SPICE simulation in Python

This assignment asked us to solve purely resistive networks (without any dependant sources) described in the SPICE format.

Usage

```
from evalSpice import evalSpice
node_voltages, source_currents = evalSpice("path/to/file.ckt")
```

```
py evalSpice.py "path/to/file.ckt"
```

Grammar

The circuit file provided must follow the following grammar while defining the circuit

```
# Comments are defined by `#`

Any lines before the circuit definition are ignored

.circuit # The circuit definition begins with `.circuit`
    # The following lines must describe the branches in the circuit
    # NOTE: All names are supposed to be alphanumeric (`_` is also allowed)
    # NOTE: `GND` is a special node that acts as a reference (V(GND) = 0)
    # NOTE: Everything is case sensitive (i.e `GND != gND` etc.)
    # NOTE: All branches must have a unique name

    # Voltage sources are defined as follows
    # Vx<alnum> nodeA<alnum> nodeB<alnum> type<'ac' | 'dc> volts<float>
    V1 n1 n2 dc 1e-3
    # NOTE: Here V(n1) - V(n2) = 1e-3 Volts

    # Current sources are defined as follows
    # Ix<alnum> nodeA<alnum> nodeB<alnum> type<'ac' | 'dc> amps<float>
    I1 n1 n2 dc 1.2
    # NOTE: Here current flowing from n1 to n2 is 1.2 Amps

    # Resistors are defined as follows
    # Rx<alnum> nodeA<alnum> nodeB<alnum> ohms<float>
    R1 n1 n2 1

.end # The circuit definition must end with `.end`

Any lines after the circuit definition are ignored
```

Implementation Details

To understand the exact specifics of how everything is done just read the code lmao (It is ~~well~~ documented).

But here is the gist of it:

Consider the following ckt file

```
.circuit
    V1 1   GND dc 2
    R1 1   2      0
    R2 2   GND    1
.end
```

1. This file first parsed into partial branch data

   (source type is discarded, since we don't handle circuits with multiple types of sources anyways)

   ```
   branch_data: dict[str, PartialBranch] = {
       "V1": (
           ("1", "GND"), Element.VOLTAGE_SOURCE,
           2.0
       ),
       "R1": (
           ("1", "2"), Element.RESISTOR,
           0.0
       ),
       "R2": (
           ("2", "GND"), Element.RESISTOR,
           1.0
       )
   }
   ```

2. The nodes are the assigned indices, branches now refer to these instead of the node names. Resistors with 0 resistance are converted into voltage sources of 0 volts in this stage.

   ```
   nodes: list[str] = ["1", "GND", "2"]
   branch_data: dict[str, Branch] = {
       "V1": (
           (0, 1), Element.VOLTAGE_SOURCE,
           2.0
       ),
       "R1": (
           (0, 2), Element.VOLTAGE_SOURCE,
           0.0
       ),
       "R2": (
           (2, 1), Element.RESISTOR,
   ```

```
        1.0
    )
}
```

3. A system of equations is now formed to solve for the unknowns.

```
# For Gx = y
G = [
    [ 0.0,  0.0,  0.0,  1.0,  1.0],
    [ 0.0,  1.0,  0.0,  0.0,  0.0],
    [ 0.0,  0.0,  1.0,  0.0, -1.0],
    [ 1.0,  0.0,  0.0,  0.0,  0.0],
    [ 1.0,  0.0, -1.0,  0.0,  0.0],
]

y = [
    0.0,
    0.0,
    0.0,
    2.0,
    0.0
]

"""
x = [
    V("1"),
    V("GND"),
    V("2"),
    I("V1"),
    I("R1") # since R1 was converted into a voltage source, this also is
solved for
]
"""
```

4. Finally the required data is extracted and returned as 2 dictionaries (The current flowing through 0 resistance resistor isn't reported even though it was computed (based on my interpretation of the problem statement this made sense))

```
node_voltages = {
    "1": 2.0,
    "GND": 0.0,
    "2": 2.0
}
source_currents = {
    "V1": -2.0
}
```

Edge Cases to Consider

Resistances of 0 resistance (wires):

Since we add terms like 1/R to the coefficient matrix, special care must be taken while handling R = 0 (i.e just a plain wire).

I've handled this by treated such resistances as voltage sources of V = 0. This was the easiest way to update the coefficient matrix correctly with the implementation I had going already.

(As an added bonus you get to solve for current in the wire also. But I don't report this value as the problem statement didn't ask for it)

The test cases `extra/wire1.ckt` and `extra/wire2.ckt` test these out

Multiple wires between the same 2 nodes:

This problem arises due to how I handle wires (0 resistance resistors).

Consider 2 wires from node1 to node2 in parallel. I would then treat these as 2 voltage sources and make corresponding equations. But notice that we can't actually solve for the current going through these *pseudo* voltage sources. Therefore the equations would not produce a solution. But notice that in the original circuit, we were never required to sovle for current in this branch.

I've handled this by simply removing duplicate wires (max only 1 wire between any 2 nodes). This fixes the afformentioned issue.

Now an argument can be made whether the circuit is *really* solvable. It is true that we can never find the individual current through the 2 wires and thereforce the circuit technically can't be solved. But since the problem statement just wanted node voltages and voltage source currents (which are solvable) I argue that a circuit error must not be raised here. (I do print out a warning though)

The test case `extra/wire3.ckt` tests this case

Empty circuit:

Techincally an empty circuit is already solved :)

(Even though there is no GND for reference)

The test case `extra/empty.ckt` tests this case

Errors and Warnings

I print out many ~~annoying~~ useful warnings on some cases where I thought that user might need some clarifications. (Give some bad inputs to check out what all warnings I give. ~~Alternatively just read the code bro~~). Set `SHOW_WARNINGS` to `False` to disable warnings.

The following are a few errors and warnings raised

- If the file doesn't exist a `FileNotFoundError` is raised
- If the circuit file is malformed (invalid format), I print out a warning stating what exactly is wrong (and the line no in which the issue was faced). Then a `ValueError` is raised

- If the circuit contains sources of both types (ac and dc), a warning is printed (As we only handle circuits that have the same type of source)
- If there is no ground node for reference a warning is printed out saying so and a `ValueError` is raised
- As mentioned in the special cases, a warning is printed when multiple wires exist between the same 2 nodes
- If the user provides an empty circuit a warning is printed
- If the circuit has no solution a `ValueError` is raised

References and Discussion

- A few of my friends and I had discussions on how to construct the coefficient matrix.
- This resource had a nice writeup on how to construct the coefficient matrix.