.
- `handleResistance()`: Updates the admittance matrix for resistive components.
- `handleISource()`: Updates the constants matrix for current sources.
- `handleVSource()`: Adds a new current unknown for a voltage source and updates the admittance matrix and constants matrix.
- `handleGNDEquation()`: Sets the ground node's voltage to zero.
- `evalSpice()`: The main function that reads the SPICE file, builds the admittance and constants matrices, and solves the circuit.

Explanation of Algorithm

The algorithm follows a systematic approach:

1. Checks if the file of given path exists and opens it. Raises an error otherwise.

2. Calls `checkMalformed` to see if the .ckt file is valid. Raises an error otherwise. The details about what is considered a malformed circuit is given in the error handling section.

3. Goes through the file line by line until a line starting with .circuit followed by whitespace is encountered.

4. Initializes an empty dictionary `unknowns`. This will contain names of nodes as given in the .ckt file mapped to indices.

5. Initializes a variable `no_of_unknowns` to 0.

6. Initializes an empty list of lists `admittance_matrix` and an empty list `constants_matrix`. This will contain the appropriate values such that `admittance_matrix` $\times$ `unknowns_matrix` = `constants_matrix`.

7. Here, the `unknowns_matrix` will contain the answers after solving the system of linear equations, indexed in order according to the `unknowns` dictionary.

8. Goes through the file line by line, checking whether the component is `R`, `V`, or `I`. Raises an error if any other component is present.

9. Calls the appropriate functions to handle new nodes and components encountered

10. For every new node, the key-value pair of the name and current number of unknowns is added to the `unknowns` dictionary. `no_of_unknowns` is incremented. Dimensions of `admittance_matrix` and `constants_matrix` are increased.

11. For every resistance R connected between nodes with indices i and j, 1/R is added to the diagonal elements and subtracted from the off diagonal elements corresponding to i and j in `admittance_matrix`. This is in accordance with coefficients when we write KCL equations at those nodes

12. For every current source I connected between nodes with indices i and j, I is added or subtracted to the elements corresponding to i and j in `constants_matrix`. This is in accordance with coefficients when we write KCL equations at those nodes

13. For every voltage source V connected between nodes with indices i and j, a key-value pair of the name of the voltage source and current number of unknowns is added to the `unknowns` dictionary. `no_of_unknowns` is incremented. Values are set to 1 and -1 in `admittance_matrix`, and V in `constants_matrix` such that Vi - Vj = V.

14. When a line starting with .end followed by whitespace is encountered, it breaks out of the loop and calls a function to set V GND = 0. *Note: I initially treat GND as an unknown so that I can minimize if statements while parsing the file.*

15. Converts the matrices from list of lists to numpy arrays.

16. Calls the numpy.linalg.solve() function to find the unknowns. If the circuit has no solution, it raises an error.

17. Sorts the obtained unknowns into V and I dictionaries based using their keys from unknowns matrix. Returns the two dictionaries

Error Handling

The script performs error checks for:

1. Invalid file.
   Raises: FileNotFoundError
   Raise Condition: The file of given path does not exist.

2. Malformed circuit
   Raises: ValueError
   Raise Condition:

   - If no line begins with .circuit
   - If no line begins with .end
   - If the .end line doesn't come after .begin *Note: There can be any junk in these lines following these markers*
   - If V, I or R has incomplete node or component data, i.e. there aren't enough words and list index goes out of range.

- If V or I is present in the circuit and the 4th and 5th space separated words are not 'dc' and a number. *Note: There can be any junk in these lines afterwards*
- If R is present in the circuit and the 4th space separated word is not a non zero positive number. *Note: There can be any junk in these lines afterwards*

3. Invalid element
   Raises: ValueError
   Raise Condition: In a line between .circuit and .end markers, the first non whitespace character is not V, I or R.

4. Circuits with no solutions.
   Raises: ValueError
   Raise Condition: When only current sources are connected to a node or voltage sources are connected in parallel

Note: 'dc', '.circuit', '.end', 'V', 'I', 'R' are all case sensitive