ELEC-A7151 - Object oriented programming with C++

Circuit simulator project documentation

Group: circuit-sim-2020-2

Vincent Eurasto, 711690
Tuukka Jaakkola, 608923
Perttu Niskanen, 460132
Ilari Ojakorpi, 609113

1. Overview

Our goal was to implement both steady state and transient analysis both for DC and AC into the circuit simulator -program. However we only managed to implement steady state analysis for DC and AC. The user can add, rotate, move, connect and remove components and wires. The user can also move and zoom the view and the user can build an electronic circuit from various components and sources by connecting them with wires. After the user has constructed a circuit he/she can solve the circuits node voltages and component currents. The program uses MNA (Modified Nodal Analysis) when solving the voltages and currents in the circuit. When calculating the currents and voltages with steady state DC analysis, inductors will appear as short circuits and capacitors will appear as open circuits for the solver.

One minus is that when solving the circuit using steady state AC analysis, the user can only choose one angular frequency for all sources. Thus the user must choose between AC and DC sources. This made the analysis much easier since we didn't have to use the superposition method when solving the circuit. After solving the circuit, node voltages and component current calculations will be printed into the console in complex notation. Our plan was also to implement graphical plots for visually showing current and voltage calculations, however we didn't have the time to do this. The user can load a circuit from a netlist file and the user can also save the currently constructed circuit into a netlist file. What comes to additional features, we managed to implement DC current sources and sinusoidal AC voltage and current sources.

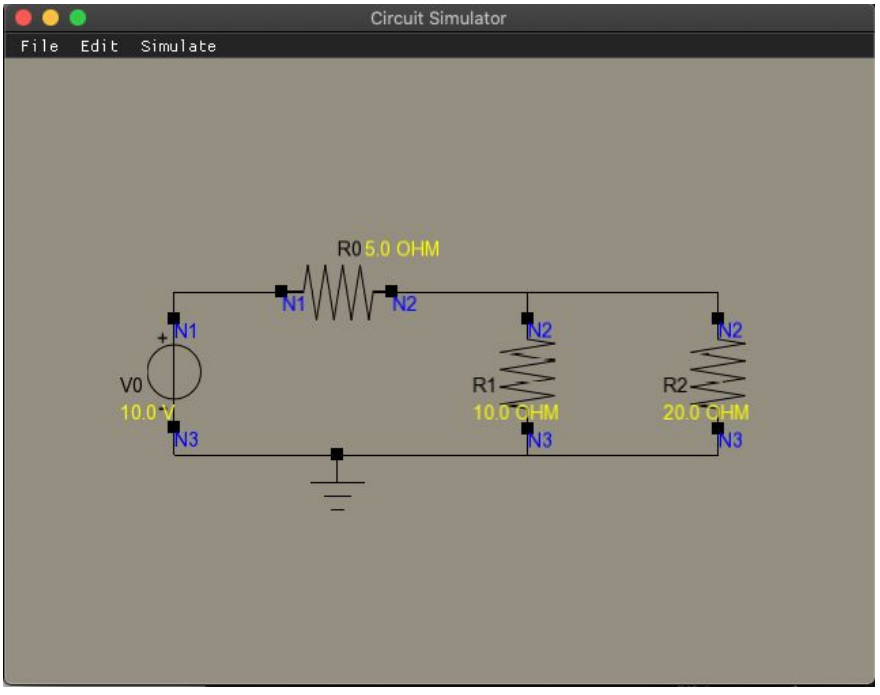

Figure 1. Preview of the program

We had some issues while implementing all functionalities correctly. Perhaps the most noticeable issue comes from parallel connected inductors; when solving/simulating the circuit with steady state DC analysis the matrices won't construct correctly or in other words the main matrix becomes singular. We tried to search the internet for answers to this kind of issue but without luck. This is not very big of an issue since usually DC analysis is not calculated with reactive elements. Regardless of this issue, inductors in series seem to work while performing steady state DC analysis. While performing steady state AC analysis, inductors work fine both in parallel and series.

2. Software structure

We used a graphical user interface library called IMGUI to produce a basic window and a graphics library called SFML to render graphics. To make our job easier when solving matrix equations we used a matrix manipulation library called Eigen. And finally we used a testing library called Doctest. We decided that it's easier to show the overall structure of the program by drawing an UML-diagram of the program than to explain everything verbally. The UML-diagram is presented below.

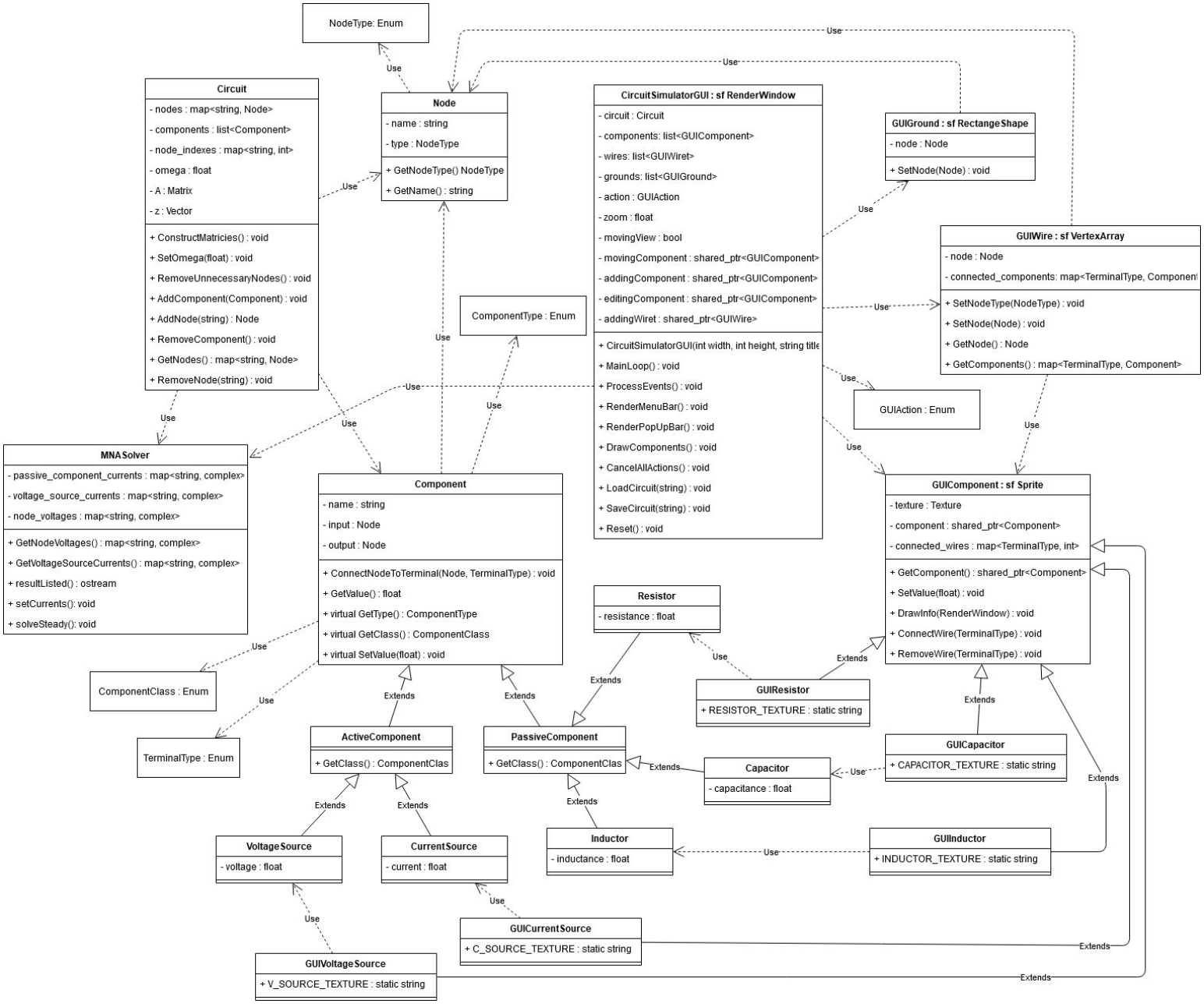

Figure 2. UML-diagram of the programs structure

3. Instructions for building and using the software

3.1 Compiling and running the program

At first the user must install the needed packages and submodules needed for the program. If using Linux, the needed packages can be installed running following commands in terminal:

```
sudo apt-get install build-essential
sudo apt install gcc
sudo apt install g++
sudo apt install make
sudo snap install cmake --classic
sudo apt-get install libx11-dev
sudo apt-get install xorg-dev
sudo apt-get install freeglut3-dev
sudo apt-get install libudev-dev
```

After downloading the package from git the user must install the submodules needed for the program. This means downloading Eigen which is a matrix manipulation library. This can be done by running following commands in terminal:

```
cd circuit-sim-2020-2
git submodule update --init --recursive
```

After downloading and installing the needed submodules the user can build and run the programming executing following commands in terminal:

```
cd circuit-sim-2020-2
mkdir build
cd build
cmake ..
make
./src/main
```

We also implemented a couple of automated tests. Tests can be run by typing the following commands into the terminal.

```
cd circuit-sim-2020-2
mkdir build
cd build
cmake ..
make
```

```
make test
```

The user can also remove everything from the build folder by executing one of the following commands.

```
rm -r build
rm -rf build (incase of error: sfml is write protected)
```

3.2 A basic user guide

The program can be run using the instructions described earlier in chapter 3.1. After building and running the program the user should see an empty window similar to the one below.

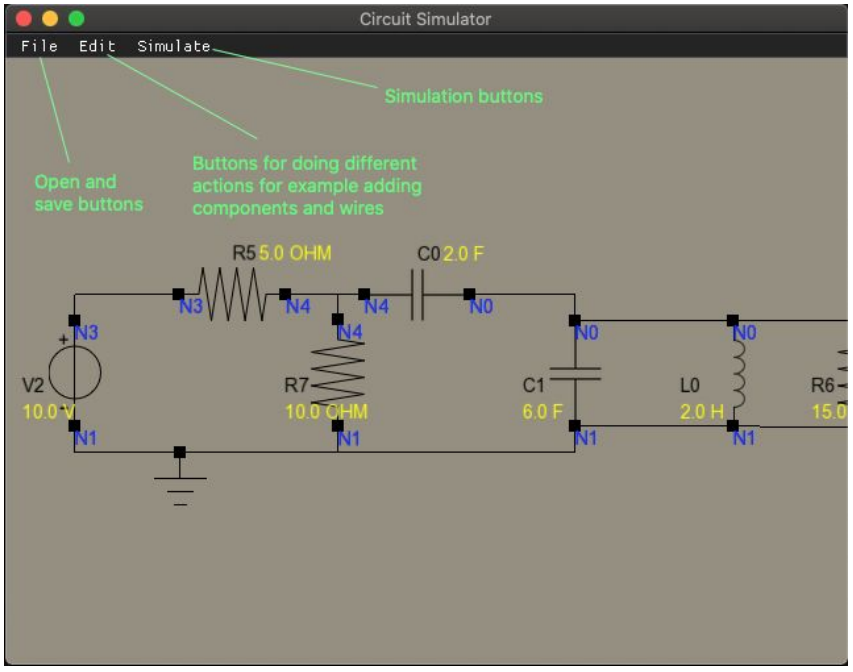

Figure 3. Basic simulator buttons

In the main menu bar, by selecting “File”, the user can open a circuit, save a circuit or exit the program. In the “Edit” submenu, the user can add different components, including resistors, inductors, capacitors and sources. The user can also choose between various actions: moving, rotating, deleting, wiring or grounding. The quick commands for each are listed next to each of the commands. If one action is chosen but the user wanted to do something else or wanted to cancel the action, it can be done by right-clicking anywhere in the circuit or by pressing the ESC-key. When components are added into the circuit view, the terminals of components should be connected in a way that there is at least one closed loop in order to allow the current to flow in the circuit. This can be done by selecting wire from the “Edit”-menu. Make

sure that the cursor is on top of a component's graphic when connecting components with wires. Otherwise the program won't notice that you meant to connect a component. If the connection is successful, a small black square should appear in the component's terminal. If some terminals are left unconnected, it will not affect the calculation results. After all connections are made a ground should be added into the circuit by selecting ground from "Edit"-submenu. Ground can be connected to any node and there must be at least one ground node in the circuit before running simulations. After the user is satisfied with the constructed circuit one can simulate the voltages and currents by selecting "Simulate"-button from the main menu bar. The user has two options to pick from: steady state DC or AC analysis. If the user picks steady state DC analysis the calculation results should appear in the console. If the user picks steady state AC analysis the program will prompt the user with a popup asking for an angular frequency. The angular frequency must be greater than zero, otherwise the circuit will be solved using steady state DC analysis. After this, the calculation results (node voltages and component currents) will be visible in the console in complex notation. There are a couple of example circuits available in the examples folder.

4. Testing

Testing was done with an automated test library called doctest which can be found in github: https://github.com/onqtam/doctest. These tests were mainly for "backend" features like circuit creation and calculations and allowed to keep track of effects of new code on old features. Perhaps the most logically challenging part of the program was to construct the MNA (modified nodal analysis) matrices correctly and that's why we thought that it would be a good thing to write a lot of tests for this functionality. We didn't have time to implement any tests for GUI features but this would also be important since we also had some complex graphic logic inside the GUI classes.

5. Work log

Our division of work is written in the table below.

Vincent Eurasto, 711690	Tuukka Jaakkola, 608923	Perttu Niskanen 460132	Ilari Ojakorpi 609113
GUI, MNA	MNA, EIGEN	GUI	MNA

Responsibilities table

5.1. Learning

Throughout the project, each of us learned various things regarding our work, not least of which is a newfound appreciation of LTSpice simulation program. We learned that implementing a GUI is deceptively difficult and that scouring through incomplete or unclear documentation for libraries gets very tiring very quickly. In addition, we dealt with various issues in coding on Windows through WSL, whereas coding on MacOS or Linux didn't present any issues other than our own bugs.

5.2. Individual worklogs

And our worklogs for each project member are presented in the following tables.

Vincent Eurasto, 711690

Week	**Description**	**Hours used**
26.10 - 1.11	Wrote base classes for active and passive components and nodes.	10
2.11 - 8.11	Integrated testing library (doctest) into the project. Added functionality to load circuits from files and save circuits into a file and wrote automated tests for these.	15
9.11 - 15.11	Integrated matrix manipulation library (Eigen) and GUI library (IMGUI) into the project. Added active and passive classes between components. Also started working on GUI; I wrote basic GUI-components (resistors, inductors & capacitors). Added a few texture sprites for components.	20
16.11 - 22.11	Continued working on GUI; added main menu bar with buttons, view zooming & moving. Added voltage source into gui. Implemented actions for adding, deleting, moving and rotation for GUI components.	40

	Wrote main class for GUI (because it was a function based approach before). Refactored main gui-loop (event polling & handling, component drawing etc.) into smaller methods. Added wire drawing and added function that determines which components terminal was clicked. Added grid snapping for components for easier alignment. Improved MNA matrix production algorithms to support voltage sources and reactive components. Added automated tests for matrix production.	
23.11 - 29.11	Worked again more on GUI: Users can now remove wires. I added keyboard shortcuts for different actions (component adding, deleting, moving, rotation). Integrated file browser into the project (https://github.com/gallickgunner/ImGui-Addons). Added fonts to project. I made component names visible to the user. Added helper lines for wire drawing and made other small improvements for easier component connecting. Started implementing component connect logic. Added functionality to change component values in the GUI.	30
30.11 - 6.12	Again GUI improvements; implemented circuit saving and loading from GUI. Added ground as GUI element. Connection marks are now visible for the user when the component is connected. Added visible node names.	15
7.12 - 11.12	Added current sources into GUI. Implemented inductor current calculations into steady state DC analysis algorithm. Removed unnecessary code and cleaned code. Added unit labels for GUI components. Combined calculation logic and GUI and added possibility to calculate steady state analysis (DC and AC). Added three example circuits for showing programs functionalities.	20

Tuukka Jaakkola, 608923

Week	**Description**	**Hours used**
26.10 - 1.11	Basic Git work, learning create branches	2
2.11 - 8.11	Integrated eigen library (locally) and implemented MNA solver solveSteady matrix calculation	10
9.11 - 15.11	Trying to figure out problem with wsl and sfml, took a lot of time, but I couldn't advance with project	15
16.11 - 22.11	Switched to VM and got sfml working with a help of Vincent	10
23.11 - 29.11	implemented setCurrents -method	10
30.11 - 6.12	implemented resultListed -method	6
7.12 - 11.12	Final fixes	6

Perttu Niskanen, 460132

Week	Description	Hours used
26.10. - 1.11.	Reading through documentation on Imgui and QT, I decided on trying to implement Imgui instead of QT.	8
2.11. - 8.11.	Tried to implement Imgui through recommended packages: GLFW and OpenGL3. Implementation required excessive dependencies that refused to work together. Essentially no progress made.	20
9.11. - 15.11.	Issues with WSL, tried to fix by using remote machines but lack of admin access made it impossible. Ordered a laptop to use linux on. Project supervisor provided an implementation of sfml that worked.	15
16.11. - 22.11.	Getting everything working on the new laptop. Fixed some quality of life issues and bugs, such as component rotation.	8
23.11. - 29.11.	Started work on moving wires with connected components. This proved much more difficult than expected.	10
30.11. - 6.12.	Worked on component connecting after Vincent expressed difficulties with it, but he finished his implementation of it before I did. Moved back to moving wires.	15
7.12. - 11.12.	Implementing the moving wires interrupted by sickness. Implemented some small quality of life changes. Bug hunting.	8

Ilari Ojakorpi, 609113

Week	Description	Hours used
44	Planning and reading on MNA	2
45	Implemented testing library with googletest, however this was changed to doctest implemented by Vincent. Basic circuit features.	15
46	Started matrix formation for circuit and made tests, learnt to use eigen.	10
47	Some work on matrix formation and tests.	5
48		
49	MNA node voltage calculations and tests.	30
50	Implemented AC voltage and current calculations and DC short circuit calculations for inductor and logic for parallel component currents.	25