|  |  |  |  |
| --- | --- | --- | --- |
| FACULTY OF ENGINEERING |  | CAIRO UNIVERSITY | Application  Description automatically generated with medium confidence |
| 4th Year | | | |
| Course: Electrical Power Systems (3)  (EPE4010) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| B.N. | Section | ID | Name |
| 5 | 3 | 9202977 | Omat Mohamed Gamal Taher Nouh |
| 6 | 3 | 9202980 | Omar Mohamed Saeed Hussein |
| 29 | 3 | 9203173 | Mohamed Ashraf Mabrouk EL-Abd |
| 31 | 3 | 9203193 | Mohamed Gamal Hob El-Din Torky |
| 43 | 3 | 9203271 | Mohamed Tarek Abd El-Salam Amin |

Project topic: Power System Solution of The Power Flow Problem Using Newton Raphson Technique

Instructor: Dr. Mohamed Yousry

Date: 22/12/2023

# TABLE OF CONTENTS

[TABLE OF CONTENTS 2](#_Toc154151680)

[TABLE OF FIGURES 2](#_Toc154151681)

[TABLE OF EQUATIONS 2](#_Toc154151682)

[INTRODUCTION 3](#_Toc154151683)

[MATHEMATICAL FORMULATION 3](#_Toc154151684)

[NEWTON-RAPHSON METHOD 3](#_Toc154151685)

[NR ALGORITHM FOR LOAD FLOW SOLUTION 3](#_Toc154151686)

[Case 1 7](#_Toc154151687)

[Case 2 7](#_Toc154151688)

[Iterative Algorithm 9](#_Toc154151689)

[ADVANTAGES AND DISADVANTAGES 9](#_Toc154151690)

[CONCLUSION 9](#_Toc154151691)

[APPENDIX 10](#_Toc154151692)

[Example 1 using nr method 11](#_Toc154151693)

[Example 2 using gs method 12](#_Toc154151694)

[Program code 13](#_Toc154151695)

[REFERENCES 19](#_Toc154151696)

# TABLE OF FIGURES

[Figure 1 8](#_Toc154151697)

[Figure 2 11](#_Toc154151698)

[Figure 3 12](#_Toc154151699)

[Figure 4 19](#_Toc154151700)

# TABLE OF EQUATIONS

[Equation 1 4](#_Toc154151701)

[Equation 2 4](#_Toc154151702)

[Equation 3 4](#_Toc154151703)

[Equation 4 4](#_Toc154151704)

[Equation 5 4](#_Toc154151705)

[Equation 6 4](#_Toc154151706)

[Equation 7 5](#_Toc154151707)

[Equation 8 5](#_Toc154151708)

[Equation 9 6](#_Toc154151709)

[Equation 10 6](#_Toc154151710)

[Equation 11 6](#_Toc154151711)

[Equation 12 7](#_Toc154151712)

[Equation 13 7](#_Toc154151713)

[Equation 14 7](#_Toc154151714)

[Equation 15 7](#_Toc154151715)

[Equation 16 7](#_Toc154151716)

[Equation 17 8](#_Toc154151717)

[Equation 18 8](#_Toc154151718)

[Equation 19 9](#_Toc154151719)

Newton-Raphson Method for Load Flow Analysis

# INTRODUCTION

The load flow problem is a fundamental problem in power system analysis that involves determining the voltage magnitudes and angles at all buses in a power system under specified load conditions. Solving the load flow problem is essential for various power system studies, including power flow analysis, contingency analysis, and optimal power flow.

# MATHEMATICAL FORMULATION

The load flow problem can be mathematically formulated as a system of nonlinear equations, known as the power flow equations. These equations represent the balance of active and reactive power at each bus in the power system. The power flow equations are given by:

# NEWTON-RAPHSON METHOD

The Newton-Raphson method is an iterative numerical method commonly used to solve the load flow problem. The method starts with an initial guess for the voltage magnitudes and angles at all buses. These initial values are then used to calculate the power mismatch at each bus. The power mismatch is defined as the difference between the injected power and the power consumed at each bus.

The Newton-Raphson method then calculates the Jacobian matrix, which is a matrix of partial derivatives of the power mismatch equations with respect to the voltage magnitudes and angles. The Jacobian matrix is used to determine the correction to the voltage magnitudes and angles that will reduce the power mismatch.

The corrected voltage magnitudes and angles are then used to update the initial guess. This process is repeated until the power mismatch at each bus is within a specified tolerance.

# **NR ALGORITHM FOR LOAD FLOW SOLUTION**

First, assume that all buses are PQ buses. At any PQ bus the load flow solution must satisfy the following non-linear algebraic equations

[Newton Raphson Method for Load Flow Analysis](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-011.jpg)

Equation

Equation

where expressions for Pi and Qi are given in Eqs. (6.27) and (6.28). For a trial  set of variables |Vi|, δi, the vector of residuals f0 of Eq. (6.57) corresponds to

[Newton Raphson Method for Load Flow Analysis](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-012.jpg)

Equation

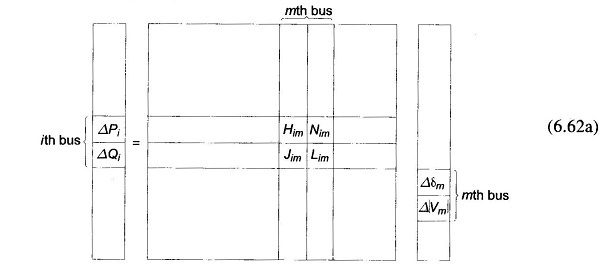
Equation

while the vector of corrections , Δx0 corresponds to

[](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-013.jpg)

Equation

Equation (6.57) for obtaining the approximate corrections vector can be written for the load flow case as

[](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-014.jpg)

Equation

where

[A close-up of a person

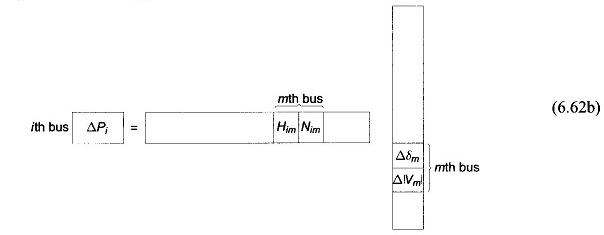
Description automatically generated](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-015.jpg)

Equation

It is to be immediately observed that the Jacobian elements corresponding to the ith bus residuals and mth bus corrections are a 2 x 2 matrix enclosed in the box in Eq. (6.62a) where i and m are both PQ buses.

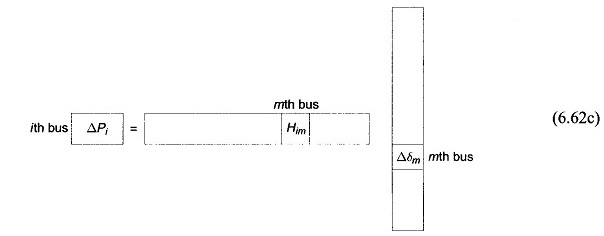
Since at the slack bus (bus number 1), P1 and Q1 are unspecified and |V1|, δ1 are fixed, there are no equations corresponding to Eq. (6.60) at this bus. Hence the slack bus does not enter the Jacobian in Eq. (6.62a).

Consider Newton Raphson Method for Load Flow Analysis Formula now the presence of PV buses. If the ith bus is a PV bus, Qi is unspecified so that there is no equation corresponding to Eq. (6.60b) for this bus. Therefore, the Jacobian elements of the ith bus become a single row pertaining to ΔPi , i.e.

[](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-016.jpg)

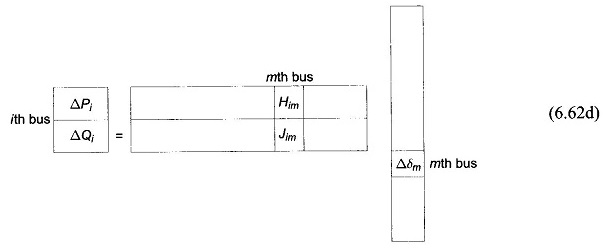
Equation

If the mth bus is also a PV bus, |Vm| becomes fixed so that Δ|Vm| = 0. We can now write

[](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-017.jpg)

Equation

Also if the ith bus is a PQ bus while the mth bus is a PV bus, we can then write

[](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-018.jpg)

Equation

It is convenient for numerical solution to normalize the voltage corrections

[](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-019.jpg)

Equation

as a consequence of which, the corresponding Jacobian elements become

[A white background with black dots

Description automatically generated](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-020.jpg)

Equation

Expressions for elements of the Jacobian (in normalized form) of the load flow Eqs. (6.60a and b) are given below:

## ****Case 1****

[A close-up of a logo

Description automatically generated](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-021.jpg)

Equation

where

[A white background with black text

Description automatically generated](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-022.jpg)

Equation

## ****Case 2****

[A white background with black and white clouds

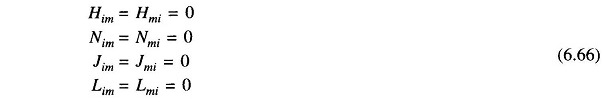
Description automatically generated](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-023.jpg)

Equation

[](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-024.jpg)

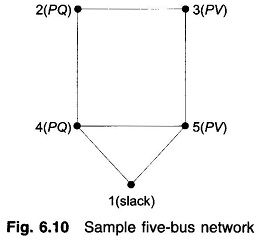
Equation

An important observation can be made in respect of the Jacobian by examination of the YBUS matrix. If buses i and m are not connected, Yim = 0 (Gim = Bim = 0). Hence from Eqs. (6.63) and (6.64), we can write

[](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-025.jpg)

Equation

Thus the Jacobian is as sparse as the YBUS matrix.

[](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-028.jpg)

Figure

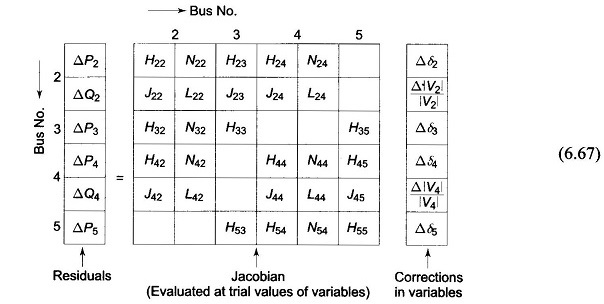
Formation of Eq. (6.62) of the NR method is best illustrated by a problem. Figure 6.10 shows a five-bus power network with bus types indicated therein. The matrix equation for determining the vector of corrections from the vector of residuals is given below.

Corresponding to a particular vector of variables [δ2|V2|δ3δ4|V4|δ5]T, the vector of residuals [ΔP2 ΔQ2 ΔP3 ΔP4 ΔQ4 ΔP5]Tand the Jacobian (6 x 6 in this example) are computed. Equation (6.67) is then solved by triangularization and back substitution procedure to obtain the vector of corrections

[Newton Raphson Method for Load Flow Analysis](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-026.jpg)

Equation

Corrections are then added to update the vector of variables.

[](https://www.eeeguide.com/wp-content/uploads/2016/11/Newton-Raphson-Method-for-Load-Flow-Analysis-027.jpg)

Equation

## ****Iterative Algorithm****

Omitting programming details, the iterative algorithm for the solution of the load flow problem by the NR method is as follows:

1. With voltage and angle (usually δ**=**0) at slack bus fixed, assume |V|, δ at all PQ buses and δ at all PV In the absence of any other information flat voltage start is recommended.
2. Compute ΔPi (for PV and PQ buses) and ΔQi, (for all PQ buses) from (6.60a and b). If all the values are less than the prescribed tolerance, stop the iterations, calculate P1 and Q1 and print the entire solution including line flows.
3. If the convergence criterion is not satisfied, evaluate elements of the Jacobian using Eqs. (6.64) and (6.65).
4. Solve Eq. (6.67) for corrections of voltage angles and magnitudes.
5. Update voltage angles and magnitudes by adding the corresponding changes to the previous values and return to step 2.

# ADVANTAGES AND DISADVANTAGES

The Newton-Raphson method is a powerful and efficient method for solving the load flow problem. It is relatively easy to implement and converges rapidly to a solution. However, the method can be sensitive to the initial guess and may not converge if the initial guess is too far from the actual solution.

# CONCLUSION

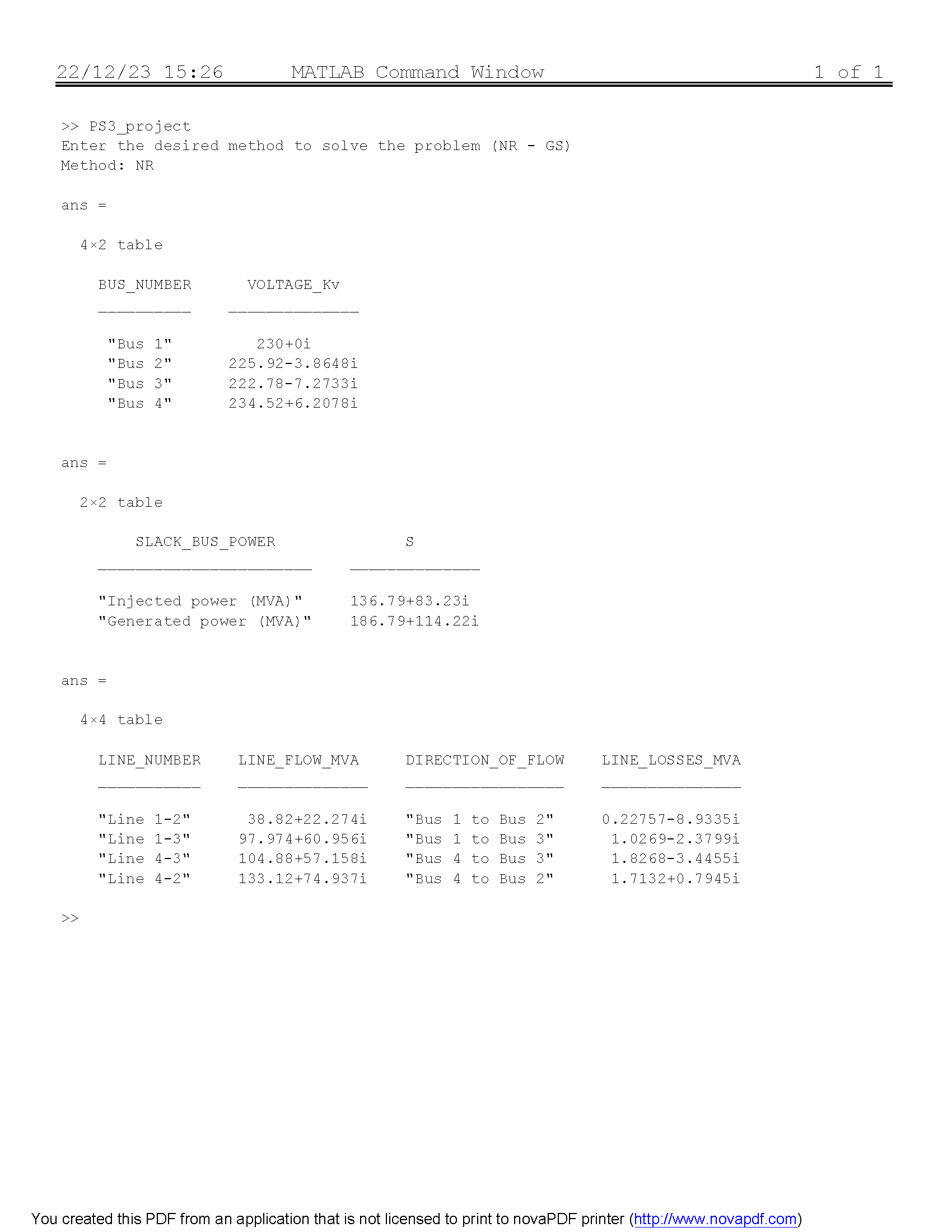
The Newton-Raphson method is a widely used method for solving the load flow problem in power system analysis. It is a powerful and efficient method that converges rapidly to a solution. However, the method can be sensitive to the initial guess and may not converge if the initial guess is too far from the actual solution.

# APPENDIX

This section shows some examples using the program and the outputs displayed therein.

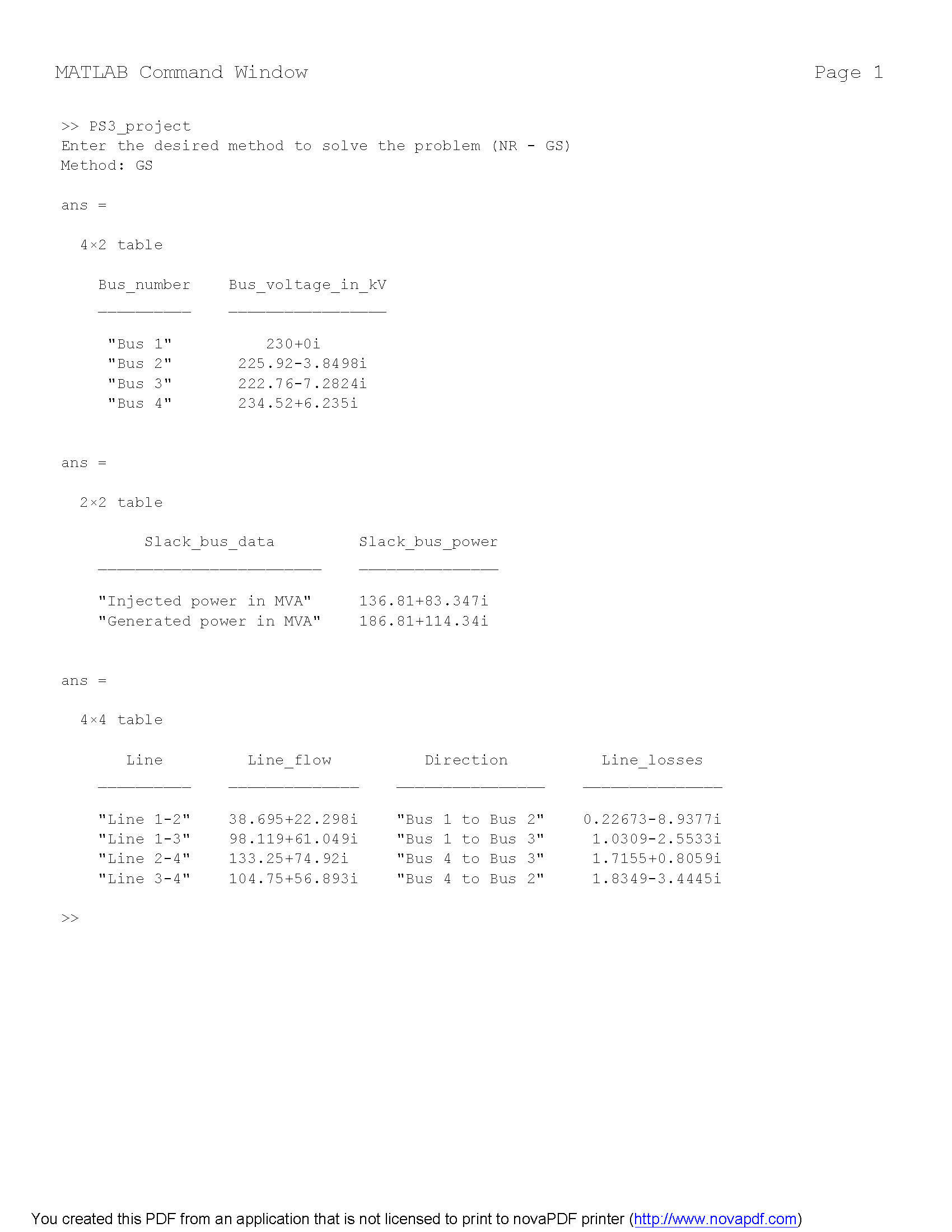
**PLEASE NOTE THAT WE HAVE CREATED AN APPLICATION THAT CALCULATES THE LOAD FLOW PARAMETERS USING BOTH NR AND GS METHODS.**

## Example 1 using nr method



Figure

## Example 2 using gs method



Figure

## Program code

A = 'Enter the desired method to solve the problem (NR - GS)';

disp(A)

method = input('Method: ', 's'); % 's' flag ensures that the input is treated as a string

if strcmp(method, 'NR')

%% Ybus FORMULATION

Y(1,1) = 3.81563 -19.07814i +5.1696 -25.8478i +0.05125i +0.03875i;

Y(1,2) = -3.81563 +19.07814i;

Y(1,3)= -5.1696 +25.8478i;

Y(1,4) = 0;

Y(2,1) = -3.81563 +19.07814i;

Y(2,2) = 3.81563 -19.07814i + 5.1696 -25.8478i + 0.03875i +0.05125i;

Y(2,3) = 0;

Y(2,4) = -5.1696 +25.8478i;

Y(3,1) = -5.1696 +25.8478i;

Y(3,2) = 0;

Y(3,3)= 5.1696 -25.8478i + 3.023705 -15.18528i + 0.06375i +0.03875i;

Y(3,4) = -3.023705 +15.18528i;

Y(4,1) = 0;

Y(4,2) = -5.1696 +25.8478i;

Y(4,3) = -3.023705 +15.18528i;

Y(4,4) = 5.1696 -25.8478i +3.023705 -15.18528i + 0.06375i +0.03875i;

%% GIVENS

%slack bus:

V(1) =1+0i;

Slack\_Bus\_Load = 0.5 +0.3099i;

%PQ Buses:

Psch(2) = -1.7;

Qsch(2) = -1.0535;

V(2) = 1+0i; %Initial condition

Psch(3) = -2;

Qsch(3) = -1.2394;

V(3) = 1+0i; %Initial condition

%PV bus:

Psch(4) = 2.38;

V(4) = 1.02 + 0i;

%% ITERATIONS

for j = 0:1:5

%Jacobian matrix calculation:

jacobian(1,1) = abs(V(2)\*Y(2,1)\*V(1))\*sin(angle(Y(2,1))+angle(V(1))-angle(V(2)))+abs(Y(2,4)\*V(4)\*V(2))\*sin(angle(Y(2,4))+angle(V(4))-angle(V(2)));

jacobian(1,2) = 0;

jacobian(1,3) = -abs(V(2)\*Y(2,4)\*V(4))\*sin(angle(Y(2,4))+angle(V(4))-angle(V(2)));

jacobian(1,4) = 2\*(abs(V(2))^2)\*real(Y(2,2)) + abs(V(2)\*Y(2,1)\*V(1))\*cos(angle(Y(2,1))+angle(V(1))-angle(V(2)))+abs(Y(2,4)\*V(4)\*V(2))\*cos(angle(Y(2,4))+angle(V(4))-angle(V(2)));

jacobian(1,5) =0;

jacobian(2,1) =0;

jacobian(2,2) = abs(V(3)\*Y(3,1)\*V(1))\*sin(angle(Y(3,1))+angle(V(1))-angle(V(3)))+abs(Y(3,4)\*V(4)\*V(3))\*sin(angle(Y(3,4))+angle(V(4))-angle(V(3)));

jacobian(2,3) = -abs(V(3)\*Y(3,4)\*V(4))\*sin(angle(Y(3,4))+angle(V(4))-angle(V(3)));

jacobian(2,4) = 0;

jacobian(2,5) = 2\*(abs(V(3))^2)\*real(Y(3,3))+ abs(V(3)\*Y(3,1)\*V(1))\*cos(angle(Y(3,1))+angle(V(1))-angle(V(3)))+abs(Y(3,4)\*V(4)\*V(3))\*cos(angle(Y(3,4))+angle(V(4))-angle(V(3)));

jacobian(3,1) = -abs(V(2)\*Y(4,2)\*V(4))\*sin(angle(Y(4,2))+angle(V(2))-angle(V(4)));

jacobian(3,2) = -abs(V(3)\*Y(4,3)\*V(4))\*sin(angle(Y(4,3))+angle(V(3))-angle(V(4)));

jacobian(3,3) = abs(V(2)\*Y(4,2)\*V(4))\*sin(angle(Y(4,2))+angle(V(2))-angle(V(4)))+abs(Y(4,3)\*V(3)\*V(4))\*sin(angle(Y(4,3))+angle(V(3))-angle(V(4)));

jacobian(3,4) = abs(V(4)\*Y(4,2)\*V(2))\*cos(angle(Y(4,2))+angle(V(2))-angle(V(4)));

jacobian(3,5) = abs(V(4)\*Y(4,3)\*V(3))\*cos(angle(Y(4,3))+angle(V(3))-angle(V(4)));

jacobian(4,1) = abs(V(2)\*Y(2,1)\*V(1))\*cos(angle(Y(2,1))+angle(V(1))-angle(V(2)))+abs(Y(2,4)\*V(4)\*V(2))\*cos(angle(Y(2,4))+angle(V(4))-angle(V(2)));

jacobian(4,2) = 0;

jacobian(4,3) = -abs(V(2)\*Y(2,4)\*V(4))\*cos(angle(Y(2,4))+angle(V(4))-angle(V(2)));

jacobian(4,4) = -2\*(abs(V(2))^2)\*imag(Y(2,2)) -abs(V(2)\*Y(2,1)\*V(1))\*sin(angle(Y(2,1))+angle(V(1))-angle(V(2)))-abs(Y(2,4)\*V(4)\*V(2))\*sin(angle(Y(2,4))+angle(V(4))-angle(V(2)));

jacobian(4,5) = 0;

jacobian(5,1) = 0;

jacobian(5,2) = abs(V(3)\*Y(3,1)\*V(1))\*cos(angle(Y(3,1))+angle(V(1))-angle(V(3)))+abs(Y(3,4)\*V(4)\*V(3))\*cos(angle(Y(3,4))+angle(V(4))-angle(V(3)));

jacobian(5,3) = -abs(V(3)\*Y(3,4)\*V(4))\*cos(angle(Y(3,4))+angle(V(4))-angle(V(3)));

jacobian(5,4) = 0;

jacobian(5,5) = -2\*(abs(V(3))^2)\*imag(Y(3,3)) -abs(V(3)\*Y(3,1)\*V(1))\*sin(angle(Y(3,1))+angle(V(1))-angle(V(3)))-abs(Y(3,4)\*V(4)\*V(3))\*sin(angle(Y(3,4))+angle(V(4))-angle(V(3)));

%Inverse jacobian matrix:

Inv\_Jacobian = inv(jacobian);

%Calculated power(P):

for cntr=2:1:4

Psum = 0;

for n=1:1:4

Psum = Psum + abs(Y(cntr,n)\*V(n))\*cos(angle(Y(cntr,n))+angle(V(n))-angle(V(cntr)));

end

Pcalc(cntr) = abs(V(cntr))\*Psum;

end

%Calculated reactive power(Q):

for cntr=2:1:3

Qsum = 0;

for n=1:1:4

Qsum = Qsum + abs(Y(cntr,n)\*V(n))\*sin(angle(Y(cntr,n))+angle(V(n))-angle(V(cntr)));

end

Qcalc(cntr) = -abs(V(cntr))\*Qsum;

end

%Delta matrix calculation:

delta\_matrix(1,1) = Psch(2)-Pcalc(2);

delta\_matrix(2,1) = Psch(3)-Pcalc(3);

delta\_matrix(3,1) = Psch(4)-Pcalc(4);

delta\_matrix(4,1) = Qsch(2)-Qcalc(2);

delta\_matrix(5,1) = Qsch(3)-Qcalc(3);

%Delta variables matrix:

delta\_var = Inv\_Jacobian\*delta\_matrix;

%New values:

V2angle = angle(V(2))+ delta\_var(1,1);

V3angle = angle(V(3))+ delta\_var(2,1);

V4angle = angle(V(4))+ delta\_var(3,1);

V2mag = abs(V(2))\*(1+ delta\_var(4,1));

V3mag = abs(V(3))\*(1+ delta\_var(5,1));

V(2) = V2mag\*(cos(V2angle)+i\*sin(V2angle));

V(3) = V3mag\*(cos(V3angle)+i\*sin(V3angle));

V(4) = 1.02\*(cos(V4angle)+i\*sin(V4angle));

end

%% SLACK BUS POWER CALCULATION

CurrentSum =0;

for n=1:1:4

CurrentSum = CurrentSum + V(n)\*Y(1,n);

end

Slack\_Bus\_Injected\_Power = V(1)\*conj(CurrentSum);

Slack\_Bus\_Generated\_Power = Slack\_Bus\_Injected\_Power + Slack\_Bus\_Load ;

%% POWER FLOWS AND POWER LOSSES CALCULATIONS

%Line 1-2:

S12 = V(1)\*conj(-(V(1)-V(2))\*Y(1,2)+V(1)\*0.05125i);

S21 = V(2)\*conj(-(V(2)-V(1))\*Y(1,2)+V(2)\*0.05125i);

Line1\_2Losses = S12+S21;

%Line 1-3:

S13 = V(1)\*conj(-(V(1)-V(3))\*Y(1,3)+V(1)\*0.03875i);

S31 = V(3)\*conj(-(V(3)-V(1))\*Y(1,3)+V(3)\*0.03875i);

Line1\_3Losses = S13+S31;

%Line 3-4:

S34 = V(3)\*conj(-(V(3)-V(4))\*Y(3,4)+V(3)\*0.06375i);

S43 = V(4)\*conj(-(V(4)-V(3))\*Y(3,4)+V(4)\*0.06375i);

Line4\_3Losses = S34+S43;

%Line 2-4:

S24 = V(2)\*conj(-(V(2)-V(4))\*Y(2,4)+V(2)\*0.03875i);

S42 = V(4)\*conj(-(V(4)-V(2))\*Y(2,4)+V(4)\*0.03875i);

Line4\_2Losses = S24 +S42;

%% RESULTS DISPLAY

%Bus voltages display:

BUS\_NUMBER = ["Bus 1";"Bus 2"; "Bus 3"; "Bus 4"];

VOLTAGE\_Kv =[V(1)\*230;V(2)\*230;V(3)\*230;V(4)\*230];

table(BUS\_NUMBER,VOLTAGE\_Kv)

%Slack bus power display:

SLACK\_BUS\_POWER =["Injected power (MVA)";"Generated power (MVA)"];

S=[Slack\_Bus\_Injected\_Power\*100; Slack\_Bus\_Generated\_Power\*100];

table(SLACK\_BUS\_POWER,S)

%Line flows and lines losses display:

LINE\_NUMBER =["Line 1-2";"Line 1-3";"Line 4-3";"Line 4-2";];

LINE\_FLOW\_MVA = [S12\*100;S13\*100;S43\*100;S42\*100];

DIRECTION\_OF\_FLOW = ["Bus 1 to Bus 2";"Bus 1 to Bus 3";"Bus 4 to Bus 3";"Bus 4 to Bus 2";];

LINE\_LOSSES\_MVA = [Line1\_2Losses\*100;Line1\_3Losses\*100;Line4\_3Losses\*100;Line4\_2Losses\*100];

table(LINE\_NUMBER,LINE\_FLOW\_MVA,DIRECTION\_OF\_FLOW,LINE\_LOSSES\_MVA)

elseif strcmp(method, 'GS')

%% Power flow analysis using Gauss method

Sbase = 100; Vbase = 230;

%% Lines data: Shunt Y is nominated by Yshunti\_j

% Line 1-2:

R1\_2 = 0.01008; X1\_2 = 0.05040i; Yshunt1\_2 = 0.05125i;

Y1\_2 = 1/(R1\_2 + X1\_2);

% Line 1-3:

R1\_3 = 0.00744; X1\_3 = 0.03720i; Yshunt1\_3 = 0.03975i;

Y1\_3 = 1/(R1\_3 + X1\_3);

% Line 2-4:

R2\_4 = 0.00744; X2\_4 = 0.03720i; Yshunt2\_4 = 0.03875i;

Y2\_4 = 1/(R2\_4 + X2\_4);

% Line 3\_4:

R3\_4 = 0.01272; X3\_4 = 0.06360i; Yshunt3\_4 = 0.06375i;

Y3\_4 = 1/(R3\_4 + X3\_4);

%-----------------------------------------------------------------------------------------------------------------------------------------------------------------

%% Buses data:

% Bus 1: Slack bus

Pload\_1 = 50; Qload\_1 = 30.99; V\_1 = 1;

Ploadpu\_1 = Pload\_1/Sbase; Qloadpu\_1 = Qload\_1/Sbase;

% Bus 2: Load bus (inductive)

Pload\_2 = 170; Qload\_2 = 105.35; V\_2 = 1;

Ploadpu\_2 = -Pload\_2/Sbase; Qloadpu\_2 = -Qload\_2/Sbase;

% Bus 3: Load bus (inductive)

Pload\_3 = 200; Qload\_3 = 123.94; V\_3 = 1;

Ploadpu\_3 = -Pload\_3/Sbase; Qloadpu\_3 = -Qload\_3/Sbase;

% Bus 4: Voltage controlled

Pgen\_4 = 318; Pload\_4 = 80; Qload\_4 = 49.58; V\_4 = 1.02;

Ploadpu\_4 = (Pgen\_4 - Pload\_4)/Sbase; Q\_4 = 0;

%-----------------------------------------------------------------------------------------------------------------------------------------------------------------

%% Admittance bus:

Y11 = Y1\_2 + Y1\_3 + Yshunt1\_2 + Yshunt1\_3; Y12 = -Y1\_2; Y13 = -Y1\_3; Y14 = 0;

Y21 = -Y1\_2; Y22 = Y1\_2 + Y2\_4 + Yshunt1\_2 + Yshunt2\_4; Y23 = 0; Y24 = -Y2\_4;

Y31 = -Y1\_3; Y32 = 0; Y33 = Y1\_3 + Y3\_4 + Yshunt1\_3 + Yshunt3\_4; Y34 = -Y3\_4;

Y41 = 0; Y42 = -Y2\_4; Y43 = -Y3\_4; Y44 = Y2\_4 + Y3\_4 + Yshunt2\_4 + Yshunt3\_4;

Y = [ Y11 Y12 Y13 Y14 ; Y21 Y22 Y23 Y24 ; Y31 Y32 Y33 Y34 ; Y41 Y42 Y43 Y44];

%-----------------------------------------------------------------------------------------------------------------------------------------------------------------

%% Define Vitertion array which contains the current iteration of each bus

%% voltage, which starts with the initial conditions as the following:

Viteration = [ V\_1 ; V\_2 ; V\_3 ; V\_4 ];

%% Define number of iterations: (15 iterations was selected to be the

%% required number of iterations till steady state results by trial & error)

Niteration = 15;

%% Bus voltage calculations based on the selected number of iterations:

for i = 1:Niteration

for n = 2:4

if n == 2

Viteration(2) = (1/Y(2,2))\*(((Ploadpu\_2 - Qloadpu\_2\*1i)/(conj(Viteration(2)))) - Y(2,1)\*Viteration(1) - Y(2,3)\*Viteration(3) - Y(2,4)\*Viteration(4));

elseif n == 3

Viteration(3) = (1/Y(3,3))\*(((Ploadpu\_3 - Qloadpu\_3\*1i)/(conj(Viteration(3)))) - Y(3,1)\*Viteration(1) - Y(3,2)\*Viteration(2) - Y(3,4)\*Viteration(4));

elseif n == 4

Q\_4 = -imag((conj(Viteration(4))\*(Y(4,1)\*Viteration(1) + Y(4,2)\*Viteration(2) + Y(4,3)\*Viteration(3) + Y(4,4)\*Viteration(4))));

Viteration(4) = (1/Y(4,4))\*(((Ploadpu\_4 - Q\_4\*1i)/(conj(Viteration(4)))) - Y(4,1)\*Viteration(1) - Y(4,2)\*Viteration(2) - Y(4,3)\*Viteration(3));

Viteration(4) = V\_4\*(cos(angle(Viteration(4))) + sin(angle(Viteration(4)))\*1i);

end

end

end

%-----------------------------------------------------------------------------------------------------------------------------------------------------------------

%% Slack bus power calculation:

Pslack\_inj = real(V\_1\*conj((Y(1,1)\*V\_1 + Y(1,2)\*Viteration(2) + Y(1,3)\*Viteration(3) + Y(1,4)\*Viteration(4))));

Qslack\_inj = imag(V\_1\*conj((Y(1,1)\*V\_1 + Y(1,2)\*Viteration(2) + Y(1,3)\*Viteration(3) + Y(1,4)\*Viteration(4))));

Pslack\_gen = Pslack\_inj + Ploadpu\_1;

Qslack\_gen = Qslack\_inj + Qloadpu\_1;

Sslack\_inj = Pslack\_inj + Qslack\_inj\*1i;

Sslack\_gen = Pslack\_gen + Qslack\_gen\*1i;

%% Line flow & power losses:

% Line 1-2:

S1\_2 = V\_1\*(conj((V\_1 - Viteration(2))\*Y1\_2 + V\_1\*Yshunt1\_2));

S2\_1 = Viteration(2)\*(conj((Viteration(2) - V\_1)\*Y1\_2 + Viteration(2)\*Yshunt1\_2));

Slosses1\_2 = S1\_2 + S2\_1;

% Line 1-3:

S1\_3 = V\_1\*(conj((V\_1 - Viteration(3))\*Y1\_3 + V\_1\*Yshunt1\_3));

S3\_1 = Viteration(3)\*(conj((Viteration(3) - V\_1)\*Y1\_3 + Viteration(3)\*Yshunt1\_3));

Slosses1\_3 = S1\_3 + S3\_1;

% Line 2-4:

S2\_4 = Viteration(2)\*(conj((Viteration(2) - Viteration(4))\*Y2\_4 + Viteration(2)\*Yshunt2\_4));

S4\_2 = Viteration(4)\*(conj((Viteration(4) - Viteration(2))\*Y2\_4 + Viteration(4)\*Yshunt2\_4));

Slosses2\_4 = S2\_4 + S4\_2;

% Line 3-4:

S3\_4 = Viteration(3)\*(conj((Viteration(3) - Viteration(4))\*Y3\_4 + Viteration(3)\*Yshunt3\_4));

S4\_3 = Viteration(4)\*(conj((Viteration(4) - Viteration(3))\*Y3\_4 + Viteration(4)\*Yshunt3\_4));

Slosses3\_4 = S3\_4 + S4\_3;

%-----------------------------------------------------------------------------------------------------------------------------------------------------------------

%% Results display:

%Bus voltages display:

Bus\_number = ["Bus 1";"Bus 2"; "Bus 3"; "Bus 4"];

Bus\_voltage\_in\_kV =Vbase\*Viteration;

table(Bus\_number,Bus\_voltage\_in\_kV)

%Slack bus power display:

Slack\_bus\_data = ["Injected power in MVA";"Generated power in MVA"];

Slack\_bus\_power = Sbase\*[Sslack\_inj; Sslack\_gen];

table(Slack\_bus\_data,Slack\_bus\_power)

%Line flows and lines losses display:

Line =["Line 1-2";"Line 1-3";"Line 2-4";"Line 3-4";];

Line\_flow = Sbase\*[S1\_2; S1\_3; S4\_2; S4\_3];

Direction = ["Bus 1 to Bus 2";"Bus 1 to Bus 3";"Bus 4 to Bus 3";"Bus 4 to Bus 2";];

Line\_losses = Sbase\*[Slosses1\_2; Slosses1\_3; Slosses2\_4; Slosses3\_4];

table(Line, Line\_flow,Direction ,Line\_losses)

%-----------------------------------------------------------------------------------------------------------------------------------------------------------------

else

disp('Invalid method. Please enter NR or GS.');

end

Figure

# REFERENCES

Analysis, P. S. (1998). *Power System Analysis.* McGraw-Hill College.

Introduction, E. P. (2018). *Syed A. Nasar.* Routledge.

J. Duncan Glover, M. S. (2016). *Power Systems Analysis and Design.* Cengage Learning.

*Newton Raphson Method for Load Flow Analysis*. (2016, November 21). Retrieved from eeeguide.com: https://www.eeeguide.com/newton-raphson-method-for-load-flow-analysis/

W. Hubbi, A. R. (1983). Starting algorithm and modification for Newton-Raphson load-flow method. *International Journal of Electrical Power & Energy Systems, 5*(3), 166-172. doi:https://doi.org/10.1016/0142-0615(83)90005-4.