EEL4837 Programming for Electrical Engineers II – Spring 2025

Excursion 1: Circuit Analysis Tool

Due Date: 4:00 AM, Tuesday, March 11th, 2025

Submission Instructions

Submission should be through Gradescope. Only 1 team member per group is required to submit the assignment. The following files must be submitted **separately**:

1. One or several source code files for the circuit analysis tool. The code should deal with the following files:
    - Input: a file named ***netlist.txt*** located in the same directory as the source code. It sticks to the format "branch label | source node label | destination node label | value" as covered in lecture.
    - Output: a file named ***output.txt*** created by your team's code and located in the same folder as the program. The output format should be a single row similar to what is shown in the slides:
        - $e_1\ e_2 \ldots e_n\ v_1\ v_2 \ldots v_n\ i_1\ i_2 \ldots i_n$
        - That is, you write the potentials, voltages, and currents with *one space between them.*
        - Each potential/voltage/current is a floating-point number in a decimal notation with *3 decimal places* after the point (e.g., 14.654).
    - Note: to compile your team's code, we will be running the following command: "g++ *.cpp -o ex1" and we will call the generated "ex1" executable to test the code. In order for this to work, please ensure that all the necessary source (.cpp) and header (.h) files are located within the root directory of the submission. No specific file name is needed; as long as all source files end with a .cpp extension.
2. A ***Readme*** file, named "Readme", that that explains how to compile, execute, and run your team's code. This includes whether or not you are going for bonus points. Any readme that does not indicate if the bonus is included will not be considered.
3. (Optional) Additional documentation that you wish to provide.

Description

The goal is to implement an elementary circuit analysis tool that reads a circuit from a netlist and computes the voltage potential, voltage drop, and current of each component.

Input: a circuit netlist (***netlist.txt***)

Output: a file of calculation results (***output.txt***)

In this excursion, you can assume that:

1. The format of the input netlist is the same as that in the Excursion 1 slides.
2. The circuit only consists of voltage sources and resistors.
3. Your team's program will not be tested on mal-formed or inconsistent netlists.

Guidelines

Taking the Excursion 1 slides as a reference, you may create several functions for reading a new netlist, counting the number of branches and nodes, creating several matrices, calculating all the parameters, etc. Here are some general steps for your reference:

1. Read a new netlist: one of the easiest ways is to use filestream. Note that we will strictly use the format of "branch label | source node label | destination node label | value" as shown in class.
2. Count the number of branches and nodes: you need to take the input from the previous step.
3. Create an incidence matrix: you may use the method introduced in class to derive a 2-D incidence matrix from a netlist.
4. Create a voltage coefficients matrix.
5. Create a current coefficients matrix.
6. Create a combined matrix.
7. Append the input to the combined matrix.
8. Calculate all the parameters (e.g., using Gaussian elimination, LU factorization, or matrix inversion).

I/O Example

Input (***netlist.txt***):

V1 1 0 5

R1 1 2 10

R2 2 0 20

Output (***output.txt***):

5 3.333 5 1.667 3.333 -0.167 0.167 0.167

Grading

Total points: 100

1. Automated grading (75 pts)
    a. Correctness (60 pts): correct output values on well-formed, “legal” inputs.
    b. Robustness (15 pts): correct output on very simple and very complex “legal” inputs. Should act reasonable on malformed inputs.
2. Manual grading (25 pts)
    a. Efficiency: the code is reasonable and not brute-force. No particular time/space complexity is expected. (10 pts)
    b. Elegance: clean code style, readable comments as needed, meaningful functions, etc. (10 pts)
    c. Documentation: Includes a readme on details about the code and whether bonus should be considered. (5 pts)
3. Bonus points (20 pts): create and use a data structure for sparse matrices (as shown in the slides). The combined matrix needs to be converted to this format and processed using it during the Gaussian elimination (or another algorithm of choice). Smaller matrices do not have to use this data structure.

Note

1. All code should be in C++ only. We encourage splitting up code into multiple files/ functions/classes.
2. Your team’s program should compile and run successfully on both the ECE Linux server and Gradescope.
3. You can only use C++ standard libraries (including STL). Do **not** use libraries that implement matrices for you. If you are not sure whether or not some libraries are allowed, please reach out to the TA or instructor for help.

Helpful Tips and Tricks

Here are some helpful links to understand the math you will need to accomplish:

https://youtu.be/eDb6iugi6Uk - Row echelon form

https://youtu.be/0gRtVz4XrZM - Kirchoff’s laws

https://www.youtube.com/watch?v=EaHFhms_Shw - read/write text files

- All circuits are closed with finite resistances.
- The number of nodes may be any positive number (greater than one digit, have decimal places).
- Only one source file is needed, however you may use multiple if desired.
- Do not upload a zip file/ folder.

- You may encounter “Segmentation fault”. This can be due to a variety of reasons, and as such, we suggest printing (cout) to check where the error occurs.
- Check out invertible matrices and finding the solution using the inverse.
- You may use prior code from homeworks.
- When testing code with your own netlist.txt, do not include empty lines.
- Be careful using certain, more unique, libraries: <iomanip>, as they have been known to cause errors.