SPICE simulation in Python

Using gaussian elimination to solve spice circuits

Functions used in the code:
There have been different functions defined in the code, which have been mentioned below with the functionalities that they execute.

1. evalSpice(file_name):

This function takes the name of a SPICE file as input, reads the file, parses the circuit information, and computes the node voltages.
It initializes dictionaries (elements, resistors, voltages, currents) to store information about different circuit elements (resistors, voltage sources, current sources) and a list (node_list) to store all unique node names.
It reads the input file line by line, skipping lines until it finds the .circuit keyword.
It parses the circuit elements and stores them in the appropriate dictionaries (resistors, voltages, currents).
It then creates a dictionary (dict_of_nodes) to map node names to their indices and calls the conductance_matrix function to create the conductance and current matrices.
Finally, it solves for the node voltages using NumPy's linear algebra solver (np.linalg.solve) and returns a dictionary with node voltages (Node_V) and a dictionary with voltage source currents (i_dict).

2. conductance_matrix(num_nodes, resistors, voltages, currents, nodenum):

This function creates the conductance and current matrices used for circuit analysis.
It initializes matrices for conductance (conductance) and current (current_matrix).
It processes resistors to populate the conductance matrix.
It processes voltage sources to update both the conductance matrix and the current matrix.
It processes current sources to update the current matrix.
Finally, it returns the conductance and current matrices.

3. create_matrices(num_nodes, resistances, voltage_sources, current_sources, node_indices):

This function appears to be an alternative version of conductance_matrix. It takes similar inputs but follows a different approach to create the conductance matrix (G) and the right-hand side vector (right_vector).

Collaborations:

I had tried writing a piece of code for this task by following the 'Modified Nodal Analysis' method, but had errors in that and due to lack of expertise in coding, I found this assignment to be really difficult and had spoken to sir about it the last class that I was not able to do so and had difficulty

in doing so. I tried to check what was incorrect but could not conclude much, while trying myseld and having had asked some of the good coders in class.

I was unable to figure much about the code, so I asked EE22B151(Varinder) for help and he explained me how to go about it and helped me in the syntax also. He explained the algorithm that he went about and though the reasoning I had and the approach in mind was similar, I had trouble dealing with syntaxes mostly and he helped write each block of codes. A lot of it was also helped by EE22B090 and EE22B144. EE22B090 mostly helped with a majority of debugging.