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: 01720256
adam.rehman19@imperial.ac.uk

Word Count: 4853

June 11, 2020

Contents

1 Overview of the report **3**

2 Project Specification **4**
- 2.1 Design Problem 4
- 2.2 Program Requirements 4
- 2.3 Design Criteria 5

3 Team Management **6**
- 3.1 Project timeline overview 6
- 3.2 Team Gnatt chart 7
  - 3.2.1 Important Notes: 7
- 3.3 Management approach 8
- 3.4 Team responsibilities breakdown 9
- 3.5 Project meeting minutes 10

4 Preliminary Designs **12**
- 4.1 Object Orientated Programming [OOP] 12
- 4.2 Modified Nodal Analysis 12
- 4.3 Eigen Matrix Library 13

5 Program Design **15**
- 5.1 Top level view of the program 15
- 5.2 Components files 16
- 5.3 *str handler.hpp* file 16
- 5.4 *node.hpp* file 16
- 5.5 *circuit.hpp* 17
- 5.6 *simulate.hpp* 18
  - 5.6.1 Non-linear components 18
  - 5.6.2 Simulation flowchart 20

6 Testing **21**

7 Critical Analysis **22**
- 7.1 OOP design 22
- 7.2 Repatriation Flight 22

8 Optimisations **24**
- 8.1 Sparse Matrices 24
- 8.2 Integration method 24
- 8.3 Inversion method 25

9 Adding on **26**
9.1 Graphics User Interface . . . 26
9.2 Components supports . . . 26
9.2.1 Diode . . . 26
9.2.2 Voltage controlled voltage sources - VCVS . . . 29
9.2.3 Voltage controlled current sources - VCCS . . . 29
9.2.4 Current controlled voltage sources - CCVS . . . 30
9.2.5 Current controlled current sources - CCCS . . . 31

10 Reflections **32**
10.1 Xin Wang . . . 32
10.2 Brandon Cann . . . 32
10.3 Adam Rehman . . . 33

11 Appendix **34**
11.1 Belbin Roles . . . 34
11.2 Gnatt Chart . . . 35
11.3 Meeting Minute Template . . . 36

12 References **38**

1 Overview of the report

Report layout loosely follows the stages of the Software Development Cycle.

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

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

Section 4 discussed the preliminary designs the team had produced and the rationale behind some design choices the team had implemented.

Section 5 gave a comprehensive overview of the program design, the functions and flowcharts used during program development.

Section 6 investigated the design constraints of the program, the accuracy of the results had produced and the speed of execution as the circuit size varied.

Section 7 discussed the major unforeseen problems encountered by the team over the course of the project and how the team attempted to mitigate that problem.

Section 8 discussed the possible improvements noted by the team that could mitigate concerns encountered by the team.

Section 9 detailed the possible features that can be added on to the existing program and the required modifications to the program to add the mentioned features.

Section 10 contained the reflections written by each teammate towards the end of the project.

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 specific file format.

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
- Parser to be designed to take in a input file of SPICE netlist format.
  - Basic syntax checks to be performed.
- 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 basic components are already implemented.

2.3 Design Criteria

The team had identified several factors from the list of factors given in the Product Design Specification [2] document.

- Maintenance: One of the most important factors the team identified. As the development timescale was constrained and the features required was open-ended, so it was very important we ensured the design could incorporate new features efficiently and cost-effectively if the client wished to add new features.

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

- 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 was important to balance proper team management techniques and creating a program that met client standards.

- Testing: 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 possibly incorporate SPICE into developing the program.

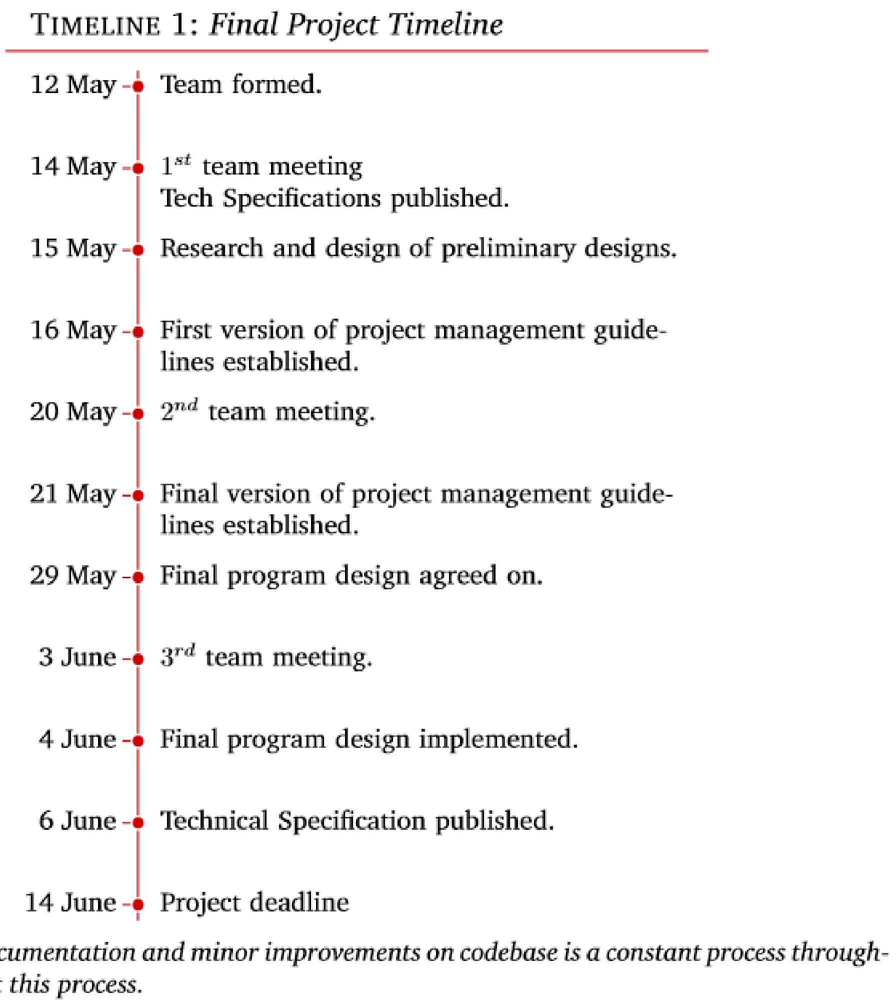
- Ergonomics: A Graphics User Interface (GUI) was not required by the client but it could possibly make the program much easier to use.

[2]Document included in *documents* folder.

3 Team Management

3.1 Project timeline overview

Xin Wang managed the team's project timeline and any modifications to the timeline required the agreement of all teammates.



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 Team Gnatt chart

The team Gnatt Chart waas created by using *GnattProject* software. It was not the most visually appealing software but, for its free features, it met the requirements of the team.

The Gnatt Project was mainly modified by Xin Wang upon requests by teammates after team meetings. Please refer to Appendix: Gnatt Chart for details of the team's Gnatt Chart.

3.2.1 Important Notes:

- Activities like ensuring standardised coding comments and formatting which had been monitored by Xin Wang was a constant background activity.

- A team meeting ocurred after each major stage of the Gnatt Chart finished (Indicated by a change in colour).

- On the 5th June of 2020, Xin Wang received a notification for a repatriation flight and, in the following team meeting, the team agreed on redistributing Xin Wang's responsibilities and for him to finish the existing Technical Report.

3.3 Management approach

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

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

This method was selected mainly due to the short duration of the project, requirements are clear and the team was not constantly required to report to the client at the early stages of the software development to ensure client satisfaction.

The Waterfall methodology prioritised proper documentation throughout the whole development process. This was important to the team as the team project required a Project Report to be submitted and, due to the ongoing remote lab orals and family-related obligations during quarantine, allowed a team member to be quickly caught up if a member stepped out of the team for a while.

The disadvantage of the Waterfall methodology was mainly in that no working software was produced until late in the cycle so there had been a level of risk and uncertainty with not being able to meet the deadline, especially in the with the remote working environment [3]. The team had later realised the methodology was not exactly suitable for the Object Orientated Approach that the team wanted to implement. As the integration was done at the end, there was concerns in the possibility that any errors in the program would not be noticed during the integration stage.

[3]Any unforeseen circumstances are listed in team meeting minutes summarised under the Critical Analysis section.

3.4 Team responsibilities breakdown

The responsibilities of each team member was discussed during the 1st meeting and had been formalised in the 2nd meeting. The respective roles are influenced by the Belbin questionnaire provided. [4]

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

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

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

3.5 Project meeting minutes

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

There was no standard protocol for calling team meetings, they are called usually when there was a problem that had affected all team members. Usually, there were smaller informal meetings across various channels from WhatsApp to Discord.

Over the course of the development cycle, there had 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.
    - Discussed 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 was 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 had a notification by South African Embassy of a Repatriation 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

The initial program design relied on the project briefing provided. Parsing a input file into a data structure was a simple task. The main point of discussion was related to how the program solved for unknown voltages.

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

4.1 Object Orientated Programming [OOP]

As the circuit components the program needed to support was an open ended requirement except the mandatory basic components listed, the team was concerned on the expandability of the program.

The OOP approach allowed the program design to be rapidly and cost-effectively altered to accommodate new features such as new circuit components or analysis techniques [4] that the client might request later on in the program's lifecycle. Increased program maintainability will save the programmer time and the client cost in program's active lifetime.

The disadvantage was that it has been 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.

It was the team's view that execution speed was not the highest priority and the ability to expand the program's feature set was also important. The design was detailed in the next section.

4.2 Modified Nodal Analysis

Modified Nodal Analysis (MNA) was an extension of normal Nodal Analysis. When the team studied similar programs like PSPICE and LTSpice, MNA had been mentioned numerous times [1] and its benefits had been proved during trails of programs based on preliminary designs. A problem the team encountered was trying to represent the current-dependent circuit elements like inductors efficiently.

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

The advantage of MNA was in its standard procedures. The standardised MNA algorithm for solving nodal equations allowed the team to modularise the solving aspect of the program. This meant not constantly be required to make changes to one of the most important parts of the program, instead just ensuring the component was in the compatible format to be entered into the MNA algorithm.

The MNA algorithm consisted of three matrices that form the matrix relation: $Ax = b$ [1]:

- A was consisted of four sub-matrices which included the admittance matrix. The program treated matrix A as a whole matrix instead of four smaller matrices and combining at the end.

$$A = \begin{bmatrix} G & B \\ C & D \end{bmatrix}$$

- x represented the unknown node voltages and current of the voltage sources.

$$x = \begin{bmatrix} v \\ i_v \end{bmatrix}$$

- b contained the values of voltage sources and current sources in the circuit.

$$b = \begin{bmatrix} i \\ e \end{bmatrix}$$

The admittance matrix was created during initialisation stage of the simulation. Non-linear elements are converted into suitable formats at each instance in time during transient simulation. The resulting model parameters are 'stamped' into the matrix at each instance. Only 'stamping' the changing values instead of creating a new system of matrices saved valuable execution time during program execution. The matrix inversion was handled by the Eigen library, the library supported various types of inversion with different accuracy and speed, this was discussed under the Optimisation sector.

4.3 Eigen Matrix Library

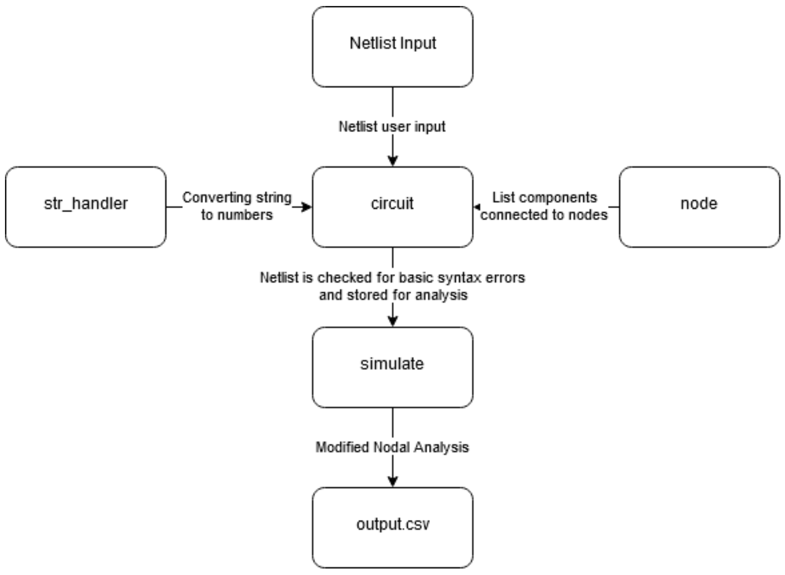
In terms of matrix operations, the team had decided to use a third-party program to handle the matrix-related operations in order to save development time and minimise potential errors. The primary concern the choice of third-party program was how seamless the program integrated into our program.

The team did not know whether the client would be comfortable installing an unknown third-party program on their system besides the program the client specifically requested.

There are third-party programs that required installation such as Armadillo and some had a large file size such as the Boost Library which was 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 was a file that had a reasonable size and only required that the Eigen library was located in the same directory. A copy of Eigen was included in the program files for client's ease of usage.

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.hpp file formed the centre of the program, handled the input of the file and stored it in a form that allowed the *simulate* file to perform Transient Analysis on it and output 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 managed all string-related functions.

The main function is the *tokenisier* function which take in an input sentence and separated it into individual words that can be processed, converted into the required formats for processing and storage.

During conversions, checks are performed at various stages. In the product briefing, there was no mention of error handling requirements but the team viewed that defensive programming was required and any errors encountered will be displayed to the user.

5.4 *node.hpp* file

A circuit can be expressed as a Network Graph, containing branches and edges. This notation was used to abstractly express the circuit and arrange how the the program operates. The function of the *node.hpp* file is to represent a node in the circuit and allow nodal equations to be created.

This function was not necessary in the final design as the program skipped creating nodal equations and directly input entries into the matrix before solving it. This part is kept to allow other functions to be developed such as printing nodal equations.

5.5 *circuit.hpp*

As mentioned previously, *circuit.hpp* managed all aspects of the program from reading in the user input, checking basic syntax errors during user entry to ensuring the simulation is successful. Numerous checks are performed over the course of the simulation and any errors encountered will instantly result in the simulation ending with an error message outputted.

The flow chart of *circuit.hpp* is shown as below:

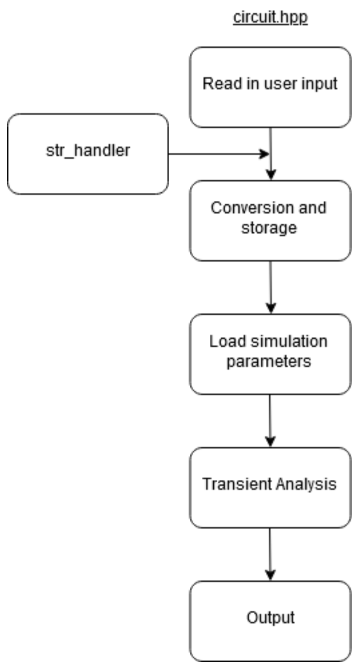


Figure 2: Circuit flow chart

5.6 *simulate.hpp*

This section of the program was where the team has spent most of the time and also where the team saw many possible optimisations which will be discussed in later sections. This section handled matrix entry and controlled the transient simulation.

But using the MNA technique, basic components such as voltage sources, current sources and resistors are simple and can be directly entered into the matrices for solving. However, components such as inductors and capacitors which are non-linear devices required further processing in order to support those components. At each instance in time, non-linear components need to be approximated into corresponding companion models. With the program using MNA, all non-linear components are converted into a current source and resistor to allow efficient entry into the matrix.

5.6.1 Non-linear components

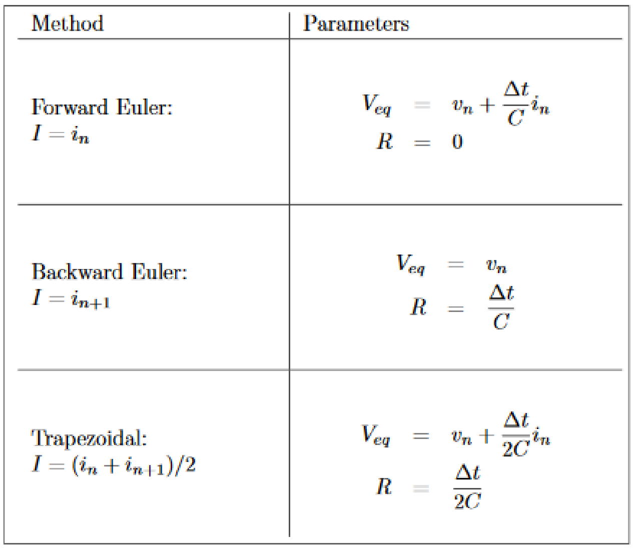
In the process of approximating non-linear components, integration would be required and there are three different methods of integration each with its own corresponding equations defining current and resistance of the companion model. Each type of integration offers different degrees of accuracy which are explored under the Optimisation section.

Capacitors are defined with the following differential equation:

$$I = C \times \frac{dv}{dt}$$

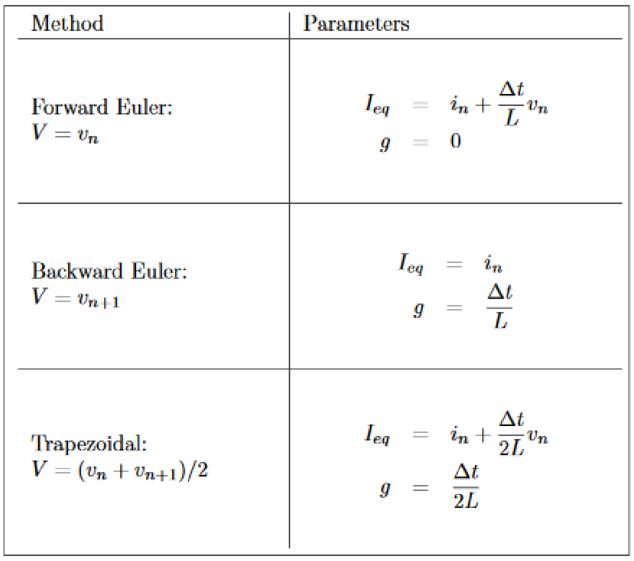
Inductors are defined with the following differential equation:

$$V = L \times \frac{di}{dt}$$



Method	Parameters
Forward Euler: $I = i_n$	$V_{eq} = v_n + \frac{\Delta t}{C} i_n$, $R = 0$
Backward Euler: $I = i_{n+1}$	$V_{eq} = v_n$, $R = \frac{\Delta t}{C}$
Trapezoidal: $I = (i_n + i_{n+1})/2$	$V_{eq} = v_n + \frac{\Delta t}{2C} i_n$, $R = \frac{\Delta t}{2C}$

Figure 3: Capacitor companion model parameters for different types of integration [2]

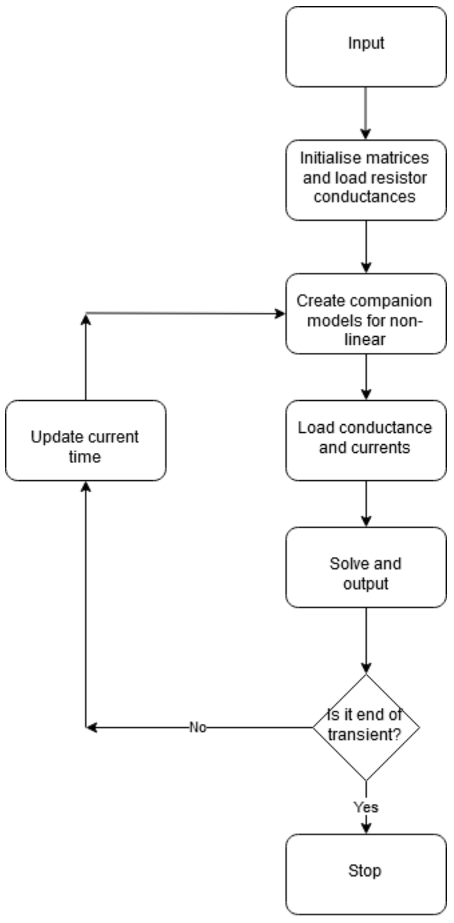


Method	Parameters
Forward Euler: $V = v_n$	$I_{eq} = i_n + \frac{\Delta t}{L} v_n$, $g = 0$
Backward Euler: $V = v_{n+1}$	$I_{eq} = i_n$, $g = \frac{\Delta t}{L}$
Trapezoidal: $V = (v_n + v_{n+1})/2$	$I_{eq} = i_n + \frac{\Delta t}{2L} v_n$, $g = \frac{\Delta t}{2L}$

Figure 4: Inductor companion model parameters for different types of integration [2]

5.6.2 Simulation flowchart

The listed simulation flowchart has not accounted for approximating components like Diodes which need Newton-Raphson iteration.


Input
Initialise matrices and load resistor conductances
Create companion models for non-linear
Load conductance and currents
Solve and output
Is it end of transient?
Yes
Stop
No
Update current time

6 Testing

It should be noted that majority of the testing was done on Linux Ubuntu 16.04.1 LTS which ran on a virtual machine in Windows 10, some testing have been completed on a Windows 10 machine. As such, the listed execution times are the average of 10 execution times.

Numerous functions are imbedded into the program itself to aide the benchmarking process and can be used to aide in debugging.

7 Critical Analysis

Throughout the process of program development, the team encountered two unforeseen circumstances that set the team back and required some adjustments on the timeline.

7.1 OOP design

It was felt strongly by the team that OOP design was required by the program. However, how the modules of the program were arranged was a topic that was argued and constantly revised.

To this problem, the team researched articles that investigated how circuit simulators could be implemented from the OOP perspective and, also, the team drew inspirations from the technical documentations of more established circuit simulation programs like SPICE3 and PSpice.

7.2 Repatriation Flight

On the 4th of June 2020, Xin Wang received a notification of a possible repatriation flight for South African citizens. As the exact time of the flight is not known yet and there was no guarantee that the quarantine site would have WIFI, it was very likely that Xin Wang could not be reached once arriving at South Africa. There was many unknown variables and the responsibilities that Xin Wang handled was very critical in this final stage of the program.

In the final team meeting, the team was briefed by Xin Wang of the state of the reports and what was still outstanding. As it was the weekend, the team could not reach the professors in charge of the project so the team had to make the decisions, mitigate the possible effects of Xin's absence and seek guidance as soon as possible. The steps the team undertook are summarised as below:

- Xin Wang to complete what can be completed on the report (Some data such as Testing has been completed yet)
- Xin Wang, once report is completed, shall handover the responsibilities to the remaining two teammates.
- Xin Wang will document any ideas he has yet to implement such as a guide to the demo video talking points.

- Tasks that the two teammates still have to complete:
  - Advanced component support
  - Demo video
  - Testing results and report

The team understood that in the workplace, there would be unforeseen circumstances that affected the team's workflow. It was important that the client's basic requirements are still met albeit some of the more advanced features would not be able to be completed but will still be documented under section: Adding On for the client to see and consider implementing.

Xin Wang will constantly keep the team brief on whether he can lend any help in preparing the program for submission. However in order to minimise unknown variables, the team planned with the possibility of Xin Wang not being able to communicate and the remaining team will have to handle report submission and video demonstration.

8 Optimisations

8.1 Sparse Matrices

Various circuit simulation engines like SPICE uses sparse matrices and it was mentioned [4] during our research stage but the current program used dense matrices. The primary concern was stability of the program as the team did not have any experience with Sparse Matrices but the Eigen Library is able to implement sparse matrices.

With the current supported circuit components, using dense matrices does not affect the program speed significantly. But as the circuit size gets bigger and more complicated components like operational amplifiers is supported, the matrices would get bigger. The memory used and the processing power used would increase proportionally.

Sparse matrices solves that problem as numerous entries of the matrices is zero and does not affect the results. The benefit is reduced memory usage and much faster program execution times which is a bottleneck in the current design. Due to the OOP program design, it can be quickly implemented but extensive testing would be needed.

8.2 Integration method

There are numerous different types of integration but the three most common forms of integration are as follows:

- Forward Euler
- Backward Euler
- Trapezoid

Out of the three, Trapezoid was the most accurate form of integration but the most processor intensive form of integration listed. The report had listed the three forms of integration and its respectively defined parameters.

But Euler's methods are only accurate to the first degree and Trapezoid to the second degree. Other methods such as the Runge-Kutta method would be much more accurate. This method would rely on creating a specialised function that exclusively handles it and would add to the processing time but, if sparse matrices were implemented, the team believes it would still

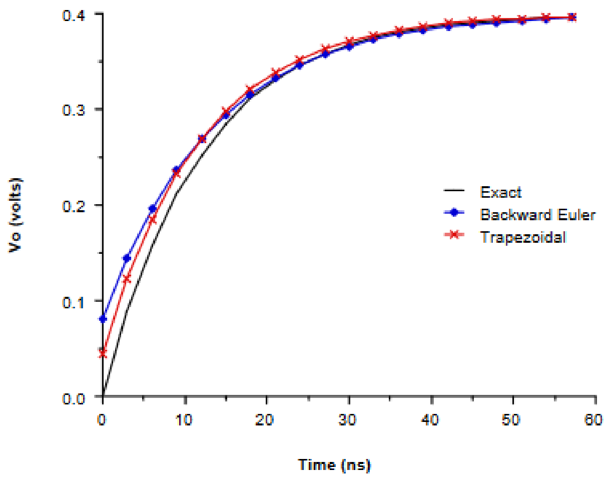


Figure 5: Accuracy comparison of different integration methods [5]

result in a faster program.

8.3 Inversion method

The Eigen library supported numerous types of equation solver functions. The default type that the Eigen documentation support was the *ColPiv-HouseholderQR* method but, using it, resulted in a average execution time of 50% longer. Instead, the team did not use the equation solver functions. The team used the inversion function on matrix A and multiplied it directly to matrix b.

The team believes that there would be more efficient third-party algorithms that can help make the program more efficient. For example, the Boost Library contains numerous functions that can entirely eliminate the need for the *str handler.hpp* file and, as it is a extensively supported library, it would be more efficient to use it. But it would require client consultations to ensure the client understands and was comfortable with a larger program size.

9 Adding on

9.1 Graphics User Interface

The current user interface was very limited, it consisted of only a basic interface created with bash script. The output file, though in .CSV format, required the user to use third-party programs like MatLab to plot the graph. If the user used the third-party program, GNUPLOT, it was possible to integrate graph plotting feature directly into the existing script.

Alternatively, a more user intuitive interface can be created in the future. Given the limited programming capability of the team, a third-party program to support the development of the User Interface is seen as the most viable by the team. Through research, the team has identified several possible programs that could be used:

- *Qt*
- *wxWidgets*
- *gtkmm*

The team's primary research is centered on the *Qt* program. *Qt* is a open source program and it is relatively easy given its complete feature set, using that program would minimise development cost and the interface features would not be limited.

Furthermore, with the C++ programming language used to create the program and the compatibility of *Qt* with C++, the team feels confident Qt would be the program used by the team to implemented the Graphical User Interface.

9.2 Components supports

9.2.1 Diode

For non-linear components like the Diode, a function would need to be created that will be able to approximate the component into a compatible model for our program. The diode is defined by the following relation

$$i(v) = I_s(e^{(\frac{v}{V_t})} - 1)$$

By using the Newton-Raphson method, the non-linear equation is linearised to the following equation[2]

$$i_{lin} = (i(v_n) - gv_n) + gv$$

The equation can then be used to derive the following companion model:

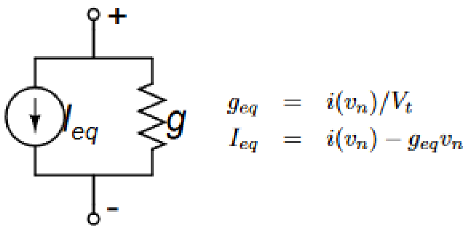


Figure 6: Companion model of the diode and its defining parameters [2]

As the Newton-Raphson iteration method requires the calculations to converge to a solution, in the event that it does not converge and max repetitions have been reached, the program could implement a convergence helper such as *gMin stepping* or *source stepping* and, if there is no more convergence helpers left, then the program will output an error to the user. The process of Newton-Raphson has many possible optimisations to improve the convergence such as improving the model used and the algorithm of the existing model.

The basic convergence parameters that would govern whether convergence has been met will be defined as:

- RELTOL: Tolerance of voltages and currents.
- VNTOL: Tolerance of nodal voltages.
- ABSTOL: Tolerance of branch current.
- ITL: Max number of iterations allowed.

The existing program flow chart would have to be modified to accommodate the Newton-Raphson iteration method.

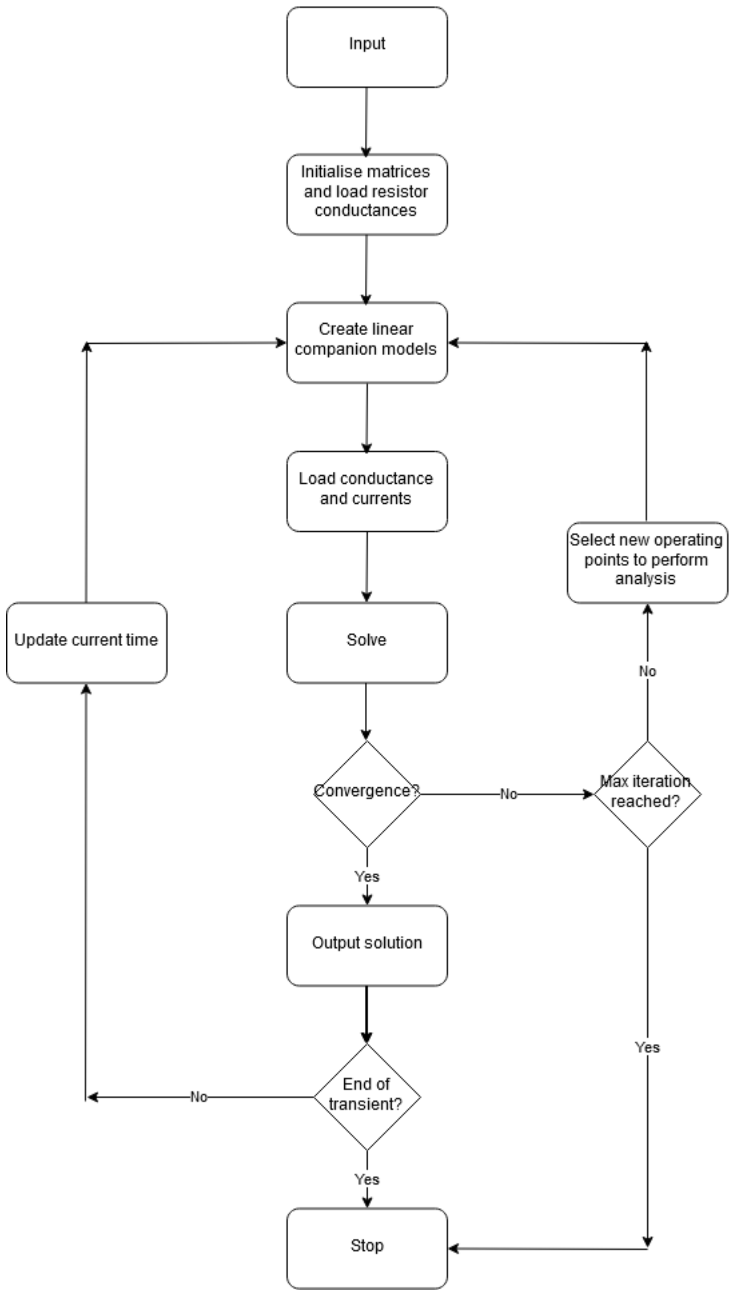
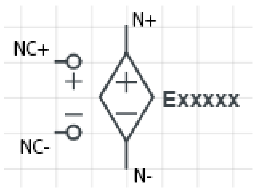

Input
Initialise matrices and load resistor conductances
Create linear companion models
Load conductance and currents
Solve
Convergence?
No
Max iteration reached?
No
Select new operating points to perform analysis
Yes
Output solution
End of transient?
No
Update current time
Yes
Yes
Stop

Figure 7: Program Flowchart incorporating Newton-Raphson method

9.2.2 Voltage controlled voltage sources - VCVS

VCVS would be defined in the netlist in the following format:



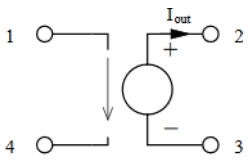
```
Exxxxx N+ N- NC+ NC- Value
```

Figure 8: Netlist format describing a VCVS [1]

The output voltage of a VCVS is defined by the following equation:

$$V_{out} = Value \times (V_{NC+} - V_{NC-})$$

where V_{NC+} and V_{NC-} are the controlling nodes. Due to the MNA method used by the program, supporting VCVS is easily done by modifying the existing system of matrices and solving for unknowns.



(a) Circuit symbol of a VCVS

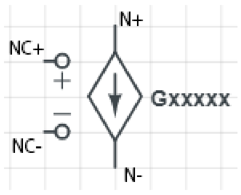
$$\begin{bmatrix} . & . & . & . & 0 \\ . & . & . & . & -1 \\ . & . & . & . & 1 \\ . & . & . & . & 0 \\ G & -1 & 1 & -G & 0 \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ I_{out} \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(b) MNA matrix entry

Figure 9: MNA matrix for a VCVS [3]

9.2.3 Voltage controlled current sources - VCCS

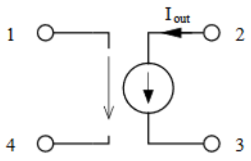
VCCS would be defined in the netlist in the following format:



```
Gxxxxx N+ N- NC+ NC- Value
```

Figure 10: Netlist format describing a VCCS [1]

VCCS output is considered as a current source hence only introduce one more unknown into the MNA matrix. It could be defined as the following:



(a) Circuit symbol of a VCCS

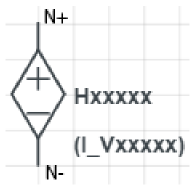
$$\begin{bmatrix} . & . & . & . & 0 \\ . & . & . & . & 1 \\ . & . & . & . & -1 \\ . & . & . & . & 0 \\ 1 & 0 & 0 & -1 & -\frac{1}{G} \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ I_{out} \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(b) MNA matrix entry

Figure 11: MNA matrix for a VCCS [3]

9.2.4 Current controlled voltage sources - CCVS

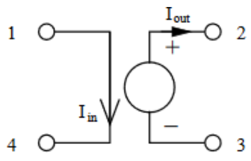
CCVS would be defined in the netlist in the following format:



```
Hxxxxx N+ N- Vxxxxx Value
```

Figure 12: Netlist format describing a CCVS [1]

The controlling variable is current through one of the voltage sources listed as the controlling element in the netlist. As MNA also calculates the current through the voltage sources, it will be relatively simple to extract that current value and find the output of the CCVS. CCVS creates two more unknown variables. It could be defined as the following:



(a) Circuit symbol of a CCVS

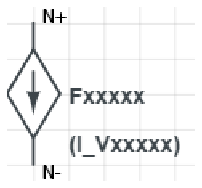
$$\begin{bmatrix} . & . & . & . & 1 & 0 \\ . & . & . & . & 0 & -1 \\ . & . & . & . & 0 & 1 \\ . & . & . & . & -1 & 0 \\ 0 & 1 & -1 & 0 & -G & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ I_{in} \\ I_{out} \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(b) MNA matrix entry

Figure 13: MNA matrix for a CCVS [3]

9.2.5 Current controlled current sources - CCCS

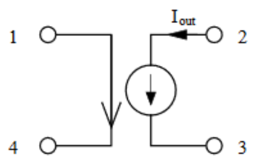
CCCS would be defined in the netlist in the following format:



```
Fxxxxx N+ N- Vxxxxx Value
```

Figure 14: Netlist format describing a CCCS [1]

Like the CCVS, this component is controlled by the current through the listed voltage source. It is handled similarly like CCVS. CCCS creates one more unknown variable.



(a) Circuit symbol of a CCCS

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \frac{1}{G} \\ \cdot & \cdot & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & -1 \\ \cdot & \cdot & \cdot & \cdot & -\frac{1}{G} \\ 1 & 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ I_{out} \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(b) MNA matrix entry

Figure 15: MNA matrix for a CCCS [3]

10 Reflections

10.1 Xin Wang

The project has made me realise that engineering is not only about the practical aspects that many see engineering as. Engineering requires as much work, if not more, in planning and documentation than implementing the ideas.

Through serving as the person in charge of documentation and project management, I have gained an appreciation for the work that goes on the background and the role has challenged me to gain new skills such as learning how to write Latex documents and researching how a project team can be efficiently managed.

I was initially worried that I would not be able to understand the program designs and be relegated to just being a minute keeper but, actually, my role required me researching the theory required to implement the program designs and it made realise that each role in the team is equally as important and it is important that the team communication effectively and be able to work together. As it is our first time handling a project like this, time was wasted with trial and error to work out a suitable team dynamic and, with COVID-19, we had to manage this is a remote working environment. It was frustrating but the skills gained are unquestionable.

Due to the repatriation flight, the project had an element of family obligations in it which made it more realistic for me. The workplace is not just about working and sometimes family emergencies require you to balance the two sides of your life. It is my view that, though the program might not have been realised to its full potential, the teamwork and management lessons I learnt made it all worth it and it has inspired me to read more about project management to allow me to contribute more the next time I work in a team.

10.2 Brandon Cann

I believe that this project was the first step into the engineering industry where it required us as a team to combine our strengths to complete an objective. This project helped me realise the importance of communication and time efficiency.

My main role of the team was to make sure that each member of the team was updated on each other's progress and to complete any areas where a member had fallen behind or could not complete.

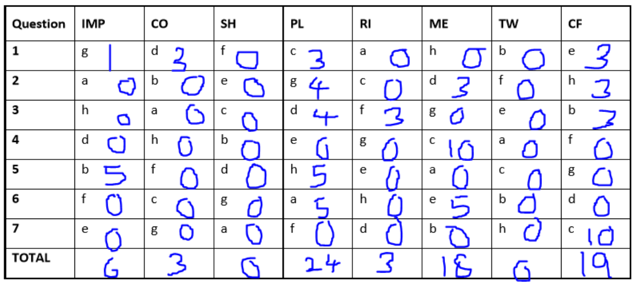
At the start of the project, the communication between the team was not particularly great where we had two members working on the exact same part of the code which was a huge waste of time. Soon after this realisation we organised a team meeting to clear up the uncertainties and from there on were on a more efficient path towards completion. Being the only member of the team that knew both team members well enough really helped me act as the centre of communication for the team.

As there was a team member with an emergency to attend to, we were left as two. This meant that I was now responsible to make sure both of us were aware of what was left to be completed and to distribute the workload between us. This was a great addition to my skill set as I am not usually comfortable with being in a lead role but being forced into this situation helped to make quick decisions on the way forward. I hope that the skills learned help me to deal with situations that are unforeseeable and make me an invaluable team member.

10.3 Adam Rehman

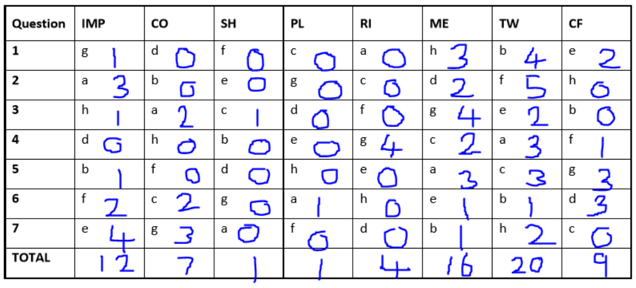
11 Appendix

11.1 Belbin Roles



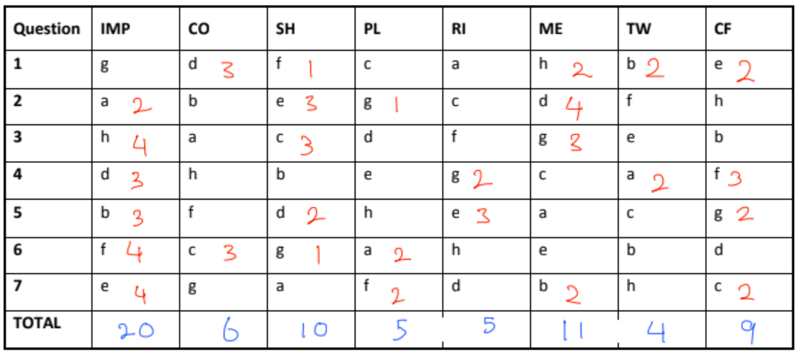
Question	IMP	CO	SH	PL	RI	ME	TW	CF
1	g 1	d 3	f 0	c 3	a 0	h 0	b 0	e 3
2	a 0	b 0	e 0	g 4	c 0	d 3	f 0	h 3
3	h 0	a 0	c 0	d 4	f 3	g 0	e 0	b 3
4	d 0	h 0	b 0	e 6	g 0	c 10	a 0	f 0
5	b 5	f 0	d 0	h 5	e 0	a 0	c 0	g 0
6	f 0	c 0	g 0	a 5	h 0	e 5	b 0	d 0
7	e 0	g 0	a 0	f 0	d 0	b 0	h 0	c 10
TOTAL	6	3	0	24	3	18	0	19

Figure 16: 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

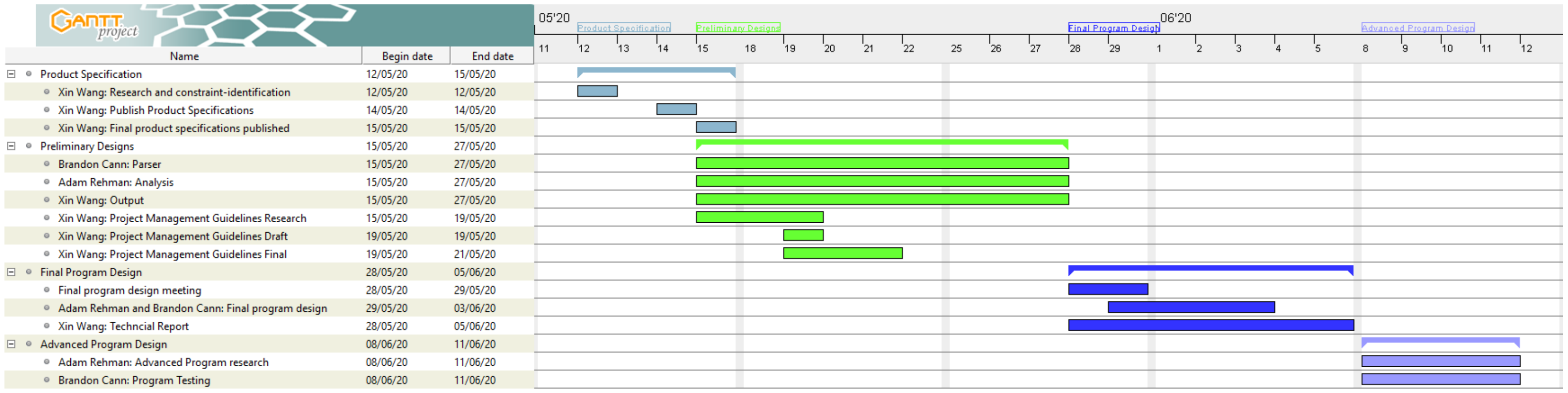
Figure 17: Belbin Roles of Brandon Cann



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 18: Belbin Roles of Xin Wang

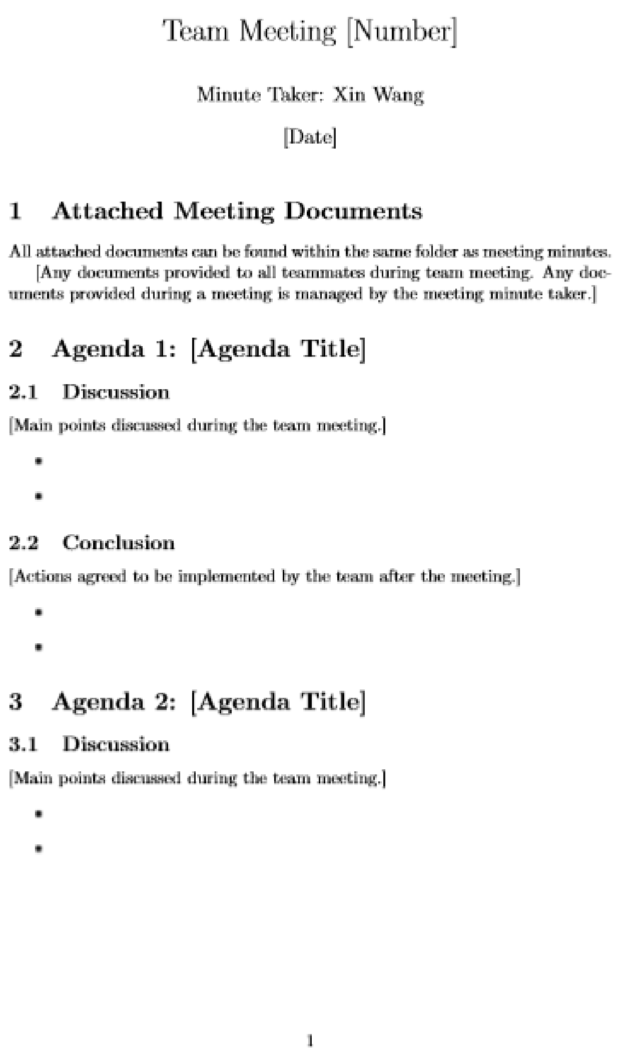
11.2 Gnatt Chart



Name	Begin date	End date
Product Specification	12/05/20	15/05/20
Xin Wang: Research and constraint-identification	12/05/20	12/05/20
Xin Wang: Publish Product Specifications	14/05/20	14/05/20
Xin Wang: Final product specifications published	15/05/20	15/05/20
Preliminary Designs	15/05/20	27/05/20
Brandon Cann: Parser	15/05/20	27/05/20
Adam Rehman: Analysis	15/05/20	27/05/20
Xin Wang: Output	15/05/20	27/05/20
Xin Wang: Project Management Guidelines Research	15/05/20	19/05/20
Xin Wang: Project Management Guidelines Draft	19/05/20	19/05/20
Xin Wang: Project Management Guidelines Final	19/05/20	21/05/20
Final Program Design	28/05/20	05/06/20
Final program design meeting	28/05/20	29/05/20
Adam Rehman and Brandon Cann: Final program design	29/05/20	03/06/20
Xin Wang: Techncial Report	28/05/20	05/06/20
Advanced Program Design	08/06/20	11/06/20
Adam Rehman: Advanced Program research	08/06/20	11/06/20
Brandon Cann: Program Testing	08/06/20	11/06/20

Figure 19: Gnatt Chart of Team Project

11.3 Meeting Minute Template



Team Meeting [Number]

Minute Taker: Xin Wang

[Date]

1 Attached Meeting Documents

All attached documents can be found within the same folder as meeting minutes.

[Any documents provided to all teammates during team meeting. Any documents provided during a meeting is managed by the meeting minute taker.]

2 Agenda 1: [Agenda Title]

2.1 Discussion

[Main points discussed during the team meeting.]

-
-

2.2 Conclusion

[Actions agreed to be implemented by the team after the meeting.]

-
-

3 Agenda 2: [Agenda Title]

3.1 Discussion

[Main points discussed during the team meeting.]

-
-

1

3.2 Conclusion

[Actions agreed to be implemented by the team after the meeting.]

-
-

This document and all its attachments have been sent to and accepted by the following members of the team:

- Adam
- Brandon Cann

12 References

References

[1] Erik Cheever. *Analysis of Circuits*. DOI: https://lpsa.swarthmore.edu/Systems/Electrical/mna/MNA1.html.

[2] Steven Herbst and Antoine Levitt. "Companion models for basic non-linear and transient devices". In: (2008).

[3] Stefan Jahn et al. *Qucs Techncial Papers*. 2003, pp. 108–110. URL: https://qucs.github.io/docs/technical/technical.pdf.

[4] Taku Noda. "Object Oriented Design of a Transient Analysis Program". In: Jan. 2007.

[5] Jose E. Schutt-Aine. *MNA and SPICE*. 2020. DOI: http://emlab.illinois.edu/ece546/Lect_16.pdf.