American University of Sharjah

College of Engineering - Department of Electrical Engineering

ELE 371 – Power Systems Analysis: Section 01

Summer 2018

Course Project: Newton-Raphson for Power Flow Analysis

Submitted on: 16th July 2018

Instructor: Dr. Mostafa F. Shaaban

Submitted By:

Name	Student ID
Abdulrahman Nasser	@00063951
Danna Fares	@00064211
Taha Ameen ur Rahman	@00066555

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Power Flow Analysis Using The Newton-Raphson Algorithm

Taha Ameen ur Rahman, Abdulrahman Nasser, and Danna Fares

Department of Electrical Engineering
American University of Sharjah
P. O. Box 26666, Sharjah, UAE.
{b00066555, b00063951, g00064211}@aus.edu

***Abstract*—The Newton-Raphson Algorithm is implemented in MATLAB to analyze bus voltages and angles for a 30-bus and 118-bus system in accordance with the requirements of the semester project for the course ELE 371 - Power Systems Analysis. The objective of this project is to comprehend the load flow problem and analyze large power systems using iterative techniques. The report initiates with a theoretical background describing the concepts that form the backbone of the algorithm, and includes equations on which the algorithm is based. The entire MATLAB code to analyze the bus systems is also provided. The results of the iterative technique are documented and presented in the report. These are compared with simulation results obtained from PowerWorld, after designing, implementing and running the bus system based on the diagram provided in the problem statement of the project. The report discusses the results obtained from both sources and compares them from a comprehensive perspective. The inferences drawn from the observations are also highlighted to verify the agreement between the analytical and simulation results.**

I. Introduction

Power Systems Analysis is the study of generation, transmission and distribution of electric power. The vast discipline encompasses a plethora of topics that have revolutionized the lives of residents across the globe. Today, electricity is a neccessity and not a luxury, and modern machines and devices are heavily reliant on electric power for their functioning. The transition of everyday devices from mechanical to electrical, such as automobiles, bears witness to the advantages of electrical energy.

However, this is a result of over a century of extensive research. While modern endeavors seek to enable two way electrical communication within a system through smart grids, green systems and the like, the subject's roots date back to the nineteenth century. The report outlines the historical events and theoreteical concepts that underly these systems. As aforementioned, power systems analysis is mainly categorized based on the three subsections that comprise it:

1) **Generation:** Machines such as synchronous generators are used to generate electricity.
2) **Transmission:** Transmission Lines are erected to carry this power from site of generation to residential and industrial areas.
3) **Distribution:** The received electrical power must be converted to ther equired levels of voltages and current to be able to safely distribute it based on the requirements.

This project report is concerned with the second step of the process. Transmission comprises an important and crucial step in the said process. Important categories of analysis include power flow analysis and fault analysis [1]. These are documented techniques to describe the system in terms of transmission parameters, so as to control and determine the state of the system at each bus. This is crucial in order to control the entire system from a monitoring room. While fault analysis deals with the study of currents during an unintentional hinderance, power flow analysis is concerned with exactly what its name suggests - the flow of power in transmission lines through different buses. A bus is essentially a large node for a system, and acts as a common meeting point for other lines that emerge from or terminate at it. Hence, the voltage at a bus is fixed and is essentially a phasor that is characterized

by its magnitude and angle [2].

The remainder of this report is organized as follows. Section II deals with historical, theoretical and technical concepts that are required to analyze and implement solutions to the power flow problem. Section **??** provides the MATLAB code that was implemented to achieve the bus voltages and angles. This is followed by Section IV which highlights the systematic implementation of the 30 Bus system on the PowerWorld software - a powerful tool to analyze power systems. Section V provides a comprehensive overview of the results of both analytical and simulation method, and compares the results obtained from each method. Lastly, Section VII concludes the paper.

II. Background

As aforementioned, transmission lines are a vast topic of extensive research, and this is justified by the huge consumption of electric power that requires efficient techniques of generation, transmission and distribution. The complexity of this problem is exacerbated by the unpredictable nature of the loads. The varying nature of loads at the consumption end of the system based on time of the day as well season of the year necessitates constant monitoring and control of the system. This is the result of years of development and research. Following is a short historical recap of important events that lead to modern power systems:

A. History of Power Systems

1) **1870s:** Arc Lamps are used for commercial distribution of electricitiy including street lamps.
2) **1882:** Thomas Edison builds first DC Power System
3) **1886:** William Stanley develops and tests first AC System.
4) **1888:** Nikola Tesla develops polyphase systems and presents several revolutionary details on polyphaser AC motors, generators and transformers.
5) **1893:** First three-phase line goes operational in North America.
6) **1895:** Alternating Current is chosen as the defacto standard for transmission.
7) **1936:** First experimental high voltage direct current (HVDC) transmission line is built in New York.

Modern electrical power systems continue to be based on polyphase AC transmission, although HVDC is an extremely active area of research. The AC transmission in a grid of buses is controlled using constant monitoring of Power Flow through control centers on a micro as well as macroscopic scale. This is the responsibility of the corporations that supply electrical power to areas of interest. Perhaps the most important tool used by such control centers is Power Flow Analysis, which provides solutions to bus voltages and angles at each bus in the system. The data that is collected from this analysis is used to take action accordingly, and is used in the best interest of the general public. The following section provides the theoretical background behind power flow analysis.

B. Power Flow Analysis

Power Flow is the name of techniques that enable engineers to study and analyze the flow of power in a system. This is done through the solution of non-linear equations that capture the important parameters and physical laws of nature that govern the flow of electrical power.

The first step in developing a model for the entire power flow is a comprehension of the type of loads that comprise the system. It is important to keep in mind that the objective of such an effort is to supply the loads with electricity and constant frequency and voltage. Therefore, aggregate models are employed to analyse systems. Two such common models are the constant power and constant impedance model. The constant power model assumes that the load demands the same amount of power undel all conditions, as opposed to the latter model which assumes that loads present the same amount of impedance under all oeprating conditions. This report assumes loads to be constant power loads, which is a reasonable assumption as the bulk of loads in a system attempt to draw constant power so as to meet the rated requirements at which they perform best.

This implies that the system becomes non-linear. Recall that a non-linear function H is one where:

$$H(\alpha_1\mu_1 + \alpha_2\mu_2 \neq \alpha_1 H(\mu_1) + \alpha_2 H(\mu_2)$$

This prevents the application of the superposition priciple to the system and also renders most circuit analysis techniques as ineffective due to the difference in assumptions. Consequently, the solution to such non-linear systems are best achieved numerically rather than analytically. Iterative approaches provide excellent approximations that give the answers within a tolerace level that is specified by the user. The following section discusses the concept behind the same.

C. Iterative Techniques for Non-Linear Systems

Iterative techniques are numerical approximations to non-linear systems that provide reasonable answers in excellent time. They rely on properties of convergence. Two such techniques are briefly mentioned below.

1) Gauss-Seidel Approach: The Gauss-Seidel approach is an iterative technique that is used for power flow analysis. Although it is relatively easy to code and has the ability to function with complex equations, thereby reducing the computational complexity of the problem, it is not commonly employed due to its slow speed of convergence [3]. This is exploited by its counterpart, which achieves quadratic convergence to provide the solution quickly. It is discussed in detail below.

2) Newton-Raphson Approach: The Newton-Raphson approach splits the complex power flow equations into two real equations per unknown variable. This is used to build a system of equations that can be represented using matrices. The concept of sequential linearization is employed to achieve convergence, which in turn yields the required answers.[3] The algorithm is based on the following power flow equations:

$$S_i = P_i + jQ_i = V_i \sum_{k=1}^{n} |V_i||V_k| e^{j\theta_{ik}} (G_{ik} - jB_{ik}) \tag{1}$$

This is resolved using basic complex analysis to yield:

$$\begin{aligned} P_i &= \sum_{k=1}^{n} |V_i||V_k| \left(G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik}\right) \\ &= P_{G_i} - P_{D_i} \end{aligned} \tag{2}$$

$$\begin{aligned} Q_i &= \sum_{k=1}^{n} |V_i||V_k| \left(G_{ik}\sin\theta_{ik} - B_{ik}\cos\theta_{ik}\right) \\ &= Q_{G_i} - Q_{D_i} \end{aligned} \tag{3}$$

In Power Flow Analysis, it is crucial to declare a certain bus as the slack bus. This implies that the voltage and angle of this bus is fixed at $1\angle 0$, and acts as the reference for all other buses.

Let $|V_i|\angle\theta_i$ represent the phasor voltage at Bus i. We create the vector of unknowns and of power flow equations as follows:

$$\mathbf{x} = \begin{bmatrix} \theta_2 \\ \vdots \\ \theta_n \\ |V_2| \\ \vdots \\ |V_n| \end{bmatrix} \quad \mathbf{f(x)} = \begin{bmatrix} P_2(\mathbf{x}) - P_{G_2} + P_{D_2} \\ \vdots \\ P_n(\mathbf{x}) - P_{G_n} + P_{D_n} \\ Q_2(\mathbf{x}) - Q_{G_2} + Q_{D_2} \\ \vdots \\ Q_n(\mathbf{x}) - Q_{G_n} + Q_{D_n} \end{bmatrix} \tag{4}$$

It is interesting to observe that the slack bus is omitted from the above vectors. This is reasonable as its end value will always remain as $1\angle 0$ and consequently, its addition to the system will only increase computational complexity without yielding any beneficial results. The vector **f(x)** is known as the vector of power mismatch equations [3].

The objective of this approach is to find the values of **x** that satisfy 2 and 3, thereby resulting in setting the mismatch vector to the zero vector.

The algorithm for Newton-Raphson is based on sequential linearization. The pseudocode for the same is simple to comprehend and provided below:

Algorithm 1: Newton-Raphson Algorithm

Inputs: x, f(x), J(x)
Steps :
Set $v = 0$;
Make initial guess for $x, x^{(v)}$;
while $|f(x^{(v)})| > \epsilon$ **do**
    Calculate $x^{(v+1)} = x^{(v)} - J(x^{(v)})^{-1} f(x^{(v)})$;
    Set $v = v + 1$;
end

Here, **J(x)** represents the Jacobian, which is the total derivative of a function from an Euclidean

space of dimension m to n. Here, the mapping can be considered as:

$$g : \mathbb{R}^{2n} \rightarrow \mathbb{R}^{2n}$$

such that

$$g(\theta_1, ..., \theta_n, |V_1|, ..., |V_n|) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_{2n}(\mathbf{x}) \end{bmatrix}$$

The Jacobian is thus defined as:

$$\mathbf{J(x)} = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix} \quad (5)$$

Evidently, the Jacobian required computations that can defeat the purpose of quick numerical solutions to the non-linear problem at hand. Fortunately, it is possible to escape these through utilizing general formulae for submatrices inside the Jacobian which are derived to be as follows:

Let

$$J = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \quad (6)$$

where

$$J_{11} = \frac{\partial P(x)}{\partial \theta} \quad J_{12} = \frac{\partial P(x)}{\partial |V|}$$
$$J_{21} = \frac{\partial Q(x)}{\partial \theta} \quad J_{22} = \frac{\partial Q(x)}{\partial |V|} \quad (7)$$

Then, the following relations hold true:
If $p \neq q$,

$$\begin{aligned} J_{pq}^{11} &= |V_p||V_q|(G_{pq}\sin\theta_{pq} - B_{pq}\cos\theta_{pq}) \\ J_{pq}^{21} &= -|V_p||V_q|(G_{pq}\cos\theta_{pq} + B_{pq}\sin\theta_{pq}) \\ J_{pq}^{12} &= |V_p|(G_{pq}\cos\theta_{pq} + B_{pq}\sin\theta_{pq}) \\ J_{pq}^{22} &= |V_p|(G_{pq}\sin\theta_{pq} - B_{pq}\cos\theta_{pq}) \end{aligned} \quad (8)$$

If $p = q$

$$\begin{aligned} J_{pp}^{11} &= -Q_p - B_{pp}|V_p|^2 \\ J_{pp}^{21} &= P_p - G_{pp}|V_p|^2 \\ J_{pp}^{12} &= \frac{P_p}{|V_p|} + G_{pp}|V_p| \\ J_{pp}^{22} &= \frac{Q_p}{|V_p|} - B_{pp}|V_p| \end{aligned} \quad (9)$$

These equations are extremely useful in reducing the computational complexity of the non-linear set of equations, as they avoid the direct symbolic computation of the derivatives [3]. Consequently, we have a closed-form expression for each component of the Jacobian. The inverse of this matrix can easily be found, and the resultant variables can be plugged in to 1 to find the unknowns directly.

3) Fast Newton-Raphson Approaches: Although the Newton-Raphson method is quick to converge and gives an excellent approximation, the computational complexity of implementing the algorithm may deter its utility in applications where approximations serve the purpose, and exact solutions are not necessary. This motivated the introduction of approximations into the Newton-Raphson algorithm, which are classified as follows based on the extent of approximation [3]:

1) **Decoupled Power Flow (DPF):** This method assumes the off diagonal elements of the Jacobian, J_{12} and J_{21} to be zero. The justification is that real power flow is from leading to lagging side with respect to voltage angle, while the magnitude has minimal impact. Similarly, Reactive Power flows from high voltage magnitudes to low voltage magnitudes, with minimal impact from the angle.
2) **Fast Decoupled Power Flow (FDPF):** The FDPF assumes that $G_{ij} = 0 \ \forall \ i, j$, thereby assuming that the Y_{bus} is purely imaginary. This is in addition to the assumptions of the DPF, and provides a fairly reasonable solution. Therefore, FPDF is used as a quick approximation to the solution when exact values are not necessary.
3) **DC Power Flow (DC):** The DC Assumptions are the most severe, and complete ignores the reactive power components of the system, in addition to the assumptions of the FPDF. This reduces the non-linear equations to a system of linear equations, which can be solved using elementary techniques.

This concludes the theoretical background required for the Newton-Raphson approach to solving the power flow equations. These concepts are utilized to present the implemented MATLAB code in the following section.

III. MATLAB Code for Newton-Raphson Algorithm

The code below consists of the main file and two functions that have been called inside the main file. The two functions are separately provided after the main file.

Listing 1: MATLAB Code: MainFile.m

```
%%  MAIN PROGRAM FOR POWER FLOW ANALYSIS
%%  The Program consists of the following Parts:
%Part 1: Importing Data and Creating Variables
    %1.1: Importing Data from .txt file
    %1.2: Creating Variables based on Matrix Data

%Part 2: Performing the Newton-Raphson Algorithm
    %2.1: Creating Vector of Unknowns,  x
    %2.2: Creating Vector of Equations, f
    %2.3: Creating the Jacobian Matrix, J
    %2.4: Applying the NR Formula

%Part 3: Saving the Data to a MS Excel File

%%  Part 1: Importing Data and Creating Variables
    %1.1: Importing Data from .txt file

txtdata = 'ieee30cdf.txt'; % Name of File
[Nbus, Nline, Bus_Data, Line_Data, S_base] = FetchData(txtdata); %
   Stores data as Matrix

    %1.2: Creating Variables based on Matrix Data
[Ybus, Pd, Qd, Pg, Qg, PV_buses, V_PV_buses, V_base, Slack_bus,
   V_Slack_bus, A_Slack_bus] = GetParameters(Bus_Data, Line_Data,
   S_base); %Creates Ybus, extracts Power Delivered and Generated at
   each bus, and the V_base for each bus

n = Nbus;                   %Renaming Nbus as n for easier referencing
s = Slack_bus;              %Renaming the Slack Bus
V = ones(n,1);              %Voltage Vector for Flat Start
delta = zeros(n,1);         %Angle Vector for Flat Start
x1 = [delta;V];             %Initializing Vector of Variables

idx1 = [s;PV_buses];        %Vector of Indices to be ignored from
   Jacobian
idx = setdiff(1:n, idx1);   %Vector of Indices Required in Jacobian

idx2 = [1:s-1 s+1:n];       %Vector of Non-Slack Buses
P = zeros(n,1);             %Initializing Vector for Real Power
Q = zeros(n,1);             %Initializing Vector for Reactive Power
f1 = [P;Q];                 %Initializing Vector of Functions
```

```
37
38 Jacobian = zeros(2*n,2*n); %Initializing the Jacobian
39
40 G = real(Ybus);            %Admittance  Matrix
41 B = imag(Ybus);            %Susceptance Matrix
42
43 %%  Part 2: Performing Newton Raphson Algorithm
44 epsilon = 10^-6;           %Allowed Tolerance for Newton-Raphson
45 mismatch = 10*ones(n,1);   %High Value to Enter Loop
46
47 while max(mismatch) > epsilon
48
49     %%2.1: Make the Vector of Unknowns, x
50     V(s) = V_Slack_bus;          %Voltage at Slack Bus
51     delta(s) = A_Slack_bus;      %Angle at Slack Bus
52     V(PV_buses) = V_PV_buses;    %PV Bus
53     x1 = [delta;V]               %Vector of Variables with Fixed
          Voltages for Slack and PV Buses
54
55     %%2.2: Make the Vector of Equations, f
56     for i = 1:n
57         P(i) = 0;                %Initializing Real Power Entry
58         Q(i) = 0;                %Initializing Reactive Power Entry
59         for k = 1:n
60             P(i) = P(i) + V(i)*V(k)*(G(i,k)*cos(delta(i) - delta(k)) +
                  B(i,k)*sin(delta(i)-delta(k))); %Load Flow Equation
61             Q(i) = Q(i) + V(i)*V(k)*(G(i,k)*sin(delta(i) - delta(k)) -
                  B(i,k)*cos(delta(i)-delta(k))); %Load Flow Equation
62         end
63     end
64     dP = P - Pg + Pd;            %Real     Power Mismatch Equation
65     dQ = Q - Qg + Qd;            %Reactive Power Mismatch Equation
66     f1 = [dP;dQ];                %Updatees Vector of Equations
67
68     %2.3: Create the Jacobian Matrix
69     for p = 1:n
70         for q = 1:n
71             if(p==q)
72                 %Using the Jacobian Entry Formulae
73                 J11(p,q) = -Q(p) - B(p,p)*V(p)^2;
74                 J21(p,q) = P(p) - G(p,p)*V(p)^2;
75                 J12(p,q) = P(p)./V(p) + G(p,p)*V(p);
76                 J22(p,q) = Q(p)./V(p) - B(p,p)*V(p);
77             else
78                 J12(p,q) = V(p)*(G(p,q)*cos(delta(p)-delta(q)) + B(p,q)
                      *sin(delta(p)-delta(q)));
79                 J22(p,q) = V(p)*(G(p,q)*sin(delta(p)-delta(q)) - B(p,q)
                      *cos(delta(p)-delta(q)));
```

```
                J11(p,q) = V(q)*J22(p,q);
                J21(p,q) = -V(q)*J12(p,q);
            end
        end
    end
    J = [J11,J12;J21,J22];                    %Forms Full Jacobian, J
    Jacobian = J([idx2 n+idx],[idx2 n+idx]); %Submatrix inside J w/o
       Slack and PV Bus Voltages

    %2.4: The Newton-Raphson Formula
    f = f1([idx2 n+idx]); f = f(:); %Subvector after deleting Slack and
        PV entries
    x = x1([idx2 n+idx]); x = x(:); %Subvector after deleting Slack and
        PV entries

    x_new = x - Jacobian\f;          %The Newton Raphson Algorithm
       Equation
    mismatch = x_new - x;            %Mismatch expressed as convergence
    x = x_new;                       %New Vector becomes Old Vector for
       Next Iteration

    delta(idx2) = x_new(1:n-1);      %The first n-1 entries are the
       Angles
    V(idx) = x_new(n:end);           %The rest of the entries are the
       Voltages

    V(s) = V_Slack_bus;              %Voltage at Slack Bus
    delta(s) = A_Slack_bus;          %Angle at Slack Bus
    V(PV_buses) = V_PV_buses;        %PV Bus
    x1 = [delta;V];                  %Final Result of Voltages and
       Angles
end
format short                         %Stores Data in short format
x1(1:n) = 180/pi *x1(1:n);           %Converts Angles to Degrees

%%  Part 3: Saving Data to Excel File
Results_Mat = [[1:n]' ,x1(n+1:2*n), x1(1:n)];     %Saves Results in a
   Matrix
if (strcmp(txtdata,'ieee30cdf.txt') == 1)
    xlswrite('Bus_Data_30.xlsx',Results_Mat);     %Saves Matrix to
       Excel File
elseif (strcmp(txtdata,'ieee118cdf.txt') == 1)
    xlswrite('Bus_Data_118.xlsx',Results_Mat);    %Saves Matrix to
       Excel File
end

%%  END OF MAIN PROGRAM
```

Listing 2: MATLAB Code: FetchData.m

```
function [Nbus, Nline, Bus_Data, Line_Data, S_base] = FetchData(
   filename)
%%  Function to read data from IEEE Common Data Format
% filename  : Name of the .txt file to read data from (Enter as string)
% Nbus      : Number of Buses
% Nline     : Number of Lines/Branches
% Bus_Data  : Matrix containing .txt data of Buses
% Line_Data : Matrix containing .txt data of Lines/Branches

%%  The Program Contains the Following Parts
    %Part 1: Opening the File to read data from
    %Part 2: Importing Bus Data
    %Part 3: Importing Line Data
    %Part 4: Closing the File

%% 1. Opening the File to read data from
fid = fopen(filename);

%% 2. Fetching Bus Data

%2.1: Loop to ignore first two lines of .txt document
    New_Line = fgetl(fid);              %Next Line of .txt document as
       a string
    S_base = str2num(New_Line(32:37));  %Extracts S Base from Line 1
    New_Line = fgetl(fid);              %Skips Line 2

    Bus_Data = []; % Initializing Bus_Data Matrix as empty matrix
    Nbus = 0;      % Initializing Number of Buses as zero

while ischar(New_Line)     %Loops While New Line is a Character Vector
   (Always True)
    New_Line = fgetl(fid); %Fetches New Line (in Bus Data)
    if(strcmp(New_Line(1:4),'-999') == 1)
        break              %Forces Loop to Stop if the New Line is
           '-999'
    end

    Col_Start = 19;                         %To Start from Column 19 as
       Data before that is not numeric or required in the Matrix
    Line_Str = New_Line(Col_Start:end);   %Stores New Line from Column
       19 to Last Column as a String
    Line_Vec = str2num(Line_Str);         %Converts the String to a
       Vector with numeric elements
    Nbus = Nbus + 1;                        %Increments number of buses
       as they are detected
    Bus_Data = [Bus_Data;[Nbus Line_Vec]] %Updates the Bus_Data Matrix
```

```
       every iteration
end

%% 3. Fetching Line Data
New_Line = fgetl(fid);      %Ignores Text Between Bus Data and Branch
   Data
Line_Data = [];             %Initializing Line_Data_Matrix as zero
   matrix
Nline = 0;                  %Initializing Number of Lines to zero

while ischar(New_Line)      %Loops While New Line is Character Vector (
   Always True)
    New_Line = fgetl(fid);  %Fetches New Line (in Line Data)
    if(strcmp(New_Line(1:4),'-999') == 1)
        break               %Forces Loop to Stop if the New Line is
           '-999'
    end
    Col_Start = 1;                        %To Start from Column 1 as all
       Columns are Numeric
    Line_Str = New_Line(Col_Start:end); %Stores New Line from Column 1
       to Last Column as a String
    Line_Vec = str2num(Line_Str);       %Converts The String to a
       Vector with numeric elements
    Nline = Nline+1;                    %Increments the Number of Lines
        as they are detected
    Line_Data = [Line_Data;[Nline Line_Vec]]; %Updates the Line_Data
       Matrix every iteration
end

%% 4. Closing the File
fclose(fid);
end

%%  END OF PROGRAM
```

Listing 3: MATLAB Code: GetParameters.m

```
function [Ybus, Pd, Qd, Pg, Qg, PV_buses, V_PV_buses, V_base, Slack_bus
   , V_Slack_bus, A_Slack_bus] = GetParameters(Bus_Data, Line_Data,
   S_base)
%%  Function to extract Individual Impedances, Powers and Parameters
   from Bus Data and Line Data
%   Ybus        : The Admittance Matrix
%   Pd          : Real Power Delivered at Bus as a vector
%   Qd          : Reactive Power Delivered at Bus as a vector
%   Pg          : Real Power Generated at Bus as a vector
%   Qg          : Reactive Power Generated at Bus as a vector
%   PV_buses    : The Index of PV Buses
```

```
%   V_PV_buses : The Voltage at PV Buses
%   Slack_bus  : The Index of Slack Buses
%   V_Slack_bus: The Voltage at Slack Bus
%   A_Slack_bus: The Angle at Slack Bus

Pd = Bus_Data(:,7)/S_base;              %Vector of Real Power Demanded
Qd = Bus_Data(:,8)/S_base;              %Vector of Reactive Power
   Demanded

Pg = Bus_Data(:,9)/S_base;              %Vector of Real Power Generated
Qg = Bus_Data(:,10)/S_base;             %Vector of Reactive Power
   Generated

V_base = Bus_Data(:,11);                %Vector of V_base

Gshunt = Bus_Data(:,15);                %Vector of Shunt Conductance p.u.
Bshunt = Bus_Data(:,16);                %Vector of Shunt Susceptance p.u.

Type_Bus    = Bus_Data(:,4);            %Vector of Type of Bus
PV_buses    = find(Type_Bus == 2);      %Finds the PV Buses
V_PV_buses  = Bus_Data(PV_buses,5);     %Returns Vector of Voltage
   Magnitudes at PV Bus
Slack_bus   = find(Type_Bus == 3);      %Finds the Slack Bus
V_Slack_bus = Bus_Data(Slack_bus,5);    %Returns Voltage at Slack Bus
A_Slack_bus = Bus_Data(Slack_bus,6);    %Returns Angle at Slack Bus in
   Degrees
A_Slack_bus = A_Slack_bus*pi/180;       %Converts Angle to Radians
Nbus = size(Bus_Data,1);                %Finds Number of Buses
Ybus = zeros(Nbus,Nbus);                %Initializes the Ybus

Index = Line_Data(:,[2 3]);             %2D Matrix with Column 1
   containing Bus i and Column 2 containing Bus j for Y_ij

%% 2. Buliding the Ybus

%2.1: Building First Upper Triangular Matrix
for k = 1:size(Line_Data,1)
    k1 = Index(k,1); k2 = Index(k,2);                       %Indices
    Ybus(k1,k2) = 1./(Line_Data(k,8) + 1j*Line_Data(k,9)); %Y = 1/(R+jX
       )
end
Ybus = -(Ybus+Ybus.');                                  %Adds Lower
   Triangular Elements
Ybus = Ybus - diag(sum(Ybus).' - Gshunt - 1j*Bshunt); %Adds Diagonal
   Elements
end
```

IV. Power World Implementation

This section deals with the implementation of the system in PowerWorld. The 30 Bus system is set up using components that are acquired from the available PowerWorld library, and the following figure is the result of the implementation.

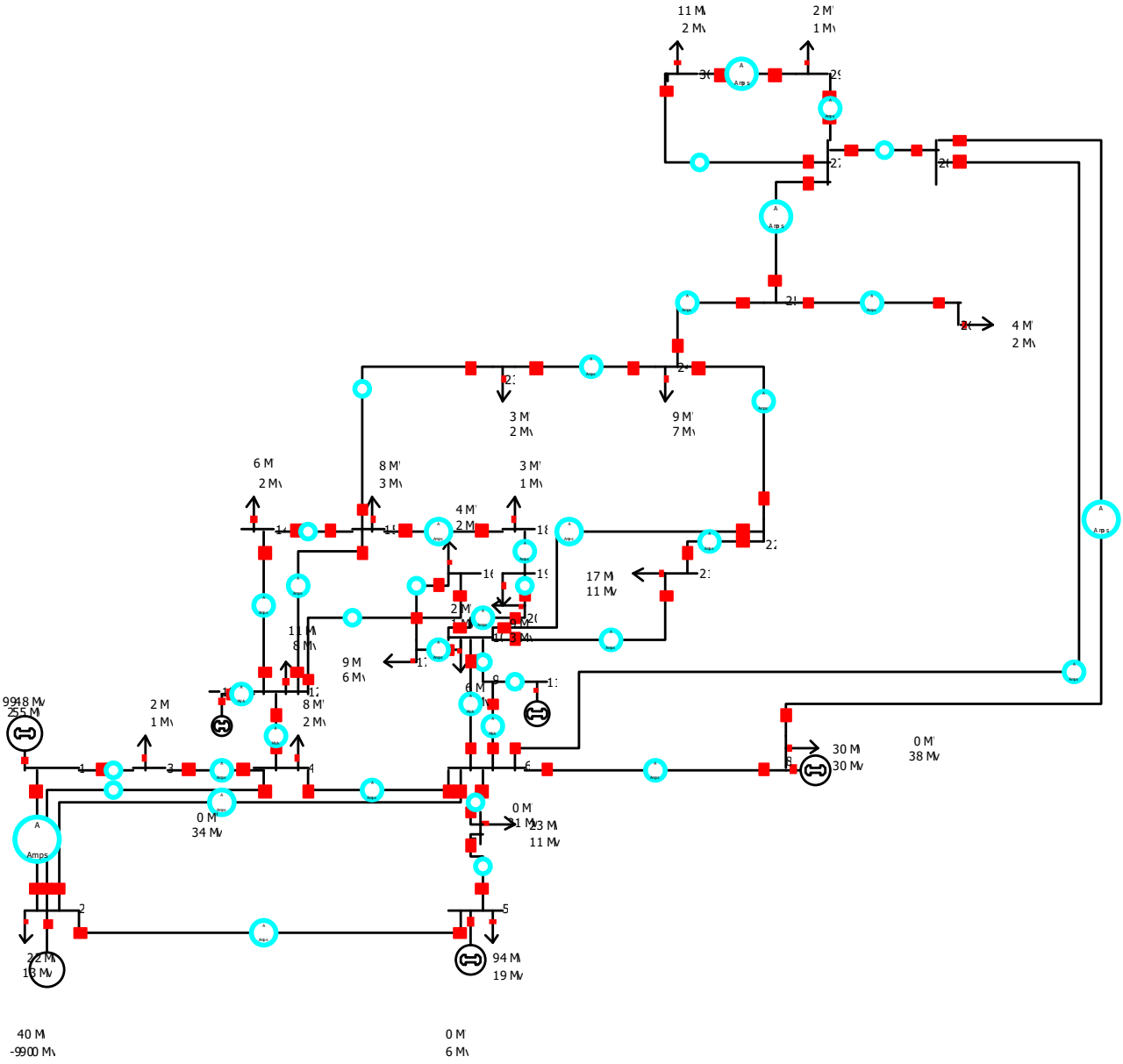

Fig. 1. 30 Bus System: Power World Implementation

The components utilized in the implementation of the above figure include transformers, transmission lines, buses, generators and loads. The parameters are fixed in the system as per the requirements of the project and the data is collected. The recorded observations are displayed and discussed in the forthcoming section.

V. Analytical and Simulation Results

The MATLAB Code is run with the 30 Bus and 118 Bus system respectively. As highlighted in the code, Bus Voltages and Bus Angles are stored in a MS Excel spreadsheet for easy access. This section deals with the presentation of the obtained results. The analytical results from MATLAB are first tabulated, and are followed by the results of Power World Simulation. This is succeeded by a section that discusses and compares the results from both sources.

A. MATLAB Results

1) 30 Bus System: The results of the 30 Bus System are tabulated in I.

TABLE I
MATLAB Results - 30 Bus System

Bus	Bus Voltage	Bus Angle
1	1.06	0
2	1.043	-5.345
3	1.024	-7.566
4	1.017	-9.34
5	1.01	-14.16
6	1.012	-11.08
7	1.002	-12.85
8	1.01	-11.81
9	1.035	-14.36
10	1.024	-16.07
11	1.082	-14.36
12	1.033	-15.36
13	1.071	-15.36
14	1.018	-16.28
15	1.014	-16.37
16	1.022	-15.94
17	1.018	-16.25
18	1.005	-16.99
19	1.003	-17.16
20	1.007	-16.94
21	1.011	-16.53
22	1.011	-16.51
23	1.003	-16.74
24	0.997	-16.87
25	0.989	-16.35
26	0.971	-16.79
27	0.993	-15.75
28	1.007	-11.68
29	0.973	-17.06
30	0.961	-18

2) 118 Bus System: The same algorithm is used for the 118 bus system, simply by changing the name of the file to read the data from. The results are tabulated in II:

TABLE II
MATLAB Results - 118 Bus System

Bus	Voltage	Angle	Bus	Voltage	Angle
1	0.955	9.344	60	0.993	23.71
2	0.971	9.89	61	0.995	24.59
3	0.966	10.26	62	0.998	24.19
4	0.998	13.99	63	0.988	23.12
5	0.999	14.48	64	0.994	25
6	0.99	11.68	65	1.005	28.27
7	0.989	11.23	66	1.05	29.23
8	1.015	19.64	67	1.018	26.16
9	1.023	27.12	68	1.01	27.98
10	1.05	34.77	69	1.035	30
11	0.984	11.4	70	0.984	22.23
12	0.99	10.85	71	0.986	21.76
13	0.965	10.01	72	0.98	20.33
14	0.982	10.11	73	0.991	21.54
15	0.97	9.646	74	0.958	21.28
16	0.979	10.59	75	0.966	22.56
17	0.985	12.42	76	0.943	21.19
18	0.973	9.993	77	1.006	25.81
19	0.963	9.457	78	1.003	25.7
20	0.951	10.53	79	1.009	26.35
21	0.95	12.24	80	1.04	29.58
22	0.96	14.94	81	1.021	28.53
23	0.995	20	82	0.98	27.83
24	0.992	19.99	83	0.975	30.16
25	1.05	26.82	84	0.975	34.48
26	1.015	28.66	85	0.985	36.84
27	0.968	14.13	86	0.984	35.52
28	0.96	12.38	87	1.015	35.75
29	0.963	11.34	88	0.985	41.55
30	0.988	17.7	89	1.005	46.67
31	0.967	11.43	90	0.985	34.46
32	0.964	13.55	91	0.98	32.65
33	0.965	8.82	92	0.993	30.41
34	0.986	9.062	93	0.985	28.52
35	0.979	8.651	94	0.987	27.31
36	0.98	8.637	95	0.976	26.86
37	0.983	9.634	96	0.986	27.33
38	0.977	15.79	97	1.007	28.12
39	0.966	4.717	98	1.021	27.32
40	0.97	2.749	99	1.01	26.25
41	0.966	1.698	100	1.017	26.68
42	0.985	1.59	101	0.991	27.42
43	0.969	9.408	102	0.99	29.32
44	0.974	12.34	103	1.001	23.08
45	0.979	14.3	104	0.971	20.34
46	1.005	17.14	105	0.965	19.23
47	1.015	19.57	106	0.959	19.01
48	1.019	18.56	107	0.952	16.17
49	1.025	19.53	108	0.965	18.05
50	0.999	17.3	109	0.966	17.6
51	0.961	14.4	110	0.973	16.73
52	0.95	13.37	111	0.98	18.38
53	0.941	12.11	112	0.975	13.63
54	0.955	12.77	113	0.993	12.28
55	0.952	12.67	114	0.96	13.24
56	0.954	12.72	115	0.96	13.23
57	0.968	14.31	116	1.005	27.58
58	0.955	13.41	117	0.971	9.338
59	0.985	19.41	118	0.948	21.47

B. Power World Results

TABLE III
POWER WORLD RESULTS - 30 BUS SYSTEM

Number	Name	Area	Nom kV	PU Volt	Volt (kV)	Angle (Deg)	Load MW	Load Mvar	Gen MW	Gen Mvar
1	1	1	132	1.06	139.92	0			254.98	9948.24
2	2	1	132	1.059	139.8	0	21.7	12.7	40	-9900
3	3	1	132	1.027	135.61	-4.76	2.4	1.2		
4	4	1	132	1.019	134.52	-5.83	7.6	1.6		
5	5	1	132	1.01	133.32	-7.29	94.2	19	0	5.71
6	6	1	132	1.01	133.32	-7.29				
7	7	1	132	1.002	132.23	-7.82	22.8	10.9		
8	8	1	132	1.01	133.32	-8.04	30	30	0	38.22
9	9	1	1	1.023	1.023	-10.55				
10	10	1	33	1.001	33.025	-12.29	5.8	2		
11	11	1	11	1.082	11.902	-10.55			0	30.5
12	12	1	33	1.026	33.874	-11.94	11.2	7.5		
13	13	1	11	1.071	11.781	-11.94			0	34.05
14	14	1	33	1.009	33.295	-12.84	6.2	1.6		
15	15	1	33	1.002	33.078	-12.85	8.2	2.5		
16	16	1	33	1.008	33.259	-12.34	3.5	1.8		
17	17	1	33	0.998	32.923	-12.55	9	5.8		
18	18	1	33	0.989	32.642	-13.4	3.2	0.9		
19	19	1	33	0.985	32.492	-13.52	9.5	3.4		
20	20	1	33	0.988	32.598	-13.28	2.2	0.7		
21	21	1	33	0.987	32.584	-12.76	17.5	11.2		
22	22	1	33	0.988	32.598	-12.74				
23	23	1	33	0.987	32.555	-13.1	3.2	1.6		
24	24	1	33	0.974	32.146	-13.07	8.7	6.7		
25	25	1	33	0.974	32.148	-12.67				
26	26	1	33	0.956	31.537	-13.13	3.5	2.3		
27	27	1	33	0.983	32.445	-12.14				
28	28	1	132	1.006	132.78	-7.91				
29	29	1	33	0.962	31.76	-13.47	2.4	0.9		
30	30	1	33	0.95	31.364	-14.43	10.6	1.9		

VI. DISCUSSION

This section deals with the comparison of the analytical and simulation results to verify their agreement. Table IV lists all the relevant data and compares each obtained value of the analytical and simulation result with the theoretical expectation. The first part of IV is the Voltage Data. The values obtained from Power World, MATLAB as well as the IEEE txt file data are explicitly listed as three separate columns. This is followed by the calculation of percentage error of the values for both MATLAB and PowerWorld. A similar comparison is then done with the angle data in degrees, where the data itself is explicitly listed, followed by the percentage error in the MATLAB and PowerWorld Data with respect to the reference IEEE txt data. It must be noted that 'PW' and 'MAT' have been used in the table as shorthand for 'Power World' and 'MATLAB' respectively.

The magntiude of absolute percentage errors are reflective of the agreement between the theoretically expected data and the practically obtained data. Firstly, all errors are within 5% of the expected value, which is within the scope of experimental inaccuracy. The Newton-Raphson algorithm is extremely precise, with the MATLAB error percentages belonging to the $0-1\%$ range with the exception of certain outliers. Further, the Power World simulation data is in agreement with both the MATLAB and the txt file data. This verifies the accuracy of the obtained results.

TABLE IV
COMPARISON OF RESULTS

	Voltage					Angle				
Bus	txt Data	PW Data	MAT Data	% Err PW	% Err MAT	txt Data	PW Data	MAT Data	% Err PW	% Err MAT
1	1.06	1.06	1.06	0	0	0	0	0	0	0
2	1.043	1.059	1.043	1.54	0	-5.48	0	-5.345	100	2.458
3	1.021	1.027	1.024	0.619	0.294	-7.96	-4.76	-7.566	40.2	4.952
4	1.012	1.019	1.017	0.703	0.464	-9.62	-5.83	-9.34	39.4	2.916
5	1.01	1.01	1.01	0	0	-14.37	-7.29	-14.16	49.27	1.475
6	1.01	1.01	1.012	0	0.168	-11.34	-7.29	-11.08	35.71	2.278
7	1.002	1.002	1.002	0.025	0.03	-13.12	-7.82	-12.85	40.4	2.03
8	1.01	1.01	1.01	0	0	-12.1	-8.04	-11.81	33.55	2.435
9	1.051	1.023	1.035	2.63	1.503	-14.38	-10.55	-14.36	26.63	0.168
10	1.045	1.001	1.024	4.234	1.99	-15.97	-12.29	-16.07	23.04	0.601
11	1.082	1.082	1.082	0	0	-14.39	-10.55	-14.36	26.69	0.238
12	1.057	1.026	1.033	2.886	2.28	-15.24	-11.94	-15.36	21.65	0.811
13	1.071	1.071	1.071	0	0	-15.24	-11.94	-15.36	21.65	0.811
14	1.042	1.009	1.018	3.173	2.303	-16.13	-12.84	-16.28	20.4	0.944
15	1.038	1.002	1.014	3.434	2.351	-16.22	-12.85	-16.37	20.78	0.906
16	1.045	1.008	1.022	3.555	2.249	-15.83	-12.34	-15.94	22.05	0.713
17	1.04	0.998	1.018	4.07	2.096	-16.14	-12.55	-16.25	22.24	0.684
18	1.028	0.989	1.005	3.778	2.247	-16.82	-13.4	-16.99	20.33	0.999
19	1.026	0.985	1.003	4.035	2.242	-17	-13.52	-17.16	20.47	0.919
20	1.03	0.988	1.008	4.094	2.184	-16.8	-13.28	-16.94	20.95	0.856
21	1.033	0.987	1.011	4.415	2.13	-16.42	-12.76	-16.53	22.29	0.649
22	1.033	0.988	1.011	4.373	2.101	-16.41	-12.74	-16.51	22.36	0.616
23	1.027	0.987	1.003	3.941	2.356	-16.61	-13.1	-16.74	21.13	0.773
24	1.021	0.974	0.997	4.592	2.341	-16.78	-13.07	-16.87	22.11	0.546
25	1.017	0.974	0.989	4.211	2.724	-16.35	-12.67	-16.35	22.51	0.015
26	1	0.956	0.971	4.432	2.89	-16.77	-13.13	-16.79	21.71	0.129
27	1.023	0.983	0.993	3.893	2.913	-15.82	-12.14	-15.75	23.26	0.437
28	1.007	1.006	1.007	0.107	0	-11.97	-7.91	-11.68	33.92	2.412
29	1.003	0.962	0.973	4.045	3.021	-17.06	-13.47	-17.06	21.04	0.016
30	0.992	0.95	0.961	4.19	3.135	-17.94	-14.43	-18	19.57	0.313

VII. CONCLUSION

The project was assigned with the objective of developing a practical feel towards some of the most important tools of power flow analysis - simulations and algorithms. The Newton-Raphson approach was implemented to iteratively and numerically obtain the solution to the non-linear system of equations that govern the flow of power in a system of buses. The algorithm was tested on two files of 30 and 118 buses respectively. The results of the 30 bus system were compared with simulation results obtained from having implemented the network on Power World - a software used for analysis of large bus systems. The results are compared and the observations are documented. Inferences and insights gained from the experiment are also provided in the report. The approach of the mismatch vector towards zero and the quick convergence of the Newton-Raphson algorithm reiterates its usefulness in load flow analysis. Further, the agreement between the results of the Power Flow simulation and the analytical algorithm implementation verifies the accuracy of the solution. The errors are within the scope of computational inaccuracy, and therefore the objectives of the course project are achieved.

REFERENCES

[1] J. D. D. Glover and M. S. Sarma, *Power System Analysis and Design*. Pacific Grove, CA, USA: Brooks/Cole Publishing Co., 3rd ed., 2001.

[2] M. Shaaban, *Lecture Slides 1: Fundamentals of Power Systems*. 2017.

[3] M. Shaaban, *Lecture Slides 6: Power Flows*. 2017.