Imperial College London

Department of Electrical and Electronic Engineering

ELEC40006: Electronics Design Project

Circuit Simulator Technical Report

Authors:

Xin Wang
CID: 01735253
xin.wang19@imperial.ac.uk

Brandon Cann
CID: 01724765
brandon.cann19@imperial.ac.uk

Adam Rehman
CID:
adam.rehman19@imperial.ac.uk

Submitted in partial fulfillment of the requirements for ELEC40006

June 4, 2020

Contents

1 Overview of the report

This report loosely follows the stages of the Software Developement Cycle.

Section 2 details the design problem presented and the program requirements at are necessary for a preliminary design to be established.

Section 3 provides a summary of the program development timeline and the details relating to project management format the format of meeting minutes used to how the responsibilities are distributed among the project team.

Section 4 discusses the preliminary designs produced by the team and the rationale behind some design choices implemented by the team.

Section 5 gives an comprehensive overview of the program design, detailing the functions and flowcharts used.

Section 6 investigates the design constraints of the program, the accuracy of the results produced and the speed of execution as the circuit size varies.

Section 7 serves as a continuation of Section 6, discussing the possible improvements to mitigate problems encountered during testing.

Section 8 details the possible features that can be added on to the existing program and the required modifications to the program to add the mentioned features.

2 Project Specification

2.1 Design Problem

Develop a program that is able to read in a file describing a circuit specified by the user, perform transient simulation on that circuit and output the calculated voltages at each instance in time into a file.

2.2 Program Requirements

The main program requirements are listed as follows:

- Program must support basic circuit components listed as follows [1]:
    - Resistors
    - Ideal Capacitors
    - Ideal Inductors
- Input file must adhere to SPICE netlist formats
- The output file must be in Comma Separated Value (.CSV) format.
    - Columns of output file represent nodes in the circuit.
    - Rows of output file represent an instance in the simulation.

[1]Advanced component can be supported provided development schedule is not constrained.

2.3 Design Criteria

The team has identified several factors from the list given in the Product Design Specification document.

- Maintenance: One of the most important factors the team identified. As the development timescale is constrained and the features required is open-ended, so it is important we ensure the design can incorporate new features efficiently and cost-effectively if the client wishes to add new features.

- Documentation: For a non-intuitive program, proper documentation is required for client and future programmers to make modifications should the client wish.

- Performance: No strict performance guidelines are given by the client but the execution speed should still be reasonable.

- Time scale: 6 weeks with a definite deadline. It is important to balance proper team management techniques and creating a program that meets client standards.

- Testing: Numerous test programs have been created to test various aspects of the program to ensure accurate results have been produced. The results produced have been verified with LTSpice, a well established circuit simulation program.

- Patents: SPICE engine is a public domain software so we could possibily utilise the engine.

- Ergonomics: A Graphics User Interface (GUI) is not specified by the client and could possibly make the program much easier to use.

3 Team Management

3.1 Project timeline overview

TIMELINE 1: *Final Project Timeline*

12 May	Team formed.
14 May	1st team meeting Tech Specifications published.
15 May	Research and design of preliminary designs.
16 May	First version of project management guidelines established.
20 May	2nd team meeting.
21 May	Final version of project management guidelines established.
29 May	Final program design agreed on.
3 June	3rd team meeting.
4 June	Final program design implemented.
6 June	Technical Specification published.
14 June	Project deadline

Documentation and minor improvements on codebase is a constant process throughout this process.

3.2 Gnatt chart

3.3 Management approach

The project team, with research on different forms of project management, decided on using a Waterfall project management approach. The phases are listed as follows:

- System and Software Requirements.
- Analysis.
- Design.
- Coding.
- Testing.

This method is selected mainly since the project is short, requirements are clear and the team is not constantly required to report to the client at the early stages of the software development to ensure client statisfaction.

The Waterfall methodology prioritises proper documentation throughout the whole development process. This is important to the team as the team project requires a Project Report to be submitted and, due to the ongoing remote lab orals and family-related obligations during quarantine, allows a team member to be quickly caught up should a member step out of the team for a while.

The disadvantage of the Waterfall methodology lies mainly in that no working software is produced until late in the cycle so there is a level of risk and uncertainty with not being able to meet the deadline, especially in the current state of the world [2]. The project team later realised the methodology is not exactly suitable for the Object Orientated Approach that the team eventually settled on. As the integration is done at the end, there is a possibility that any errors in the program is not caught during integration.

[2]Any unforseen circumstances are listed in team meeting minutes summarised the following sections.

3.4 Team responsibilities breakdown

The responsibilities of each team member is discussed during the 1st meeting and formalised in the 2nd meeting. The respective roles are partly determined by the Belbin questionnaire provided by Mrs. Perea [3]

The main responsibilities of each team member is listed as follows:

- Adam Rehman
  - Plant
    - Research program.
    - Coding the program.
    - Testing program.
- Brandon Cann
  - Monitor Evaluator
    - Ensure documentation managed by Xin Wang and coding managed by Adam is in sync.
    - Contributes to program aspects that is falling behind.
    - Ensure project requirements are met.
- Xin Wang
  - Implementer
    - Creating and managing the team Project Report and various other documentations.
    - Meeting minute keeper.
    - Ensuring code written is up to a standard format with proper comments.
    - Manages repository base.

[3] Belbin questionnaire forms are found in Appendix: Belbin Roles.

3.5 Project meeting minutes

The meeting minutes follow a standard template that is attached under Appendix: Meeting Minutes.

There is no standard protocol for calling team meetings, team meeting are called usually when there is a problem that is applicable to all team members. Usually, there is informal communication across various channels from WhatsApp to Discord.

Over the course of the development cycle, there has been three formal meetings and the main reasons and conclusions are listed below.

- Meeting 1
  - Reason:
    - Project team meeting each other. Not every team member knows each other.
    - Discuss the Circuit Simulator briefing.
  - Conclusions:
    - Xin Wang was assigned as documentation manager and tasked with formalising the Project Requirement.
- Meeting 2
  - Reason:
    - With a draft of the project management guidelines, team discussed the team dynamics.
    - Discussed general direction of our program design.
  - Conclusion:
    - Agreed that practicing proper team management is just as important as showing practical skills in coding.
    - Final team management guidelines are agreed and tasked to Xin Wang to formalise in documentation.

- Meeting 3
  - Reason:
    - Program basic requirements are nearly fulfilled.
    - Meeting to discuss future program design direction.
    - Xin Wang has a notification by South African Embassy of a Repartiation flight.
  - Conclusion:
    - Plans set to divide up Xin's responsibilities among the other two team members.
    - Documentation finalised and notes written by Xin passed to team members.

4 Preliminary Designs

Our initial program design relied on the project briefing provided by Dr.Stott. Parsing a input file into a data structure is a standard procedure. The main point of discussion is related to how the program solves for unknown voltages.

The main design choices taken by the team has been set out below.

4.1 Object Orientated Programming

As the number of circuit components that we need to support is open ended besides the mandatory basic components according to the Project Breifing, the team was concerned on the ease of adding a new features or components in the future.

The team initially created a simple program based on creating nodal equations and using matrices to solve for nodal voltages. The codebase soon became unmaintainable as any errors in communication results in the program crashing. During the 2nd team meeting, the team is agreed that an Object Orientated Programming (OOP) is necessary. The final program architecture was then designed OOP as the central factor affecting our other decisions. The design is detailed in the next section.

The OOP approach allows the program design to be rapidly and cost-effectively altered to accomodate new features such as additional circuit components or analysis techniques [2] that the client might request later on in the program's lifecycle. This approach saves the programmer and the client cost and time in program maintainability.

The disadvantage to that approach is that it has shown to be slower than other preliminary programs designed without OOP to an average of of 5% to 10% slower depending on other optimisations in the program.

4.2 Modified Nodal Analysis

Modified Nodal Analysis (MNA) is an extension of normal Nodal Analysis. When the team start researching similar programs like PSPICE and LT-Spice, MNA was mentioned numerous times [1] and its benefits was clearly seen when we tried to write a program that implemented normal Nodal Analysis. A problem the team encountered was trying to represent the current-dependent circuit elements like inductors efficiently.

As stated earlier, a primary concern of the team lies in the scalability of the program. By using MNA, just like other commercial programs, the team is confident that the analysis component of the program will not be a bottleneck to future developments of the program.

4.3 Eigen Matrix Library

In terms of matrix operations, the team decided to use a third-party program to handle the matrix-related operations. The primary concern with choosing a third-party program was how seamless the program integrates into our program. The team does not know whether the client would be comfortable installing an unknown third-party program on their system besides the program that the client specifically requested for.

There are third-party programs that require installation such as Armadillo and some has a large file size such as the Boost Library which is close to 1GB. Based on the concerns laid out, the team looked for a program that has a reasonable size, does not require installation in order to be used. The third-party program chosen was Eigen. It is a file that has a reasonable size and only requires that the Eigen is located in the same directory and the execution scripts we provided is used.

5 Program Design

5.1 Top level view of the program

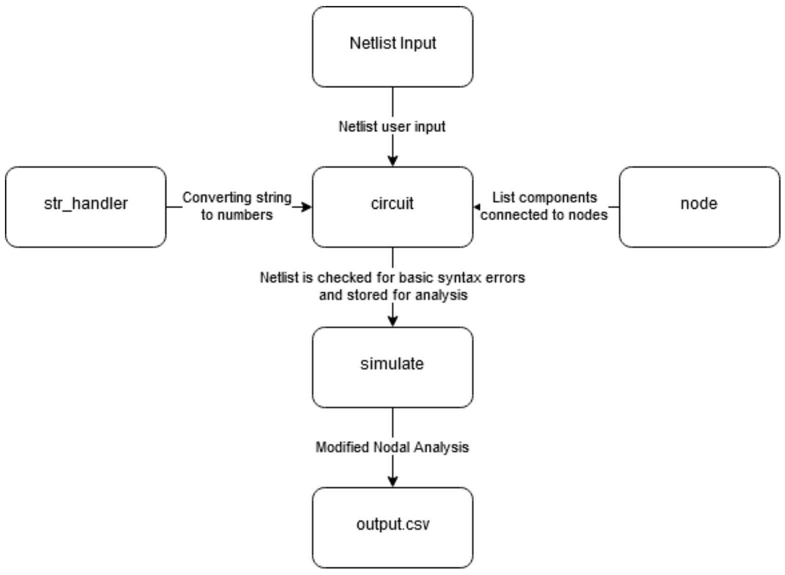

Netlist Input
Netlist user input
str_handler
Converting string to numbers
circuit
List components connected to nodes
node
Netlist is checked for basic syntax errors and stored for analysis
simulate
Modified Nodal Analysis
output.csv

Figure 1: General flowchart of program

circuit file forms the central section of the program, handling the input of the file and storing it in a form that allows the *simulate* file to perform Transient Analysis on it and outputs the file in the format (.CSV) specified by the client.

The theoretical information and program design for each of the main files are discussed in the following sections.

5.2 Components files

Due to the OOP design methodology, each component and its necessary functions are described in its own respective file. The file *edge.hpp* is the base from which all other circuit components are derived from the **components** folder.

5.3 *str handler.hpp* file

The *str handler* file manages all string-related functions. The main function is the

5.4 *node.hpp* file

As a circuit can be expressed as a Network Graph, containing branches and edges

5.5 Circuit.cpp

5.6 Simulate.cpp

6 Testing

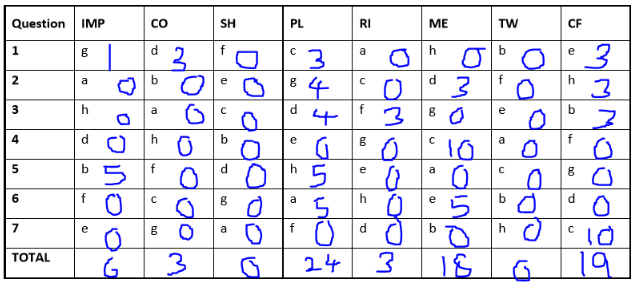
7 Optimisations

7.1 Sparse Matrices

8 Adding on

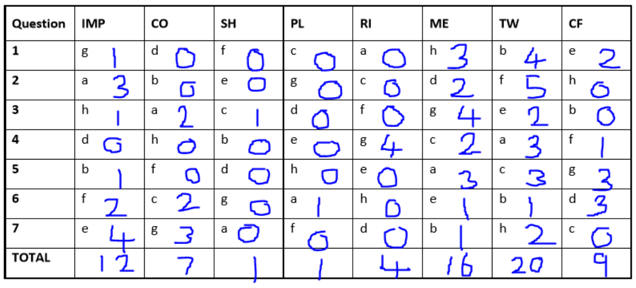
9 Appendix

9.1 Belbin Roles



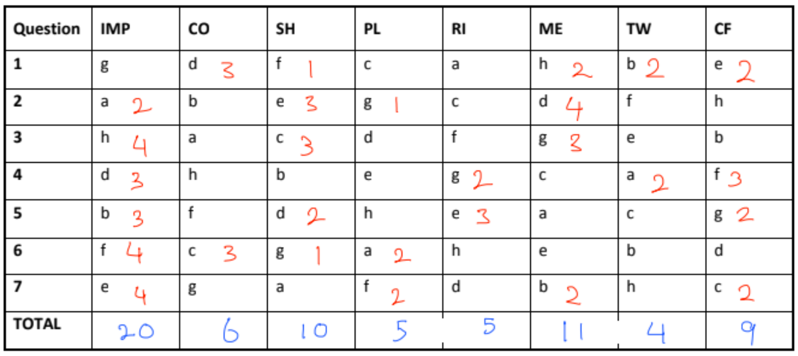
Question	IMP	CO	SH	PL	RI	ME	TW	CF
1	g 1	d 3	f 0	c 3	a 0	h 0	b 0	e 3
2	a 0	b 0	e 0	g 4	c 0	d 3	f 0	h 3
3	h 0	a 0	c 0	d 4	f 3	g 0	e 0	b 3
4	d 0	h 0	b 0	e 6	g 0	c 10	a 0	f 0
5	b 5	f 0	d 0	h 5	e 0	a 0	c 0	g 0
6	f 0	c 0	g 0	a 5	h 0	e 5	b 0	d 0
7	e 0	g 0	a 0	f 0	d 0	b 0	h 0	c 10
TOTAL	6	3	0	24	3	18	0	19

Figure 2: Belbin Roles of Adam Rehman



Question	IMP	CO	SH	PL	RI	ME	TW	CF
1	g 1	d 0	f 0	c 0	a 0	h 3	b 4	e 2
2	a 3	b 0	e 0	g 0	c 0	d 2	f 5	h 0
3	h 1	a 2	c 1	d 0	f 0	g 4	e 2	b 0
4	d 0	h 0	b 0	e 0	g 4	c 2	a 3	f 1
5	b 1	f 0	d 0	h 0	e 0	a 3	c 3	g 3
6	f 2	c 2	g 0	a 1	h 0	e 1	b 1	d 3
7	e 4	g 3	a 0	f 0	d 0	b 1	h 2	c 0
TOTAL	12	7	1	1	4	16	20	9

Figure 3: Belbin Roles of Brandon Cann



Question	IMP	CO	SH	PL	RI	ME	TW	CF
1	g	d 3	f 1	c	a	h 2	b 2	e 2
2	a 2	b	e 3	g 1	c	d 4	f	h
3	h 4	a	c 3	d	f	g 3	e	b
4	d 3	h	b	e	g 2	c	a 2	f 3
5	b 3	f	d 2	h	e 3	a	c	g 2
6	f 4	c 3	g 1	a 2	h	e	b	d
7	e 4	g	a	f 2	d	b 2	h	c 2
TOTAL	20	6	10	5	5	11	4	9

Figure 4: Belbin Roles of Xin Wang

10 References

References

[1] Chung-Wen Ho, Albert Ruehli, and Pierce Brennan. “The Modified Nodal Approach to Network Analysis”. In: *Circuits and Systems, IEEE Transactions on* 22 (July 1975), pp. 504–509. DOI: `10.1109/TCS.1975.1084079`.

[2] Taku Noda. “Object Oriented Design of a Transient Analysis Program”. In: Jan. 2007.