A2-SPICE

ABHINAV I S EE23B002

Implementation Algorithm

- The function `evalSpice()` takes a file, that contains a circuit in a spice netlist.
- The circuit is then parsed into a graph datastructure using the function `parseCircuit()`

`parseCircuit` Function

- We define a variable `circuit_graph` which is a dictionary where:
  - each node is a key, which corresponds to a tuple having the node to which it is connected, and the component in between.
  - nodes are strings and the component is stored internally as a string in the netlist fromat
  - eg: `V1  1 GND  dc 2`
- the function then return the `circuit_graph` dictionary.

`genEquations` Function:

The `genEquations` function does most of the work in this program. - It creates - A `coeff_matrix` which is a numpy array, which stores the coefficients of the system of linear equations. - A `const_matrix` which stores the constants in the system of linear equations. - A `nodeList` which is a list of nodes in the circuit - A `vsourceList` which is a dictionary containing voltage sources, where keys are the names of the nodes themselves, and the values is another dict containing metadata about the voltage sources like the index, current node, destination node, and ac or dc. - It assumes there is always a node named `GND` in the circuit

- It generates the system of linear equations through an iterative approach as described below:
  1. We consider each node(key) in the `circuit_graph` dictionary, and obtain the component, and the destNode.
  2. Then we check if each component is either a Vsource, Isource, or a resistor
  3. If it is a voltage source :
     1. Add it to `vsourceList`, add if its not.
     2. If its not part of `vsourceList`, increase the dimensions of the solution matrices by 1 (an extra variable of current through that voltage source has been added.)
     3. We check the polarity of the voltage sources, and add the current in the appropriate index (in the KCL equation for the nodes at which the voltage sources is connected). For example if the

voltage source is connected at the node, check whether the current is going out of the node, we update as :

`coeff_matrix[n1][nodeNum+1] = 1`

or -1 in the other case.

4. If it is a Resistor:
    1. Add 1/R to the current node and -1/R to the destination node in the coefficient matrix.
    2. Ignore if the destination node is `GND`
    3. If resistance is 0, raise `ValueError`. For example if the resistor is connected at n1(index 0), n2(index 1) and has value R , we update as:

`coeff_matrx[0][0] += 1/R`

`coeff_matrix[]`

5. If it is a current source:
    1. Add the value of the current source in the constant matrix for the corresponding node equation
6. Now for each voltage source in `vsourceList` we need to add a separate equation
    1. We check each node, and add +1 or -1 to the coefficient matrix according to the correct polarity in which the voltage source is connected and add the value of the voltage source itself in the constant matrix.
7. It then returns `coeff_matrix`, `const_matrx`, `vsourceList`, `nodeList`

`evalSpice` function

- This is the main function that runs the program
- It creates the `circuit_graph` using `parseCircuit`
- Then generates the matrices, voltage lists, and node lists using `genEquations`
- It then calls `numpy.linalg.solve()` to generate solutions, and then constructs the solution dictionaries.

Special tests / Border cases

Handling circuit without GND

- raises `ValueError`

Testing circuit with a voltage source of 0 Volts

```
.circuit
V0 1 GND dc 5
R1 1 2 1
```

```
R2 2 GND 1
V1 2 GND dc 0
.end
```

- Treats it as a short circuit as expected.

Testing circuit with a current source and open circuit

```
.circuit
I0 GND 1 dc 1
R1 1 2 1
.end
```

- Throws an error as expected `ValueError: Circuit error: no solution`

Testing an open circuit.

```
.circuit
V0 1 GND dc 2
R1 1 2 1
R2 GND 3 1
.end
```

- Circuit behaves as expected with voltage of voltage sources at appropriate nodes and GND Voltage at others

Connecting two Resistances only

```
.circuit
R1 GND n2 5
R2 n2 GND 5
.end
```

- As expected n2 voltage is 0.

Having Negative Resistance Values

- Raises `ValueError`

Connecting both terminals of a component to the same node - The component is ignored if its a resistor - Raises `ValueError: Circuit error: no solution` if it is a Voltage Source as expected. -

References

- Numpy Documentation