Assignment 2 - SPICE simulation

valid_file()

The valid_file() function helps to identify if the filename is valid and throws a `FileNotFoundError` if in case the file doesn't exist.

file_check()

The file_check() function takes the file input and converts lines of it as a list and checks for first occurence of `.circuit`. Once it is found, `found_circuit flag` becomes True after which each line is split by whitespace characters and read. However presence of another '.circuit' in between would mean starting another netlist within the initial netlist got over and raises an error. The netlist stops at occurence of `.end`. For lines between '.circuit' and '.end' the line is split by whitespace characters and first character is checked. Only if its `'I','R' or 'V'`, it is further checked if `dc` is the 4th element if its a Voltage or Current source and if 5th element it could be a `valid numeric`. Similarly for resistance, 4th value is tested if numeric.(Absence of 'dc' throws an error as the circuit is solved only for dc voltages and 'V','I','R' elements)

The `dictionary mapping` of nodes to names and node-pair to value is updated and the presence of `voltage sources in parallel` is raised with error. The current sources in parallel are added up. Similarly resistance equivalent across node-pair is computed. The `repetition of an element name` (say, R1 in two different lines) raises an error. The presence of `less than 4 elements`(name, node1, node2,value) and dc in case of voltage or current source raises error. Presence of `extra characters apart from comments` also raises error. Presence of `zero resistance` is ignored as division by zero could later raise an error though circuit is valid.

readfile()

readfile() takes filename as input, calls the `valid_file()` to check if filename is valid. Then it opens the file and converts to file object. Then lines seperated by \n are stored into lines which are passed to `file_check()` which returns `number of nodes` in circuit. It also creates dictionary mapping of `node to integers` starting from 0, a mapping of `node-pair to value` of element and its type('V','I','R') and node-pair to name of voltage source.

non_supernode()

non_supernode() computes the `nodal analysis` equations for nodes not connected to voltage sources. It puts `current entering node` from current sources in `matrix, B` and `inverse of resistances` into A(or its negative according to whether it's the node under consideration). (For example, if node,1 is connected to nodes 2 and 3 via R1 and R3 respectively and I1 enters the node n1, current entering is I1. This is equal to current leaving the node,i.e,(V1-V2)/R1 + (V1-V3)/R3)=V1((1/R1)+(1/R3))+V2(-1/R1)+V3(-1/R3)

supernode()

For nodes connected to voltage sources, first form equations of form `V1-V2=V` where V1 is the positive terminal, V2 is the negative terminal and V is the value of voltage source. Then form supernodal equations where current entering the supernode(2 nodes across which volatage source is connected) is equal to current leaving. This is continued until as many equations as the number of nodes is reached.

V_InotinV()

Solve for voltage on all nodes considering `GND` voltage as `zero`. LinAlg error is raised if matrix has no solution which occurs if current sources in a loop are of varying values (Circuit error is reported). Circuit error is not raised in case a node is disconnected from rest of circuit(connected to only 1 element) considering `disconnected element` to have `no current` through it. Then current across nodes where there are no voltage sources is computed.

I_create()

The total current across node pairs where voltage sources are connected is found by first taking that node to which no more than one voltage source is connected and find current entering or leaving the node. Then eliminate it and work on the rest. Here, a copy of voltage mapping is created as it is more sensible to modify a copy of voltage source mapping rather than tampering with the original dictionary

An error is raised in case all nodes are connected to atleast 2 voltage sources (which means there is no change in dictionary) and thus it has a `loop of voltage sources` where curent through each voltage source can't be computed

IinVsources()

Initially number of voltage sources each node is connected to is passed to I_create(). It then returns total current across a node pair. Current through voltage sources is found by subtracting the currents through resistances or current sources from that of entire node-pair current. The dictionary (of volatge sources) can be iterated in nested loops in a particular order by sorting keys and iterating on them. The node voltages are mapped to node names using dict1 mapping. Simlarly dictV is used to find node-pair mapped to name of voltage source.

evalSpice()

The function evalSpice() takes filename and reads it to dictionaries. Then conductance, current and volatge matrices are formed that are filled using non_supernode() and supernode(). Then V_InotinV solves the matrices to give voltage at nodes and current across non-supernodes. Then IinVsources() calls I_create() (which computes the total current matrix) and then finds current across voltage sources. evalSpice() then returns dictionary of node names to voltages and volage source names to current across them.(I am finding the entire current across all node pairs according to initial problem statement and modifying it for I across voltage sources)

Extra test cases

Extra test cases are included testing cases like multiple voltage sources attached to a single node,non-planar circuits involving current and voltage sources and so on.