



**Department of Electrical, Computer  
& Biomedical Engineering**  
Faculty of Engineering & Architectural Science

# Ladder Iterative Load-Flow Algorithm

by

Muhammad Shirazi, Abrar Ahsan, Rehnuba Fairoj, Parham Habibi

Electrical Engineering Capstone Design Project

Ryerson University, 2022

## Acknowledgements

We would like to express our appreciation and gratitude to Dr Bala Venkatesh as well as The Clean Energy Zone (CEZ) for allowing us the opportunity to work on this project.

We want to thank Dr Thiago Ramos Fernandes and Shima Bagher Zade Homayie for their patience and help. We would not have been able to complete this project without their guidance and assistance, along with their patience through every roadblock we faced.

Most importantly, none of this could have happened without our families. To our parents and grandparent – it would be an understatement to say that, as a family, we have experienced some ups and downs in the past year. This report stands as a testament to your unconditional love and encouragement.

## Certification of Authorship

We hereby acknowledge and certify that we are the authors of the following paper and that any assistance we received is disclosed and acknowledged within the document. All sources, ideas and data which were used have been paraphrased and referenced and cited within this document.

Signatures of all group members

<i>Muhammad Shirazi</i>	<i>Rehnuma Fairoj</i>	<i>Abrar Abrar</i>	<i>Parham Habibi</i>
-------------------------	-----------------------	--------------------	----------------------

# Table of Contents

Acknowledgements .....	I
Certification of Authorship.....	II
Table of Contents .....	III
Abstract.....	VI
1 Introduction.....	1
1.1 Motivation .....	1
1.2 Objectives.....	1
1.3 Related Works.....	1
2 Theory Review.....	3
2.1 Distribution Systems .....	3
2.1.1 Types of Distribution Systems .....	3
2.1.2 Radial Distribution System .....	3
2.2 Power Flow Analysis in the Radial System .....	4
2.2.1 Backward & Forward Sweep Method .....	4
2.2.2 Power Loss .....	4
2.2.3 Power Injection .....	5
2.2.4 Power Flow .....	5
3 Design .....	6
3.1 Graphical User Interface .....	6
3.1.1 GUI Overview .....	6
3.1.2 Input Window.....	7
3.1.3 Message Window .....	7
3.1.4 Preview Window .....	8

3.2	Data Parser .....	10
3.2.1	Data-Format Verification .....	10
3.2.2	Data Extraction .....	11
3.2.3	Branch Reorder .....	11
3.2.4	Data Exporter .....	12
3.3	Calculation Engine .....	12
3.3.1	First Iteration Design .....	13
3.3.2	Final Iteration Design .....	13
3.3.3	Forward Sweep .....	13
3.3.4	Backward Sweep .....	13
3.3.5	Power Loss .....	14
3.3.6	Power Injection .....	14
3.3.7	Power Flow .....	14
4	Test System and Analysis .....	15
4.1.1	33 Bus System .....	17
4.1.2	69 Bus System .....	19
5	Conclusions .....	22
5.1	Accomplishments .....	22
5.2	Challenges .....	22
5.3	Future Work .....	22
6	References .....	23
	Appendices .....	25
	Appendix A: Input CDF For Testing .....	26
	A.1 IEEE CDF 33 Bus System .....	26
	A.2 IEEE CDF 69 Bus System .....	27

B.1 33 Bus System .....	30
B.2 69 Bus System .....	31
C.1 33 Bus System .....	33
C.2 69 Bus System .....	34
D.1 33 Bus System .....	36
D.2 69 Bus System .....	37
E.1 33 Bus System.....	38
E.2 69 Bus System.....	38

## Abstract

Load flow can be described as the analysis of steady state power systems, and in order to analyse these large-scale distribution systems an accurate method of conducting load flow analyses must be used. Using the Ladder iterative method, we can reach convergence allot more reliably and solve for voltage, current and power. The method functions by calculating the voltage and error during the forward sweep, while the current is calculated during the backwards sweep. This is calculated for each of the busses within the system and respective branches, and the calculations are conducted until the error value is lower than the respective tolerance assigned. The overall design takes in input from the user for the error tolerance value and the IEEE CDF file for n-defined busses and can parse through the data. The data is then passed along to the calculation engine which runs through the voltage and current calculations on the forward and backwards sweep respectively. This is done until the condition is satisfied that the error value is less than the tolerance given by the user, thus reaching convergence. If convergence is not met after 50 iterations the program is made to terminate. Simultaneously the progress of the program is being displayed to the GUI Message Window, and when the program concludes all calculations and convergence is achieved the final results are exported to an Excel file as well as shown in the different tabs in the Preview window GUI. Overall, the system can handle n-bus distribution systems given the restriction they are radial and do not have transformers stepping up/down the voltage. It performs each of the calculations are done efficiently given the overall algorithm's restrictions and convergence is achieved fast.

**Keywords:** Backward/Forward sweep method, Distribution System, Load flow analysis

# 1 Introduction

## 1.1 Motivation

Load-flow is a simulation of a static power system to determine the behaviour of the installed network as well as planning future networks. It assists in predicting the power flow on the transmission lines for static loads. Load studies are important in planning future development projects. As these projects tend to be planned for 10 years or more, the designers need to be able to run simulations and design the systems quickly and accurately. As such, fast and accurate operation of these simulations has been a research topic for decades.

While there are many proposed methods, conventional techniques such as Newton Raphson and other transmission system algorithms while successful in transmission systems with mesh structures, can fail for distribution systems due to high X/R ratios, unbalanced distributed loads and multi-phase, unbalanced operation [1]. As such, iterative algorithms are used in distribution systems.

## 1.2 Objectives

The objective is to design and implement a robust software solution in Python following the ladder-iterative technique to calculate the load-flow solutions for various power distribution systems. The implemented algorithm is designed to be optimized enough to compete, if not surpass, past works involving iterative techniques.

While distribution systems are usually in three-phase, our Python program has been modeled to run calculations in single phase based on the data-input via excel file and present the results after convergence in an excel file. The software is designed to allow the user to control the tolerances as well as view the results using an interactive graphical user interface (GUI).

## 1.3 Related Works

Conventional Newton Raphson and Fast Decoupled Load Flow (FDLF) methods are inefficient at solving large radial systems. The method and its variants have been developed specifically to work with transmission systems which tend to have a mesh structure with parallel lines and multiple redundant paths.

Distribution systems tend to be radial or weakly meshed, meaning it has one main supply from a substation which converts from high voltage to low voltage for delivery to customers, and no other supply connected. Numerical methods like Newton-Raphson can provide great accuracy with fewer iterations [2], however, it is computationally expensive even after efficiency optimizations are applied [3]. Tripathy et al. [4] implemented a Newton-like method for solving ill-conditioned power systems which can achieve voltage convergence; however, it is unable to achieve optimal power flow calculations. Decoupled methods such as FDLF face problems in convergence due to distribution systems having high R/X ratios [5], as coupling between real and



reactive power in a circuit is because of the resistive component of the line impedance. Tortelli et al. implemented a complex per-unit normalization technique to tackle the high R/X ratio [6]. Gauss-Seidal power flow algorithm is also shown to be extremely inefficient for solving large power systems [7].

For such systems, a backward-forward sweeping method is introduced to analyze the radial network. First introduced by Berg et al. [8], it has several variations and improvements made. Kersting [9] introduced a load-flow technique that uses a ladder-iterative algorithm to solve for such radial systems. Ellis [10], Al-Awadi [11] extends this to be applied to different systems. Augugliaro and Dusonchet [12] improved on this further by adding a decomposed forward sweep. Stevens et al. [13] implementation of the ladder-iterative algorithm demonstrated that it is the fastest approach, however, it failed to converge on 5 out of 12 cases studied. Arunagiri [14] demonstrated that using classical method is not the only way, as his implementation used an artificial neural network on a radial system.

While multiple algorithms exist, their software implementations also contribute to their efficiency. Musti et al. [15] implemented load-flow analysis using four different algorithms on Microsoft Excel using VBA, Ellis [10] implemented his algorithm on DEWorkstation, and Rupa et al. implemented a backward/forward sweep method using MATLAB [1].

In this work, we present a Python-based ladder-iterative solution for radial networks that takes in inputs from Excel files, runs the algorithm and generates the output as an Excel file. The method is tested on the 33-bus and 69-bus radial distribution systems. The theory and mathematical formulations are in the following sections.

## 2 Theory Review

### 2.1 Distribution Systems

A power distribution system is the final stage in delivering electrical power from multiple generation sources, through the transmission system and finally to the individual customers, from high to low voltages [1]. Voltages from the generation are high voltages, whereas the voltage needed by customer needs to be low voltage, as such the high voltages are delivered to substations, which are located closer to the customers, and stepped down via transformers to utilization voltage level. Typically, a substation can deliver power to multiple customers through the distribution system.

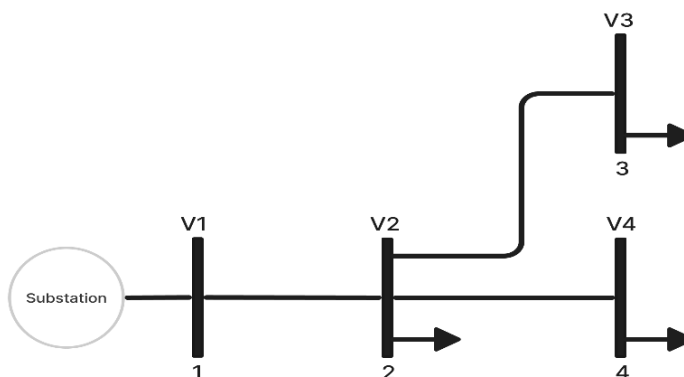
#### 2.1.1 Types of Distribution Systems

- a) Radial Distribution System
- b) Ring Main Distribution System

A ring system contains more switches and conductors in its construction, which makes this type of distribution system more expensive [1], which is undesirable in a system that uses low voltage.

#### 2.1.2 Radial Distribution System

A radial distribution system originates from a single source, which is a substation or a feed distributor, and spreads out towards the customers or loads. There is a main branch which is the longest line, which can have branches followed by sub-branches. The radial structure guarantees that there will be no circular connection in the network and each bus is connected to the source via one path only [1]. This is the simplest configuration; however, it is also the least reliable [16]. This is primarily used in areas which are sparsely populated. The main advantage a radial network provides is that due to its simple configuration, the construction cost is also much lower [1] [6].



**Figure 2.1.1:** Radial Distribution System from a Substation

## 2.2 Power Flow Analysis in the Radial System

Power flow analysis is done to obtain the voltage magnitude, power losses, as well as the power flow in the system.

### 2.2.1 Backward & Forward Sweep Method

The backward and forward sweep method is the core part of the algorithm. It is solved iteratively, calculating voltage and current values at each bus, which is further used to find the power flow, loss, and injection.

#### 2.2.1.1 Forward Sweep

The forward sweep calculates the voltage drop for each bus in the radial system. The source voltage is considered as 1 per-unit, or the actual rated voltage in V, and the following buses are calculated using updated current values.

$$V_{k+1} = V_k - Z_{k,k+1} I_{k,k+1} \quad (1)$$

#### 2.2.1.2 Backward Sweep

The backward sweep calculates the current flow between buses in the radial system. The voltages calculated during the forward sweep are used. It starts from the last bus and moves towards the source bus.

$$I_{k+1} = \left( \frac{S_{k+1}}{V_{k+1}} \right)^* = I_{k,k+1} \quad (2)$$

$$I_{k-1,k} = I_k + I_{k,k+1} \quad (3)$$

The current is calculated across each bus as seen in equation (3), along with the load currents as calculated using equation (2), which are added to account for its draw.

Once the backward sweep is complete, the forward sweep is run again, which would result in different voltage values as the current values have changed. This process will run recursively until voltage convergence is reached.

### 2.2.2 Power Loss

Power is lost during the distribution stage due to the line's internal impedance:

$$S_{loss_{km}} = V_k \left[ \frac{V_k - V_m}{Z_{km}} \right]^* + V_m \left[ \frac{V_m - V_k}{Z_{km}} \right]^* \quad (4)$$

$$S_{Total\ loss_{km}} = \sum_{\forall \{k,m\}} S_{loss_{km}} \quad (5)$$

### 2.2.3 Power Injection

Power is delivered to each bus via the transmission line. The current flow is calculated by accounting for the voltage difference between the two buses and then dividing by the line impedance. As a single bus can be connected to multiple buses, all these currents must be added together and finally multiplied by the voltage of the bus to find the power injected into it.

$$S_{injection_k} = V_k \sum_{\forall \{k,m\}} \left[ \frac{V_k - V_m}{Z_{km}} \right]^* \quad (6)$$

### 2.2.4 Power Flow

The final objective is to find the power flow. The power flow is calculated by removing the line loss and the load loss from the power injection

$$P_{k+1} = P_k - P_{loss,k} - P_{load,k+1} \quad (7)$$

$$Q_{k+1} = Q_k - Q_{loss,k} - Q_{load,k+1} \quad (8)$$

## 3 Design

The design of the software went through multiple iterations to find the most optimized and efficient, as well as most user-friendly implementation of the system. Detailed below are the functionality of each section of the software system.

### 3.1 Graphical User Interface

The Graphical User Interface (GUI) allows the user to interact with the software. The user can select the input data file, the tolerance value they want to set for the calculations, view error prompts when problems are encountered and preview their output data. The initial GUI was primarily an input-based GUI but during implementation, new user interactions were needed and therefore the GUI evolved.

The first half of this project was used to primarily in trial and error as there were no solid requirements for the structure of the GUI and this period was used to development a better understanding of the PySimpleGUI library in Python. The first iteration of the GUI to set up a small pop-up window with a button to close the said window. The second iteration allowed the user to browse and select the data file to be inputted. The third iteration expanded on this, adding a file-type verification, limiting the user to only select Excel file types. Finally, the fourth iteration expanded to display error messages. There are three primary error messages:

1. Incorrect File-Type: If the program is unable to recognize the file type, it will send a file-type error.
2. Data is not in CDF Format: If the data in Excel does not follow the IEEE Common Data Format, it will send a data-format error.
3. Failure to Converge: If the calculation engine is unable to converge, i.e., unable to reduce the error percentage low enough, it will send a convergence error.

Only the file-type error is checked in this module. The data-format error is checked in the Data Parser module and the convergence error is checked in the Calculation Engine.

As the second half of the project was dedicated to the implementation of the project, there was a greater need for more user interactions with the software. Therefore, during this stage the first GUI was further developed to account for user input of tolerance and two new windows were introduced. These two news windows are the message and preview windows.

#### 3.1.1 GUI Overview

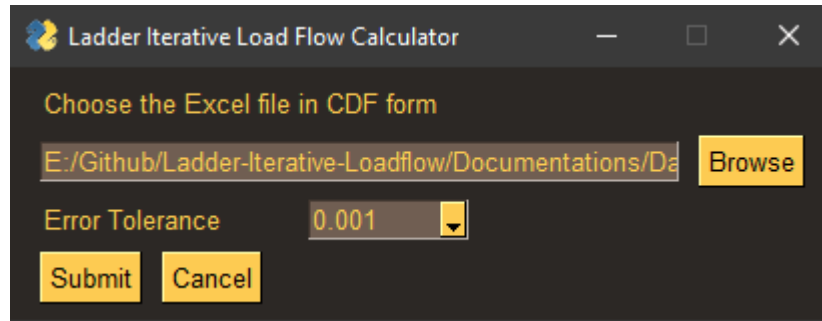
The GUI of the Software consists of three windows. The input window accepts addresses containing the location of the CDF or .xlsx file along with a tolerance selected by the user. If the user does not specify the tolerance, a default tolerance of 0.001 is applied. This information is passed onto the Data Parser and the Calculation Engine. As the Data Parser and Calculation Engine of the software is being run, the Message window displays the status of the program while the software is run. This window will show checkpoint print statements throughout the program for a successful run. However, if there is an error, this would trigger an exception and the message

will be displayed on the GUI. After successfully running completing the calculations, a third window called the Preview window which exhibits the results from the calculation engine in a tabulated format will pop up. This output data is saved immediately after a successful run. The preview window consists of five different tabs labelled Voltage, Power Flow, Power Loss, Power Injection and Error Iteration. The user can then choose to open the saved excel file or exit the program.

### 3.1.2 Input Window

The second phase saw to a redesigning of the input GUI. The input GUI up until was only able to accept spreadsheet, more specifically only able to accept (.xlsx) type of spreadsheets during the first half of the project. This is since excel has many different supported file types and the one used for IEEE CDF is through .xlsx specifically. This was an improvement on the initial case of presenting an error message for the wrong file type as this makes it easier for the software to eliminate any user file input error completely. Furthermore, the convergence failure error was removed from the input window entirely and moved to the new Message Window.

The input window consists of a textbox where the user can input the address of the IEEE CDF file through entering the location or browsing it. The user can also specify the input tolerance which includes in given by a drop-down menu. This menu has a default error tolerance of  $10^{-3}$  but this can be extended to  $10^{-6}$ .



**Figure 3.1.1:** Input Window GUI.

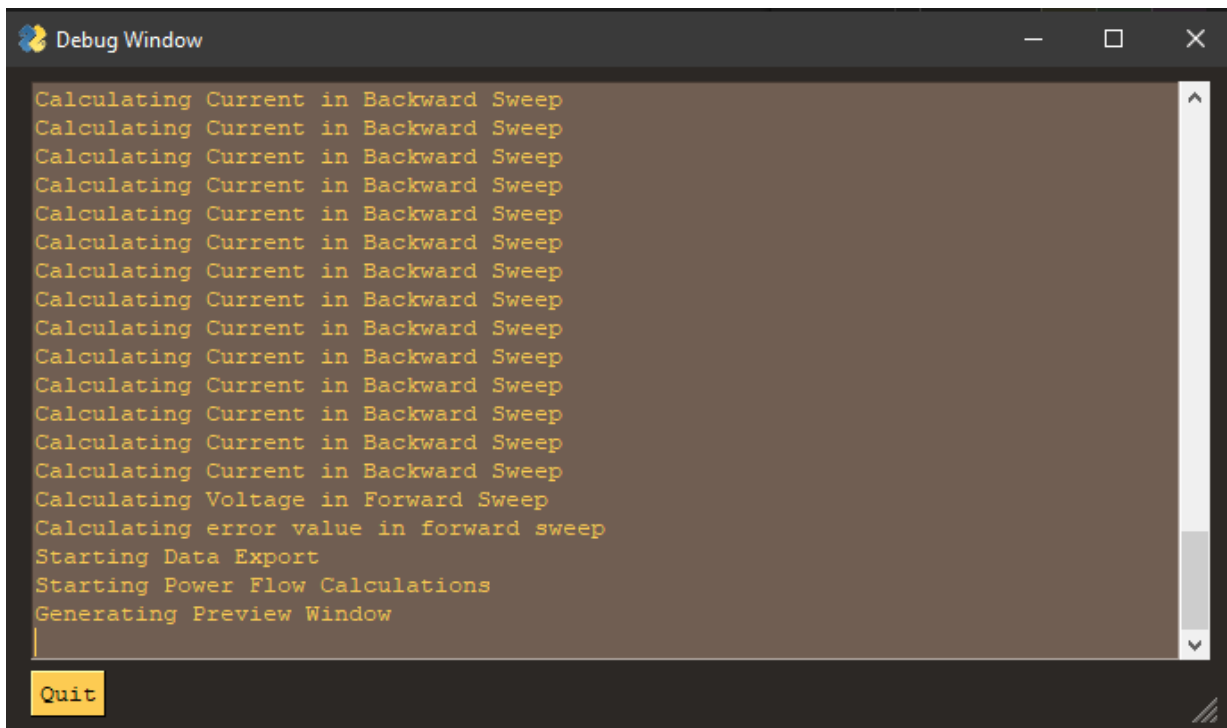
The development of the error tolerance drop-down menu went through stages of implementation, where initially it was made via a text window where the user could input their error tolerance. However, as the complexity of the program rose the error checking and exception handling for the error tolerance field was too high of a complexity to add in with the rest of the program, thus it was swapped to buttons and ultimately to a drop-down menu.

### 3.1.3 Message Window

The next GUI presented to the user when the data parser and the calculation engine are running is the message window which utilizes PySimpleGUI's print function [14]. The print function allows for a generation of the message window which can demonstrate to the user which checks have passed and where it has failed (i.e., if the data doesn't converge or incorrect values). These print statements were included at key portions of the software which include the start of

the data parsing, once the parsing has completed successfully or unsuccessfully, start of the forward and backward sweeps for the calculation engine, once convergence occurs/ doesn't occur and finally when the data is being output to the excel file.

One of the issues faced with the Message Window GUI is the fact that it would close automatically after all the messages have been displayed. This wouldn't allow for the user to even read if an error had occurred in the program thus a delay was introduced via the sleep function in python to allow user to read where the program had encountered an error before the window would automatically close.



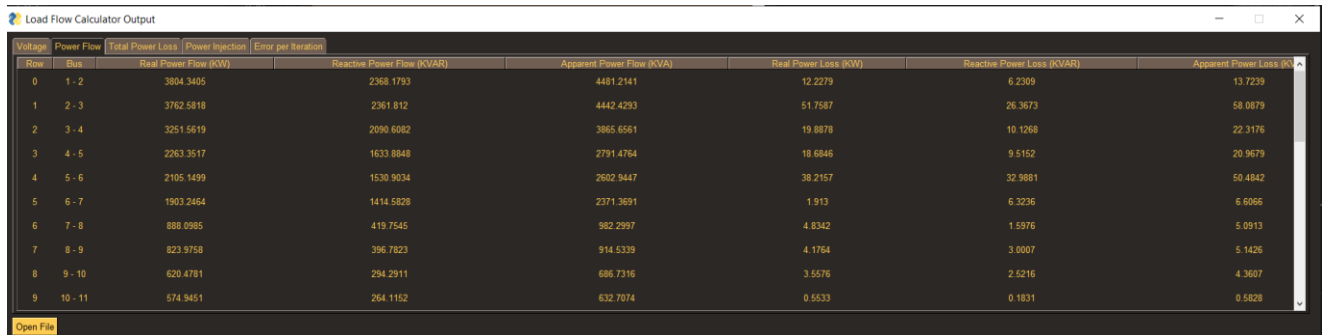
**Figure 3.1.2: Message Window GUI**

### 3.1.4 Preview Window

The final GUI the user sees is the Preview Window which takes the output from the calculation engine as well as the data parser and presents the data in tables through the various tabs set up. This GUI was one of the most difficult ones to setup because the PySimpleGUI table command requires the input data to be in a very particular format [16].

The data is initially input as a Pandas data frame from the calculation engine and the data parser is then organized depending on which tab the data will belong in. This data frame is then converted to a list and each of the components are converted from integer and float values to strings. After the conversion to a list of strings the data is converted to a Numpy array since those are easier to manipulate the data with, and stacks are created depending on

how the data is to be organized and in which tab. Finally, the stacks are transposed and converted from Numpy arrays to lists as per the requirements of PySimpleGUI's table format. Each stack of data corresponds to a tab in the GUI which will show specific data[16].



Row	Bus	Real Power Flow (kW)	Reactive Power Flow (kVAR)	Apparent Power Flow (kVA)	Real Power Loss (kW)	Reactive Power Loss (kVAR)	Apparent Power Loss (kVA)
0	1 - 2	3004.3405	2368.1793	4481.2141	12.2279	6.2309	13.7239
1	2 - 3	3762.5818	2361.812	4442.4293	51.7587	26.3673	58.0879
2	3 - 4	3251.5519	2090.6082	3865.6561	19.8878	10.1268	22.3176
3	4 - 5	2263.3517	1633.8848	2791.4764	18.6846	9.5152	20.9679
4	5 - 6	2105.1499	1530.9034	2602.9447	38.2157	32.9881	50.4842
5	6 - 7	1903.2464	1414.5828	2371.3691	1.913	6.3236	6.6066
6	7 - 8	888.0985	419.7545	982.2997	4.8342	1.5976	5.0913
7	8 - 9	823.9758	396.7823	914.5339	4.1764	3.0007	5.1426
8	9 - 10	620.4781	294.2911	686.7316	3.5576	2.5216	4.3607
9	10 - 11	574.9451	264.1152	632.7074	0.5533	0.1831	0.5828

**Figure 3.1.3:** A general image of the Preview window

i. 1<sup>st</sup> stack= Voltage Data

This is the first tab in the preview window. In the Voltage Data stack, details pertaining to the voltages in each bus are tabulated in this tab. The set of data displayed in this tab are under the headers Bus (number), Voltage Magnitude (PU), Voltage Magnitude (V) and Voltage Angle (Degrees).

ii. 2<sup>nd</sup> stack= Power Flow

Under the second tab, the power flow characteristics of the final iteration is displayed. The headers used to extract data from the stack are Bus (Line between indicated buses), Real Power Flow (kW), Reactive Power Flow (kVAR), Apparent Power Flow (kVA), Real Power Loss (kW), Reactive Power Loss (kVAR) and, finally, Apparent Power Loss (kVA). Power flow is a crucial information in calculating efficiency and can be used to optimize distribution designs.

iii. 3<sup>rd</sup> stack= Total Power Loss

The third tab only contains a single line of data which is the total Power Loss. This is given by finding the sum of the losses in between each bus. There are only three headers in this tab titled Total Real Power Loss, Total Reactive Power Loss and finally the Apparent Power Loss.

iv. 4<sup>th</sup> stack=Power Injection

Power Injection refers to the power entering a bus in the last forward sweep from the previous bus. The tab contains the headers Bus (number), Real Injected Power, Reactive Injected Power and finally Apparent Injected Power.



- v. 5<sup>th</sup> stack=Error Per Iteration

The final tab in the preview window displays the errors calculated for every iteration that the software ran before reaching convergence. This tab has two headings Iteration and Error Percentage. The percent deviance is calculated, and iteration is specified.

The layout was designed with sole intent of having the data distributed amongst the 5 tabs specified above. Each of the tabs had their layouts designed in such a way to include the table of the corresponding stack. In addition, to the tabs, the user has the flexibility of seeing the data in the output .xlsx file created. This output file can be accessed using the open file button as shown in the Figure 4.1.4. Once clicked, the spreadsheet produced in the very last run is opened.

## 3.2 Data Parser

The Data Parser consists of multiple functions, but the primary objective is to extract the relevant information of the input Excel file from the path provided by the GUI. To accomplish this, the development made use of multiple Python libraries, but the most important ones are Pandas and NumPy, both very powerful data manipulation tools. These libraries allow the data to be extracted efficiently without disrupting the worksheet structure of Excel. Once extracted, multiple verification checks are run to ensure that the data contains the necessary information to allow the load-flow calculations to begin.

### 3.2.1 Data-Format Verification

Power systems have only grown in complexity over time. As the volume of data grew, a standard system was required as it would allow the ease of translatability as well as make it easier to run across different software tools without making major modifications to the software or the data. As such, a common data format (CDF) was established to allow the ease of data exchange [17] [18]. The CDF format ensured that the relevant information was split across multiple tables/cards, allowing faster modeling of a power system. This CDF format was followed in our implementation of the data parser.

Once the Panda DataFrame has been created during the data parser's initialization, multiple verification checks are completed to find:

1. Bus Data Table
2. Branch Data Table
3. Header with SBase

The verifications for the bus and branch data tables are completed by searching for the respective table's header. As both table's header contains the string "BUS DATA FOLLOWS" and "BRANCH DATA FOLLOWS", the absence of these strings would mark that the tables are not present and would throw a false on the verification check. If found, it would return a true to mark that the tables exist and can proceed to extraction.

Similarly, the SBase is required for the calculations to convert the load values to per-unit (PU) values. To ensure that the SBase is present, the presence of the header of the entire Excel is

verified as it contains the value. This verification is completed by searching for the string “RYERSON UNIVERSITY”, as that would be the header of the CDF files. If found, true is passed and the extraction of SBase can begin.

### ***3.2.2 Data Extraction***

Once all the verification checks have been completed, the DataFrame is converted to a NumPy array and passed onto the extraction function. This is done to easily iterate through the data to find the relevant information while preparing the data for the calculation engine to take in easily.

As mentioned in the previous section, the tables are identified using their respective strings to mark their respective header. Once found, the following row is the first row of data, as such, its index is saved. The end of each table is numerically marked with the value “-999”, as such, the extraction searches for this value in the first column and once found, the previous row is the last row with relevant data and its index is saved. With both the start and end indexes, the extraction begins.

For the Bus Table, the data extracted are the Bus Number, Nominal Load in MW and MVAR, and Voltages. For the Branch Table, the data extracted are the From Bus, To Bus and Line Impedance in per-unit (PU). The relevant columns for each data are pre-programmed as the Excel follows the CDF format, and as there is no column headers present, making a search function non-viable.

For the initial design of the data extraction, the “-999” marker was found using built-in commands, but this opened doors to errors as there are multiple such markers and sometimes the wrong one is picked up, leading to more than one table to be extracted at once. To remedy this, the iterative approach was chosen as it would only pick up the first marker.

The SBase value is in the header of the Excel file. Once verified, the header is split according to white spaces, separating each word and each value into its own individual list element. The SBase is expected to be located right after the “RYERSON UNIVERSITY” string, as such, is expected there. However, as white spaces might not be consistent, a while loop is also added to iteratively search all the elements after the “UNIVERSITY” until it encounters the first number which will then be extracted as the SBase value.

### ***3.2.3 Branch Reorder***

For simple linear radial systems, the network leaves the substation and passes one bus after another until it reaches the end of the network. However, most subsystems are not simple and contain branches which need to be accounted for. As such, the branch data is rearranged such that the branches are moved to the top of the table as they occur instead of the bottom of the table. This is accomplished with the help of NumPy and the FROM and TO columns of the branch table.

The function initializes by creating a new array for the rearranged branch table. It iteratively goes down the original table, copies over the values into the new array while checking each bus in the FROM column on whether a duplicate of that same bus exists. A duplicate will only exist if there is a branch, as a single FROM bus will result in multiply TO buses. Once a duplicate is found, the copying will start from the branch index instead of the previous index and continue copying until we reach the end of the branch, and then return back to the previous index to resume operation. This ensures that if a branch also has a branch, it will be accounted for, as well as if multiple branches exist, they will be moved to the top as they occur.

Once all the data has been rearranged, the branch table will be returned, and both the reordered branch table and the bus table is passed onto the calculation engine.

### ***3.2.4 Data Exporter***

If the calculation engine can do voltage convergence, the calculated data is returned back to the Data Parser. The Data Exporter takes this and rearranges it such that the following outputs are generated:

1. Voltages of each bus in per-unit (PU), magnitude (V) and angle (degrees)
2. Power flow and power loss in KW, KVAR and KVA
3. Total power loss for line impedances
4. Power Injection at each bus in KW, KVAR and KVA
5. Error percentages for each iteration until convergence

The generated outputs are then converted back into Panda DataFrames with column headers, and finally saved in the output folder. If the output folder does not exist, it will be created such that the Excel can be saved. The naming convention of the Excel is “**Bus Number\_DDM-MYYYY HHMM.xlsx**”. This naming convention was chosen such that the total number of buses are located, and the time is to ensure that each output is unique.

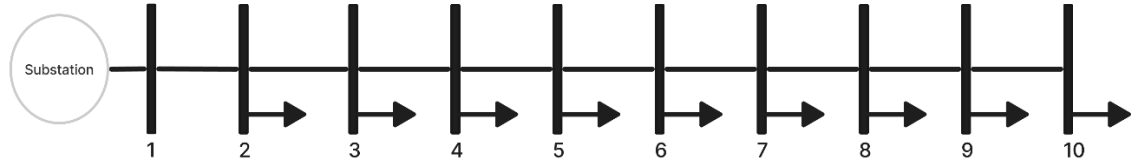
## **3.3 Calculation Engine**

The design and implementation of the calculation engine is the most important part of this project. The calculation engine acts as the brain of the entire system, taking the input data from the data parser and the tolerance value provided by the user, and calculating the voltages, current, power loss, injection, and flow.

After the initialization, the calculation engine follows Kirchhoff’s Laws and employs the forward and backward sweep approach. The forward sweep calculates the voltages on each bus, whereas the backward sweep calculates the current using the updated voltage values if the voltage convergence fails. The forward and backward sweep method is related to the Ladder Network Theory. To determine voltage convergence, the value of the final bus is compared against the value from the previous iteration. Convergence happens when the error tolerance is satisfied. Once all the voltages and current has been calculated, it is passed to the power calculations before returning to the Data Parser for exporting.

### 3.3.1 First Iteration Design

The initial design of the calculation engine was prepared for radial systems without any branches. The slack bus was set to be located at the first bus and the program was run with an error tolerance of 0.0001 for a simple 3-bus system which was then extended to a 10-bus system. For both systems, the initial design was successful in calculating the values but required a total of 4 iterations. However, a radial system is not only linear in nature and has branches, and the calculation engine must be able to account for that.



**Figure 3.3.1:** 10-Bus Radial System.

### 3.3.2 Final Iteration Design

The final design of the calculation engine involved major modifications to both the forward and backward sweeps. Expecting that there will be branches, the system has been modified such that it will always search for more than one instances of the “FROM” bus, as that would indicate that it’s a duplicate and more than one branch exists for that specific bus.

### 3.3.3 Forward Sweep

The forward sweep calculates the voltages of each bus using the current found in the previous backward sweep or using current as 0 if it was not performed yet. It starts at the source bus, and iteratively moves its way down the network until it reaches the end bus. The duplicate is searched for in the “FROM” column as a duplicate will only exist if that bus has multiple “TO”s, meaning there are multiple connections or branches.

If a duplicate is found and the previous row’s bus doesn’t match the bus in the current row, it is assumed that a branch has been found and the first call of the duplicate bus is searched for, as the voltage of that bus is required to proceed. Once this branch has been processed, the code returns to its original position and continues calculation on the main network until it either encounters another branch or it reaches the end bus.

### 3.3.4 Backward Sweep

The backward sweep calculated the current after the voltage convergence has failed. It begins with the final bus and works its way up to the source node. It is derived by considering the preceding iteration’s bus voltages. It follows equations (2) and (3) as the current calculated need to account for the load current as well as the current flowing into the next bus as well.

As KCL is to be considered, the duplicate is also searched for in the “FROM” column while performing the backward sweep, and when a duplicate is encountered, the backward sweep skips that bus and starts at the next branch of that duplicate. Once all the branches have been accounted for, their currents are all added to find the current flowing out of the duplicate bus. This process is repeated until the current for the source bus is found.

### ***3.3.5 Power Loss***

Once the voltage has converged, the resulting output array is passed to the power loss calculation to find the loss on each line. This is accomplished using equation (4). As branches are an issue, for the first bus for each loss calculation, the duplicate is always searched for to ensure that the initial voltage is always the correct one. It follows the same method for finding the duplicate as the backward sweep.

### ***3.3.6 Power Injection***

The power injected into the bus follows equation (6). Like the power loss, the branches need to be accounted for, as such, the duplicates are always searched for to ensure that the correct initial bus is used.

### ***3.3.7 Power Flow***

The power flow for each line follows equation (7) and (8). It subtracts the power loss and load loss for each bus to find the power flowing between one bus to the other. Similarly, it needs to account for the branches and duplicates are always searched for.

## 4 Test System and Analysis

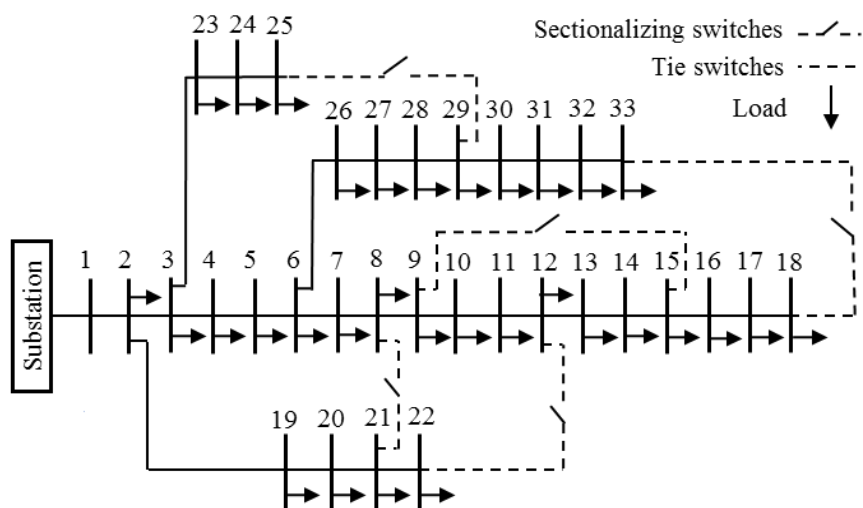
The proposed ladder iterative load flow algorithm is applied to the IEEE 33-bus network as well as a IEEE 69-bus network. Both the networks contain tie switches and sectional switches which are not accounted for in the system. To test the efficiency of the program both networks were tested and convergence was reached. The power flow, total loss and the error per iteration are shown below for each of the systems under test. The different systems were also tested with various different error tolerance values to ensure convergence and sanity check for the software. For every test, the calculated power injection is below the maximum line capacity, confirming that the calculated values match the requirements.

The CDF data of the 33 and the 69 bus are located in the appendix, as well as the calculated voltages, line loss and the power injection.



**Figure 4.1: Software Flowchart**

### 4.1.1 33 Bus System



**Figure 4.1.1:** IEEE CDF 33 Bus System

**Table 4.1.1:** Power Flow for the 33-Bus System

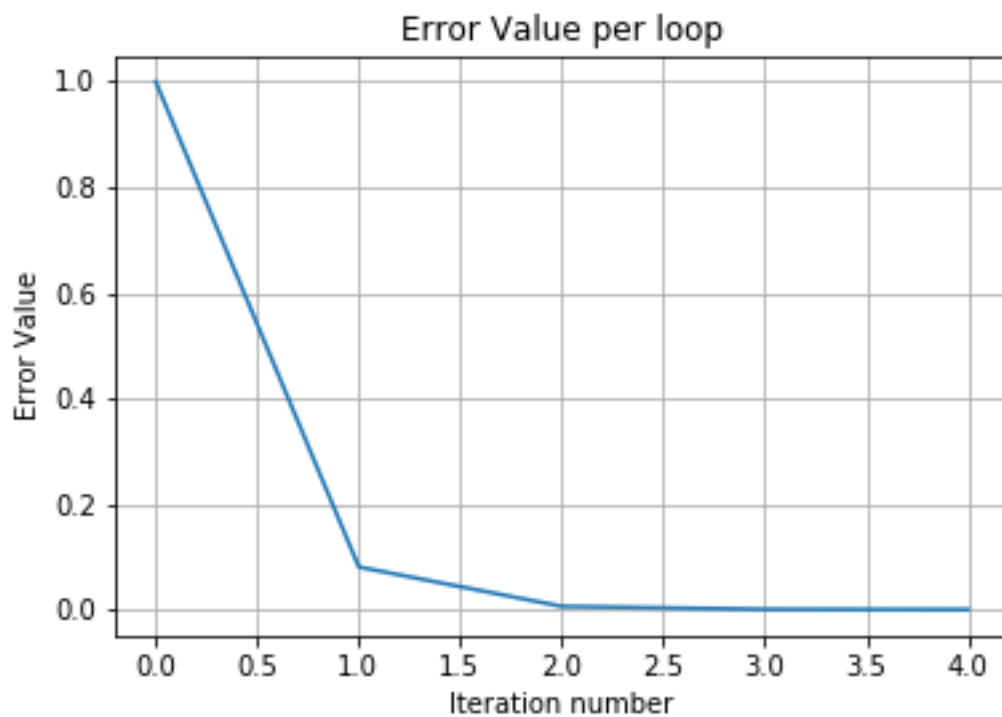
Bus From – Bus To	Real Power Flow (KW)	Reactive Power Flow (KVAR)	Apparent Power Flow (KVA)
1 - 2	3805.361	2368.864	4482.442
2 - 3	3763.571	2362.480	4443.623
3 - 4	3252.531	2091.267	3866.828
4 - 5	2264.225	1634.504	2792.547
5 - 6	2105.980	1531.489	2603.960
6 - 7	1904.066	1415.161	2372.372
7 - 8	888.483	419.958	982.734
8 - 9	824.307	396.956	914.908
9 - 10	620.751	294.435	687.040
10 - 11	575.199	264.252	632.995
11 - 12	499.319	238.947	553.548
12 - 13	451.655	206.851	496.769
13 - 14	330.927	125.892	354.065
14 - 15	330.572	160.576	367.509
15 - 16	210.295	70.373	221.757
16 - 17	150.045	60.038	161.611
17 - 18	59.994	19.997	63.239
2 - 19	3815.200	2388.710	4501.298
19 - 20	270.144	120.175	295.669
20 - 21	180.043	80.058	197.040
21 - 22	90.000	40.000	98.488
3 - 23	3299.253	2129.227	3926.663
23 - 24	511.282	251.005	569.572



24 - 25	419.995	199.998	465.182
6 - 26	2043.379	1495.165	2531.979
26 - 27	884.816	945.585	1295.002
27 - 28	813.518	915.622	1224.816
28 - 29	685.686	833.799	1079.530
29 - 30	541.792	281.815	610.704
30 - 31	470.203	740.243	876.955
31 - 32	209.998	110.014	237.070
32 - 33	209.990	99.995	232.583

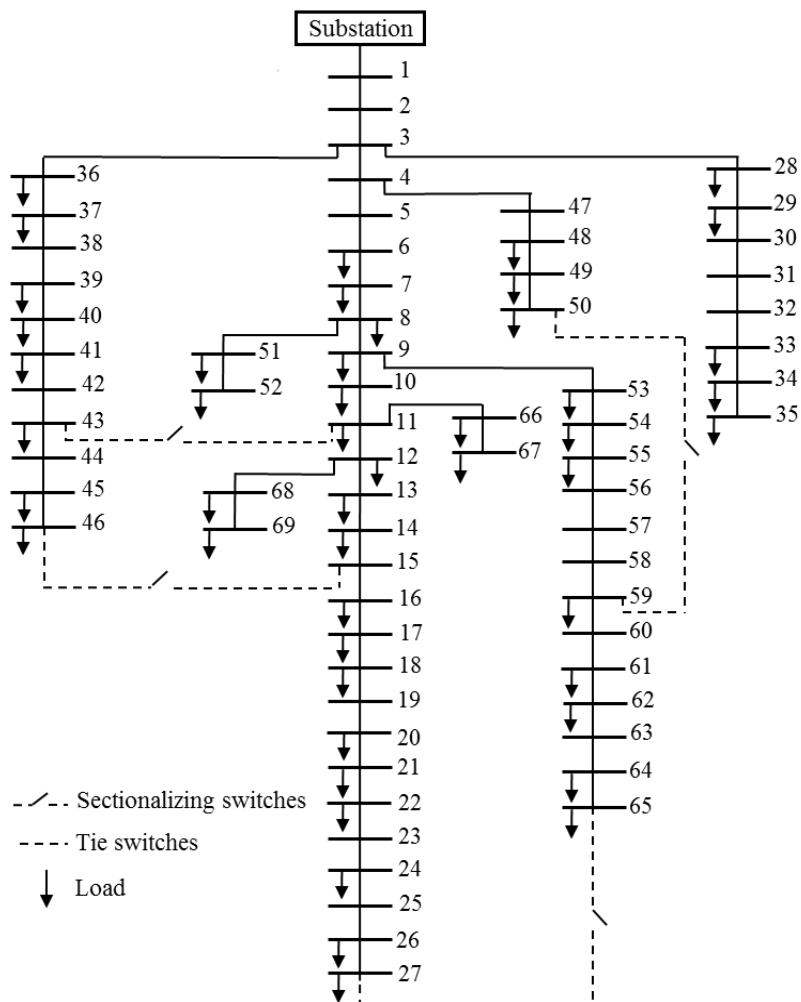
**Table 4.1.2: Total Loss**

<b>Total Real Power Loss (KW)</b>	<b>Total Reactive Power Loss (KVAR)</b>	<b>Total Apparent Power Loss (KVA)</b>	<b>Minimum Voltage (PU)</b>
202.668	135.148	247.028	0.913



**Figure 4.1.2: Error Plot**

### 4.1.2 69 Bus System



**Figure 4.1.3: IEEE CDF 69 Bus System**

**Table 4.1.3: Power Flow 69 Bus System**

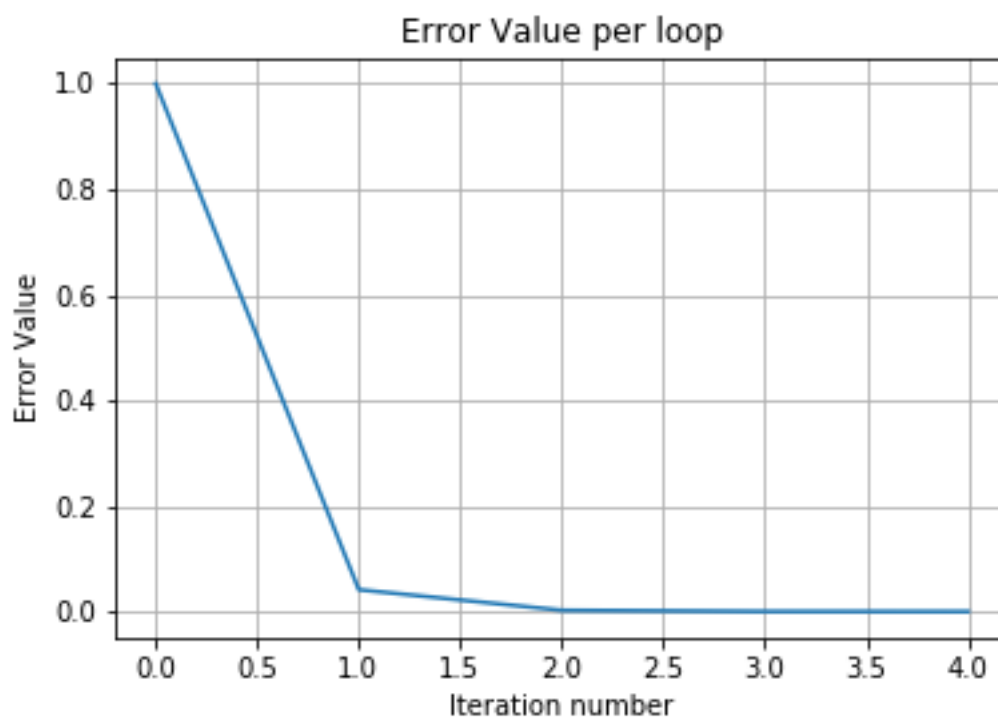
Bus From – Bus To	Real Power Flow (KW)	Reactive Power Flow (KVAR)	Apparent Power Flow (KVA)
1 - 2	4016.621	2795.968	4893.943
2 - 3	4016.546	2795.788	4893.779
3 - 4	4016.351	2795.320	4893.352
4 - 5	3747.132	2598.680	4560.059
5 - 6	2865.544	1970.937	3477.921
6 - 7	2798.404	1928.194	3398.381
7 - 8	2754.311	1898.480	3345.214
8 - 9	2755.537	1898.763	3346.384
9 - 10	2633.664	1815.185	3198.607
10 - 11	629.290	426.839	760.393
11 - 12	599.101	407.116	724.338

12 - 13	553.817	375.693	669.222
13 - 14	351.550	230.775	420.529
14 - 15	350.346	230.877	419.579
15 - 16	296.622	195.304	355.145
16 - 17	281.802	190.198	339.981
17 - 18	236.300	160.197	285.483
18 - 19	236.197	160.163	285.379
19 - 20	175.131	124.542	214.899
20 - 21	62.023	44.106	76.107
21 - 22	170.023	121.006	208.687
22 - 23	61.019	43.506	74.941
23 - 24	28.008	20.002	34.417
24 - 25	56.002	40.000	68.820
25 - 26	14.000	10.000	17.204
26 - 27	14.000	10.000	17.204
3 - 28	3990.546	2777.188	4861.813
28 - 29	55.524	46.608	72.493
29 - 30	55.520	46.607	72.489
30 - 31	29.519	28.006	40.690
31 - 32	29.515	28.005	40.687
32 - 33	15.507	18.002	23.760
33 - 34	20.000	14.000	24.414
34 - 35	9.500	14.000	16.919
3 - 36	3990.545	2777.235	4861.839
36 - 37	159.741	110.568	194.274
37 - 38	159.723	110.548	194.248
38 - 39	109.718	74.992	132.898
39 - 40	109.718	74.992	132.898
40 - 41	108.469	73.935	131.271
41 - 42	85.649	57.911	103.390
42 - 43	78.447	52.608	94.454
43 - 44	84.446	56.908	101.831
44 - 45	39.220	26.300	47.222
45 - 46	39.220	26.300	47.222
4 - 47	3749.045	2600.891	4562.891
47 - 48	771.149	553.280	949.100
48 - 49	463.816	331.183	569.919
49 - 50	384.700	274.500	472.593
8 - 51	2748.410	1894.180	3337.914
51 - 52	40.500	28.300	49.407
9 - 53	2656.306	1829.319	3225.271
53 - 54	1817.469	1255.650	2209.039
54 - 55	1806.396	1249.305	2196.322
55 - 56	1795.207	1243.028	2183.550

56 - 57	1721.530	1209.154	2103.739
57 - 58	1697.044	1200.936	2078.992
58 - 59	1587.540	1125.793	1946.200
59 - 60	1676.870	1194.555	2058.848
60 - 61	318.851	227.415	391.642
61 - 62	1530.739	1092.358	1880.534
62 - 63	318.682	227.343	391.463
63 - 64	59.023	42.008	72.446
64 - 65	226.982	161.987	278.856
11 - 66	728.288	498.838	882.747
66 - 67	18.000	13.000	22.203
12 - 68	535.078	361.109	645.529
68 - 69	27.999	19.999	34.408

**Table 4.1.4: Total Loss**

<b>Total Real Power Loss (KW)</b>	<b>Total Reactive Power Loss (KVAR)</b>	<b>Total Apparent Power Loss (KVA)</b>	<b>Minimum Voltage (PU)</b>
224.922	102.130	250.039	0.909

**Figure 4.1.4: Error Plot**

## 5 Conclusions

### 5.1 Accomplishments

The performance of the ladder-iterative algorithm for radial systems have been presented. The method calculates the voltage and current during the forward and backward sweeps and calculates the power flow at the end of the process to find load-flow of the distribution system. The voltage magnitudes for each node are calculated in the forward sweep and reaches voltage convergence fast. The system was tested for the IEEE 33-bus as well as the IEEE 69-bus. It was found that for both systems, the algorithm can reach fast convergence.

### 5.2 Challenges

One of the primary challenges of this capstone was the COVID-19 global pandemic. Due to this, most of the work had to be done remotely, leading to communication breaks, and raised difficulties during brainstorming session. This also raised difficulties in knowledge sharing, as what one person learned was not easily communicated to the rest of the team. While this was manageable, it slowed down progress significantly. In addition to this, optimizing the code for maximum speed was another challenge that had to be tackled. And finally, not being able to work together led to bugs arising during integration that took significant time to resolve.

### 5.3 Future Work

The current system works great for rather simplistic systems however it can be expanded on even further. The current system does not account for transformers, however that is an expansion to the software which would make simulating the power systems more accurately. Another addition can be to add calculation for an active system in which the load may change within each iteration. And finally, the data parser can be improved further by implementing code for detecting the slack bus and rearranging as such and checking if the system is radial. Through this, a more accurate modelling of a distribution system can be achieved.

## 6 References

- [1] J. Rupa and S. Ganesh, "Power Flow Analysis for Radial Distribution System Using Backward/Forward Sweep Method," *International Journal of Electrical and Computer Engineering*, vol. 8(10), pp. 1628 - 1632, 2014.
- [2] W. F. Tinney and C. E. Hart, "Power Flow Solution by Newton's Method," *IEEE Transactions on Power Apparatus and Systems*, Vols. PAS-86, pp. 1449-1460, 1967.
- [3] W. Tinney and J. Walker, "Direct solutions of sparse network equations by optimally ordered triangular factorization," *Proceedings of the IEEE*, vol. 55, pp. 1801-1809, 1967.
- [4] S. Tripathy, G. D. Prasad, O. Malik and G. Hope, "Load Flow for Ill-Conditioned Power Systems by a Newton like Method," *IEEE Trans.*, p. 3648, 1982.
- [5] F. F. Wu, "Theoretical study of the convergence of the fast decoupled load flow," *IEEE Transactions on Power Apparatus and Systems*, vol. 96, pp. 268-275, 1977.
- [6] O. L. Tortelli, E. M. Lourenço, A. V. Garcia and B. C. Pal, "Fast Decoupled Power Flow to Emerging Distribution Systems via Complex pu Normalization," *IEEE Transactions on Power Systems*, vol. 30, pp. 1351-1358, 2015.
- [7] C. Wadhwa, "Electrical Power Systems," in *New Age International*, 2010.
- [8] R. Berg, E. S. Hawkins and W. W. Pleines, "Mechanized Calculation of Unbalanced Load Flow on Radial Distribution Circuits," *IEEE Transactions on Power Apparatus and Systems*, Vols. PAS-86, pp. 415-421, 1967.
- [9] W. Kersting, "A Method to Teach the Design and Operation of a Distribution System," *IEEE Transactions on Power Apparatus and Systems*, Vols. PAS-103, pp. 1945-1952, 1984.
- [10] M. V. Ellis, "The ladder load-flow method extended to distribution networks," 1994.
- [11] A. K. Al-Awadi, "Load Flow Method for Radial Systems with Distributing Generation," 2008.
- [12] A. Augugliaro and L. Dusonchet, "A backward sweep method for power flow solution in distribution networks," *Electrical Power and Energy Systems*, vol. 32, pp. 271-280, 2010.
- [13] R. A. Stevens, D. T. Rizy and S. L. Purucker, "Performance of conventional power flow routines for real-time distribution automation applications," 01 01 1986.
- [14] A. Arunagiri, B. Venkatesh and K. Ramasamy, "Artificial neural network approach-an application to radial loadflow algorithm," *IEICE Electronics Express*, vol. 3, no. 14, pp. 353-360, 2006.

- [15] K. S. S. Musti and R. B. Ramkhelawan, "Power System Load Flow Analysis Using Microsoft Excel," *Spreadsheets in Education*, vol. 6, p. 1, 2012.
- [16] K. Daware, "Radial, Parallel, Ring Main And Interconnected Distribution Systems," ElectricalEasy, [Online]. Available: <https://www.electricaleasy.com/2018/02/radial-parallel-ring-main-interconneted-distribution.html>.
- [17] G. W., "Common Format For Exchange of Solved Load Flow Data," *IEEE Transactions on Power Apparatus and Systems*, Vols. PAS-92(6), pp. 1916-1925, 1973.
- [18] R. Christie, "IEEE Common Data Format," University of Washington, 1993. [Online]. Available: <http://labs.ece.uw.edu/pstca/pf30/ieee30cdf.txt>  
<https://labs.ece.uw.edu/pstca/formats/cdf.txt>.
- [19] W. H. Kersting, *Distribution System Modeling and Analysis*, CRC Press, 2018.
- [20] A. AppaRao and M. W. Babu, "Forward Sweeping Method for Solving Radial Distribution Networks," *IJAREEIE*, vol. 2, no. 9, 2013.
- [21] N. Gnanasekaran, S. Chandramohan, S. T. D and P. .. Kumar, "Maximum Cost Saving Approach for Optimal Capacitor Placement in Radial Distribution Systems using Modified ABC Algorithm,," *International Journal on Electrical Engineering and Informatics*, vol. 7, no. 4, p. 665–678, 2015.
- [22] N/A, "PySimpleGUI," [Online]. Available: <https://pysimplegui.readthedocs.io/en/latest/cookbook/>. [Accessed October 2021].
- [23] J. D. Glover, T. J. Overbye and M. S. Sarma, *Power System Analysis & Design*, Cengage Learning, 2017.

## Appendices

A	Input CDF for Testing
B	Voltage Output PU
C	Power Flow
D	Power Injection
E	Error Percentage for Each Iteration



## Appendix A: Input CDF For Testing

### A.1 IEEE CDF 33 Bus System

11/13/21 RYERSON UNIVERSITY 100.0 2021 W IEEE 33 Bus Test Case. <b>Slack Bus: 3</b>			
BUS DATA FOLLOWS			
Bus No.	Nominal Load (P and Q)		V_Base
Bus 1	0.000	0.000	12.66
Bus 2	0.100	0.060	12.66
Bus 3	0.090	0.040	12.66
Bus 4	0.120	0.080	12.66
Bus 5	0.060	0.030	12.66
Bus 6	0.060	0.020	12.66
Bus 7	0.200	0.100	12.66
Bus 8	0.200	0.100	12.66
Bus 9	0.060	0.020	12.66
Bus 10	0.060	0.020	12.66
Bus 11	0.045	0.030	12.66
Bus 12	0.060	0.035	12.66
Bus 13	0.060	0.035	12.66
Bus 14	0.120	0.080	12.66
Bus 15	0.060	0.010	12.66
Bus 16	0.060	0.020	12.66
Bus 17	0.060	0.020	12.66
Bus 18	0.090	0.040	12.66
Bus 19	0.090	0.040	12.66
Bus 20	0.090	0.040	12.66
Bus 21	0.090	0.040	12.66
Bus 22	0.090	0.040	12.66
Bus 23	0.090	0.050	12.66
Bus 24	0.420	0.200	12.66
Bus 25	0.420	0.200	12.66
Bus 26	0.060	0.025	12.66
Bus 27	0.060	0.025	12.66
Bus 28	0.060	0.020	12.66
Bus 29	0.120	0.070	12.66
Bus 30	0.200	0.600	12.66
Bus 31	0.150	0.070	12.66
Bus 32	0.210	0.100	12.66
Bus 33	0.060	0.040	12.66

BRANCH DATA FOLLOWS			
1	2	0.0575	0.0293
2	3	0.3076	0.1567
3	4	0.2284	0.1163
4	5	0.2378	0.1211
5	6	0.5110	0.4411
6	7	0.1168	0.3861
7	8	0.4439	0.1467
8	9	0.6426	0.4617
9	10	0.6514	0.4617
10	11	0.1227	0.0406
11	12	0.2336	0.0810
12	13	0.9159	0.7206
13	14	0.3379	0.4448
14	15	0.3687	0.3282
15	16	0.4656	0.3400
16	17	0.8042	1.0738
17	18	0.4567	0.3581
2	19	0.1023	0.0976
19	20	0.9385	0.8457
20	21	0.2555	0.2985
21	22	0.4423	0.5848
3	23	0.2815	0.1924
23	24	0.5603	0.4424
24	25	0.5590	0.4374
6	26	0.1267	0.0645
26	27	0.1773	0.0903
27	28	0.6607	0.5826
28	29	0.5018	0.4371
29	30	0.3166	0.1613
30	31	0.6080	0.6008
31	32	0.1937	0.2258
32	33	0.2128	0.3308

## A.2 IEEE CDF 69 Bus System

03/28/22 RYERSON UNIVERSITY 100.0 2022 W IEEE 69 Bus Test Case: <b>Slack Bus = 3</b>			
BUS DATA FOLLOWS			
Bus 1	0.000	0.000	12.66
Bus 2	0.000	0.000	12.66
Bus 3	0.000	0.000	12.66
Bus 4	0.000	0.000	12.66
Bus 5	0.000	0.000	12.66
Bus 6	0.003	0.002	12.66
Bus 7	0.040	0.030	12.66
Bus 8	0.075	0.054	12.66

Bus 9	0.030	0.022	12.66
Bus 10	0.028	0.019	12.66
Bus 11	0.145	0.104	12.66
Bus 12	0.145	0.104	12.66
Bus 13	0.008	0.005	12.66
Bus 14	0.008	0.006	12.66
Bus 15	0.000	0.000	12.66
Bus 16	0.046	0.030	12.66
Bus 17	0.060	0.035	12.66
Bus 18	0.060	0.035	12.66
Bus 19	0.000	0.000	12.66
Bus 20	0.001	0.001	12.66
Bus 21	0.114	0.081	12.66
Bus 22	0.005	0.004	12.66
Bus 23	0.000	0.000	12.66
Bus 24	0.028	0.020	12.66
Bus 25	0.000	0.000	12.66
Bus 26	0.014	0.010	12.66
Bus 27	0.014	0.010	12.66
Bus 28	0.026	0.019	12.66
Bus 29	0.026	0.019	12.66
Bus 30	0.000	0.000	12.66
Bus 31	0.000	0.000	12.66
Bus 32	0.000	0.000	12.66
Bus 33	0.014	0.010	12.66
Bus 34	0.010	0.014	12.66
Bus 35	0.006	0.004	12.66
Bus 36	0.026	0.019	12.66
Bus 37	0.026	0.019	12.66
Bus 38	0.000	0.000	12.66
Bus 39	0.024	0.017	12.66
Bus 40	0.024	0.017	12.66
Bus 41	0.001	0.001	12.66
Bus 42	0.000	0.000	12.66
Bus 43	0.006	0.004	12.66
Bus 44	0.000	0.000	12.66
Bus 45	0.039	0.026	12.66
Bus 46	0.039	0.026	12.66
Bus 47	0.000	0.000	12.66
Bus 48	0.079	0.056	12.66
Bus 49	0.385	0.275	12.66
Bus 50	0.385	0.275	12.66
Bus 51	0.041	0.028	12.66
Bus 52	0.004	0.003	12.66
Bus 53	0.004	0.004	12.66
Bus 54	0.026	0.019	12.66
Bus 55	0.024	0.017	12.66
Bus 56	0.000	0.000	12.66

Bus 57	0.000	0.000	12.66
Bus 58	0.000	0.000	12.66
Bus 59	0.100	0.072	12.66
Bus 60	0.000	0.000	12.66
Bus 61	1.244	0.888	12.66
Bus 62	0.032	0.023	12.66
Bus 63	0.000	0.000	12.66
Bus 64	0.227	0.162	12.66
Bus 65	0.059	0.042	12.66
Bus 66	0.018	0.013	12.66
Bus 67	0.018	0.013	12.66
Bus 68	0.028	0.020	12.66
Bus 69	0.028	0.020	12.66

BRANCH DATA FOLLOWS			
1	2	0.0003	0.0007
2	3	0.0003	0.0007
3	4	0.0009	0.0022
4	5	0.0157	0.0183
5	6	0.2284	0.1163
6	7	0.2378	0.1211
7	8	0.0575	0.0293
8	9	0.0308	0.0157
9	10	0.5110	0.1689
10	11	0.1168	0.0386
11	12	0.4439	0.1467
12	13	0.6426	0.2121
13	14	0.6514	0.2153
14	15	0.6601	0.2181
15	16	0.1227	0.0406
16	17	0.2336	0.0772
17	18	0.0029	0.0010
18	19	0.2044	0.0676
19	20	0.1314	0.0431
20	21	0.2131	0.0704
21	22	0.0087	0.0029
22	23	0.0993	0.0328
23	24	0.2161	0.0714
24	25	0.4672	0.1544
25	26	0.1927	0.0637
26	27	0.1081	0.0357
3	28	0.0027	0.0067
28	29	0.0399	0.0976
29	30	0.2482	0.0820
30	31	0.0438	0.0145
31	32	0.2190	0.0724
32	33	0.5235	0.1757

33	34	1.0657	0.3523
34	35	0.9197	0.3040
3	36	0.0027	0.0067
36	37	0.0399	0.0976
37	38	0.0657	0.0767
38	39	0.0190	0.0221
39	40	0.0011	0.0013
40	41	0.4544	0.5309
41	42	0.1934	0.2260
42	43	0.0256	0.0298
43	44	0.0057	0.0072
44	45	0.0679	0.0857
45	46	0.0006	0.0007
4	47	0.0021	0.0052
47	48	0.0531	0.1300
48	49	0.1808	0.4424
49	50	0.0513	0.1255
8	51	0.0579	0.0295
51	52	0.2071	0.0695
9	53	0.1086	0.0553
53	54	0.1267	0.0645
54	55	0.1773	0.0903
55	56	0.1755	0.0894
56	57	0.9920	0.3330
57	58	0.4890	0.1641
58	59	0.1898	0.0628
59	60	0.2409	0.0731
60	61	0.3166	0.1613
61	62	0.0608	0.0309
62	63	0.0905	0.0460
63	64	0.4433	0.2258
64	65	0.6495	0.3308
11	66	0.1255	0.0381
66	67	0.0029	0.0009
12	68	0.4613	0.1525
68	69	0.0029	0.0010

## Appendix B: Voltage Output

### B.1 33 Bus System

Bus No	Voltage Magnitude (PU)	Voltage Magnitude (V)	Voltage Angle (Degree)
1	1	12.66	0

2	0.99703	12.6224	0.0145
3	0.98294	12.44402	0.096
4	0.97546	12.34932	0.16165
5	0.96806	12.25564	0.22831
6	0.94966	12.0227	0.13391
7	0.94617	11.97851	-0.09642
8	0.94133	11.91724	-0.06034
9	0.93506	11.83786	-0.13342
10	0.92925	11.7643	-0.19594
11	0.92838	11.75329	-0.18869
12	0.92688	11.7343	-0.17849
13	0.92076	11.65682	-0.2698
14	0.9185	11.62821	-0.34848
15	0.91708	11.61023	-0.38617
16	0.91572	11.59302	-0.40942
17	0.91369	11.56732	-0.48669
18	0.91308	11.55959	-0.49628
19	0.99651	12.61582	0.00368
20	0.99293	12.57049	-0.06331
21	0.99222	12.56151	-0.08267
22	0.99159	12.55353	-0.10301
23	0.97935	12.39857	0.06501
24	0.97268	12.31413	-0.02371
25	0.96936	12.2721	-0.0674
26	0.94773	11.99826	0.1734
27	0.94517	11.96585	0.22953
28	0.93373	11.82102	0.31244
29	0.92551	11.71696	0.39037
30	0.92195	11.67189	0.49561
31	0.91779	11.61922	0.41123
32	0.91688	11.6077	0.38818
33	0.91659	11.60403	0.38045

## ***B.2 69 Bus System***

<b>Bus No</b>	<b>Voltage Magnitude (PU)</b>	<b>Voltage Magnitude (V)</b>	<b>Voltage Angle (Degree)</b>
1	1	12.66	0
2	0.99997	12.65962	-0.00122
3	0.99993	12.65911	-0.00245

4	0.99984	12.65797	-0.00588
5	0.99902	12.64759	-0.0185
6	0.99009	12.53454	0.04928
7	0.9808	12.41693	0.12104
8	0.97858	12.38882	0.13824
9	0.97745	12.37452	0.14705
10	0.97245	12.31122	0.23173
11	0.97135	12.29729	0.25051
12	0.96819	12.25729	0.30325
13	0.96527	12.22032	0.34958
14	0.96237	12.1836	0.39582
15	0.9595	12.14727	0.4418
16	0.95897	12.14056	0.45037
17	0.95809	12.12942	0.46453
18	0.95808	12.12929	0.46468
19	0.95761	12.12334	0.47322
20	0.95732	12.11967	0.47876
21	0.95684	12.11359	0.48764
22	0.95683	12.11347	0.48777
23	0.95676	12.11258	0.48911
24	0.9566	12.11056	0.49201
25	0.95643	12.1084	0.49516
26	0.95636	12.10752	0.49645
27	0.95634	12.10726	0.49682
28	0.99993	12.65911	-0.00266
29	0.99986	12.65823	-0.0047
30	0.99976	12.65696	-0.0021
31	0.99975	12.65684	-0.00165
32	0.99966	12.6557	0.00065
33	0.99946	12.65316	0.00608
34	0.99923	12.65025	0.01395
35	0.99916	12.64937	0.01502
36	0.99992	12.65899	-0.00296
37	0.99975	12.65684	-0.00937
38	0.99959	12.65481	-0.01179
39	0.99954	12.65418	-0.01249
40	0.99954	12.65418	-0.01252
41	0.99884	12.64531	-0.02351
42	0.99855	12.64164	-0.02816
43	0.99851	12.64114	-0.02877
44	0.9985	12.64101	-0.02892
45	0.99841	12.63987	-0.03073
46	0.9984	12.63974	-0.03074
47	0.99979	12.65734	-0.00769
48	0.99854	12.64152	-0.05252
49	0.9947	12.5929	-0.19162
50	0.99415	12.58594	-0.21143
51	0.97854	12.38832	0.13853

52	0.97853	12.38819	0.13872
53	0.97466	12.3392	0.16897
54	0.97142	12.29818	0.19458
55	0.96694	12.24146	0.23017
56	0.96258	12.18626	0.26513
57	0.9401	11.90167	0.66166
58	0.92904	11.76165	0.86421
59	0.92477	11.70759	0.94518
60	0.91974	11.64391	1.04967
61	0.91235	11.55035	1.11872
62	0.91206	11.54668	1.12144
63	0.91167	11.54174	1.12508
64	0.90977	11.51769	1.14294
65	0.90919	11.51035	1.14832
66	0.97129	12.29653	0.25166
67	0.97129	12.29653	0.25167
68	0.96786	12.25311	0.30931
69	0.96786	12.25311	0.30933

## Appendix C: Power Flow

### C.1 33 Bus System

Bus Con- nection	Real Power Flow (KW)	Reactive Power Flow (KVAR)	Apparent Power Flow (KVA)	Real Power Loss (KW)	Reactive Power Loss (KVAR)	Apparent Power Loss (KVA)
1 - 2	3805.361	2368.864	4482.442	12.23442	6.23424	13.73123
2 - 3	3763.571	2362.48	4443.623	51.78976	26.38314	58.12271
3 - 4	3252.531	2091.267	3866.828	19.90315	10.13457	22.33483
4 - 5	2264.225	1634.504	2792.547	18.6996	9.5228	20.98473
5 - 6	2105.98	1531.489	2603.96	38.24677	33.01497	50.52528
6 - 7	1904.066	1415.161	2372.372	1.91444	6.32847	6.6117
7 - 8	888.4832	419.9575	982.7343	4.83814	1.59891	5.0955
8 - 9	824.3073	396.9565	914.9081	4.18003	3.0033	5.14708
9 - 10	620.751	294.435	687.0398	3.56084	2.52386	4.36457
10 - 11	575.1987	264.2523	632.9951	0.55384	0.18326	0.58337
11 - 12	499.3193	238.9475	553.5482	0.8811	0.30552	0.93256
12 - 13	451.6546	206.8509	496.7687	2.66603	2.09754	3.39225
13 - 14	330.9273	125.8922	354.0646	0.72908	0.95974	1.20527
14 - 15	330.5723	160.5758	367.5087	0.35692	0.31771	0.47784
15 - 16	210.2949	70.37339	221.7575	0.28143	0.20551	0.34848
16 - 17	150.0455	60.03795	161.6113	0.25161	0.33596	0.41973
17 - 18	59.99448	19.99719	63.23943	0.05313	0.04166	0.06752
2 - 19	3815.2	2388.71	4501.298	0.16092	0.15352	0.2224



19 - 20	270.1441	120.1753	295.6686	0.83216	0.74988	1.12019
20 - 21	180.0434	80.0576	197.0402	0.10076	0.11772	0.15495
21 - 22	89.99984	39.99995	98.48841	0.04363	0.05769	0.07233
3 - 23	3299.253	2129.227	3926.663	3.18141	2.17444	3.85351
23 - 24	511.2815	251.0049	569.572	5.14374	4.06138	6.55384
24 - 25	419.9947	199.9978	465.1824	1.28735	1.00731	1.63461
6 - 26	2043.379	1495.165	2531.979	2.6016	1.32442	2.91932
26 - 27	884.8158	945.5854	1295.002	3.32839	1.69517	3.73521
27 - 28	813.5176	915.6221	1224.816	11.29942	9.96374	15.06496
28 - 29	685.6855	833.7992	1079.53	7.8334	6.82339	10.3885
29 - 30	541.7923	281.8153	610.7035	3.89487	1.98434	4.37122
30 - 31	470.2026	740.2425	876.9547	1.59364	1.57477	2.24045
31 - 32	209.9977	110.0136	237.0697	0.21315	0.24847	0.32737
32 - 33	209.99	99.99535	232.5831	0.01317	0.02047	0.02434

Total Real Power Losses (KW)	Total Reactive Power Losses (KVAR)	Total Apparent Power Losses (KVA)
202.6679	135.1479	247.0279

## C.2 69 Bus System

Bus Con- nection	Real Power Flow (KW)	Reactive Power Flow (KVAR)	Apparent Power Flow (KVA)	Real Power Loss (KW)	Reactive Power Loss (KVAR)	Apparent Power Loss (KVA)
1 - 2	4016.621	2795.968	4893.943	0.07472	0.17933	0.19428
2 - 3	4016.546	2795.788	4893.779	0.07472	0.17933	0.19428
3 - 4	4016.351	2795.32	4893.352	0.19492	0.4678	0.50679
4 - 5	3747.132	2598.68	4560.059	1.93618	2.26787	2.98195
5 - 6	2865.544	1970.937	3477.921	28.2327	14.37862	31.68328
6 - 7	2798.404	1928.194	3398.381	29.34029	14.94345	32.92657
7 - 8	2754.311	1898.48	3345.214	6.89262	3.51359	7.73651
8 - 9	2755.537	1898.763	3346.384	3.37405	1.71782	3.78617
9 - 10	2633.664	1815.185	3198.607	4.77269	1.57749	5.02663
10 - 11	629.2902	426.8388	760.393	1.01376	0.33521	1.06774
11 - 12	599.1014	407.1156	724.338	2.18918	0.72347	2.30563
12 - 13	553.8166	375.6927	669.2218	1.28423	0.42392	1.35239
13 - 14	351.5504	230.7754	420.5294	1.2454	0.41156	1.31164
14 - 15	350.3459	230.8775	419.5791	1.20455	0.39803	1.26861
15 - 16	296.6222	195.3036	355.1453	0.22383	0.074	0.23575
16 - 17	281.8018	190.1976	339.9814	0.32039	0.10594	0.33745
17 - 18	236.2998	160.1972	285.4834	0.0026	0.00089	0.00275
18 - 19	236.1966	160.1634	285.3789	0.10412	0.03442	0.10966
19 - 20	175.1305	124.542	214.8986	0.06693	0.02193	0.07043
20 - 21	62.02314	44.10648	76.10684	0.1074	0.0355	0.11311
21 - 22	170.0226	121.0063	208.6869	0.00054	0.00018	0.00056
22 - 23	61.01917	43.50593	74.94068	0.00514	0.0017	0.00541

23 - 24	28.00806	20.00229	34.41719	0.01118	0.0037	0.01178
24 - 25	56.00201	40.00029	68.82041	0.00605	0.002	0.00637
25 - 26	13.99994	9.9998	17.20449	0.00249	0.00082	0.00263
26 - 27	13.99958	9.99967	17.20412	0.00035	0.00012	0.00037
3 - 28	3990.546	2777.188	4861.813	0.0003	0.00073	0.00079
28 - 29	55.52364	46.60786	72.49253	0.0021	0.00513	0.00555
29 - 30	55.51953	46.6065	72.48851	0.00411	0.00136	0.00433
30 - 31	29.5188	28.00626	40.69042	0.00073	0.00024	0.00076
31 - 32	29.51517	28.00506	40.68696	0.00363	0.0012	0.00382
32 - 33	15.5065	18.00215	23.75982	0.00867	0.00291	0.00915
33 - 34	20.00048	14.00016	24.4136	0.00602	0.00199	0.00634
34 - 35	9.5	14	16.91892	0.00048	0.00016	0.0005
3 - 36	3990.545	2777.235	4861.839	0.00141	0.00345	0.00372
36 - 37	159.7406	110.5682	194.274	0.01508	0.03687	0.03984
37 - 38	159.7233	110.5479	194.2482	0.01732	0.02023	0.02664
38 - 39	109.7183	74.99208	132.8982	0.005	0.00584	0.00769
39 - 40	109.7181	74.99185	132.8979	0.0002	0.00023	0.00031
40 - 41	108.4694	73.93494	131.2707	0.04871	0.05691	0.07492
41 - 42	85.64926	57.91143	103.3902	0.02012	0.02351	0.03094
42 - 43	78.44661	52.60833	94.45373	0.00266	0.0031	0.00409
43 - 44	84.4461	56.90768	101.8314	0.00051	0.00065	0.00083
44 - 45	39.22001	26.30001	47.22181	0.00608	0.00767	0.00978
45 - 46	39.22	26.29999	47.22179	0.00001	0.00002	0.00002
4 - 47	3749.045	2600.891	4562.891	0.02329	0.05753	0.06206
47 - 48	771.1494	553.2804	949.0999	0.58281	1.42656	1.54102
48 - 49	463.8159	331.1835	569.919	1.63351	3.99696	4.31787
49 - 50	384.7	274.4999	472.5932	0.1159	0.28354	0.30631
8 - 51	2748.41	1894.18	3337.914	0.00176	0.0009	0.00197
51 - 52	40.49957	28.29963	49.40733	0.00004	0.00001	0.00005
9 - 53	2656.306	1829.319	3225.271	5.78047	2.94339	6.48671
53 - 54	1817.469	1255.65	2209.039	6.71053	3.41807	7.5309
54 - 55	1806.396	1249.305	2196.322	9.12346	4.6452	10.23794
55 - 56	1795.207	1243.028	2183.55	8.7889	4.47725	9.8636
56 - 57	1721.53	1209.154	2103.739	49.67778	16.67486	52.40165
57 - 58	1697.044	1200.936	2078.992	24.48583	8.21714	25.82784
58 - 59	1587.54	1125.793	1946.2	9.50439	3.14313	10.01063
59 - 60	1676.87	1194.555	2058.848	10.66953	3.23872	11.15025
60 - 61	318.8513	227.4149	391.6423	14.02431	7.14342	15.73879
61 - 62	1530.739	1092.358	1880.534	0.11204	0.05705	0.12573
62 - 63	318.6818	227.3434	391.4628	0.13491	0.06867	0.15138
63 - 64	59.02276	42.00804	72.44558	0.66107	0.33672	0.74189
64 - 65	226.9815	161.9871	278.8556	0.04121	0.02099	0.04624
11 - 66	728.2879	498.8383	882.7474	0.00262	0.0008	0.00274
66 - 67	17.99957	12.99965	22.20305	0.00002	0	0.00002
12 - 68	535.0775	361.1089	645.5289	0.02332	0.00771	0.02456
68 - 69	27.9993	19.99943	34.4084	0.00004	0.00001	0.00004

Total Real Power Losses (KW)	Total Reactive Power Losses (KVAR)	Total Apparent Power Losses (KVA)
224.9219	102.1295	250.039

## Appendix D : Power Injection

### *D.1 33 Bus System*

Bus No	Real Power Injection (KW)	Reactive Power Injection (KVAR)	Apparent Power Injection (KVA)
1	3917.596	2435.098	4612.728
2	3905.361	2428.864	4599.046
3	3392.434	2181.402	4033.252
4	2342.925	1674.027	2879.525
5	2204.226	1584.504	2714.64
6	2105.98	1521.49	2598.092
7	1093.321	521.5564	1211.352
8	888.4873	419.9598	982.739
9	684.3118	316.9589	754.1522
10	620.7525	294.4356	687.0415
11	560.2004	274.253	623.7301
12	514.3206	243.9484	569.2421
13	451.6564	206.852	496.7708
14	390.9293	170.8935	426.6501
15	270.5764	90.5789	285.3351
16	210.2971	80.37391	225.1329
17	150.0476	60.03885	161.6136
18	89.99669	39.99816	98.4848
19	360.9762	160.9252	395.2224
20	270.1442	120.1753	295.6687
21	180.0435	80.05764	197.0403
22	89.99992	39.99997	98.4885
23	936.4253	455.0663	1041.142
24	841.282	401.0051	931.966
25	419.9973	199.9989	465.1852
26	948.1442	972.2806	1358.053
27	884.817	945.5859	1295.003
28	813.5189	910.6226	1221.084
29	745.6872	883.7997	1156.353
30	621.7962	811.8173	1022.584
31	420.2109	210.2621	469.8801
32	270.0032	140.0158	304.1482

## D.2 69 Bus System

Bus No	Real Power Injection (KW)	Reactive Power Injection (KVAR)	Apparent Power Injection (KVA)
1	4016.696	2796.147	4894.107
2	4016.621	2795.968	4893.943
3	4016.546	2795.788	4893.779
4	3749.068	2600.948	4562.943
5	2896.377	1987.516	3512.722
6	2868.144	1973.137	3481.31
7	2836.204	1955.994	3445.281
8	2788.911	1922.481	3387.323
9	2666.437	1835.763	3237.269
10	775.304	531.174	939.8096
11	746.2906	511.8391	904.9469
12	563.1009	381.1166	679.9503
13	360.7958	236.687	431.5024
14	351.5505	231.2755	420.8041
15	342.346	225.3776	409.8729
16	342.1222	225.3036	409.6453
17	296.3024	195.1981	354.8203
18	236.3007	160.1978	285.4844
19	176.1975	125.1639	216.1286
20	176.1305	125.142	216.0613
21	175.0232	124.5065	214.7905
22	61.02431	43.50763	74.9458
23	56.01924	40.00599	68.8377
24	56.00806	40.00229	68.8265
25	28.00243	20.00062	34.4116
26	27.99993	19.99979	34.4091
27	13.99979	9.99984	17.2044
28	81.52574	65.21299	104.3991
29	55.52364	46.60786	72.4925
30	29.51953	28.0065	40.6911
31	29.5188	28.00626	40.6904
32	29.51517	28.00506	40.687
33	29.5065	28.00215	40.6787
34	15.50048	18.00016	23.7544
35	6	4	7.2111
36	185.7557	129.155	226.2437
37	159.7406	110.5682	194.274
38	133.7233	91.99792	162.3131
39	133.7183	91.99208	162.3057
40	109.7181	74.99185	132.8979
41	85.66938	57.93494	103.42
42	84.44927	56.91143	101.8361
43	84.44661	56.90833	101.8322
44	78.44609	52.60768	94.4529

45	78.44001	52.60001	94.4436
46	39.22	26.3	47.2218
47	850.7322	611.107	1047.472
48	850.1494	609.6804	1046.166
49	769.5159	549.2835	945.4454
50	384.7	274.5	472.5932
51	44.09961	30.99964	53.905
52	3.59996	2.69997	4.5
53	1850.58	1278.068	2249.023
54	1839.519	1271.15	2235.991
55	1803.996	1247.505	2193.325
56	1771.208	1225.828	2154.027
57	1721.53	1209.154	2103.739
58	1697.044	1200.936	2078.992
59	1687.54	1197.793	2069.42
60	1576.876	1122.558	1935.633
61	1562.851	1115.415	1920.066
62	318.8167	227.412	391.6125
63	286.6838	204.3448	352.0574
64	286.0228	204.008	351.3236
65	58.99617	41.99732	72.4177
66	35.99959	25.99965	44.4067
67	17.99979	12.99982	22.2033
68	55.99934	39.99944	68.8177

## Appendix E: Error Percentage per Iteration

### *E.1 33 Bus System*

Iteration Number	Error Percentage
0	1
1	0.080542
2	0.00592
3	0.000455
4	3.5E-05

### *E.2 69 Bus System*

Iteration Number	Error Percentage
0	1
1	0.041246
2	0.002265
3	0.000169
4	1.47E-05