	CAIRO UNIVERSITY		FACULTY OF ENGINEERING
4th Year			
Course: Electrical Power Systems (3) (EPE4010)			

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

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 FIGURES

TABLE OF EQUATIONS

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

$$f_{iP}(|V|,\ \delta) = P_i\ (\text{specified}) - P_i = 0 \qquad (6.60a)$$

$$f_{iQ}(|V|,\ \delta) = Q_i\ (\text{specified}) - Q_i = 0 \qquad (6.60b)$$

Equation 1

Equation 2

where expressions for P_i and Q_i are given in Eqs. (6.27) and (6.28). For a trial set of variables $|V_i|$, δ_i, the vector of residuals f^0 of Eq. (6.57) corresponds to

$$f_{iP} = P_i \text{ (specified)} - P_i \text{ (calculated)} = \Delta P_i \tag{6.61a}$$

$$f_{iQ} = Q_i \text{ (specified)} - Q_i \text{ (calculated)} = \Delta Q_i \tag{6.61b}$$

Equation 3

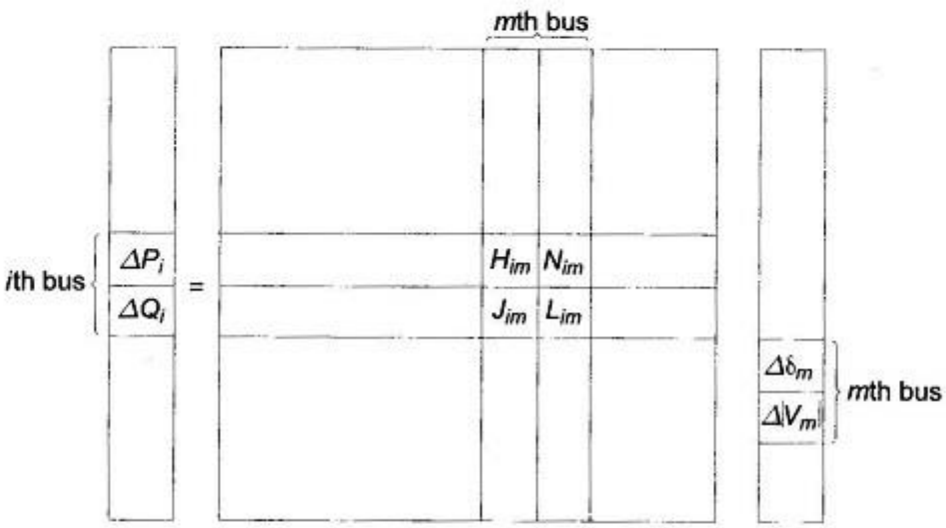
Equation 4

while the vector of corrections , Δx^0 corresponds to

$$\Delta|V_i|,\ \Delta\delta_i$$

Equation 5

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



(6.62a)

Equation 6

where

$$
\begin{aligned}
H_{im} &= \frac{\partial P_i}{\partial \delta_m} \\
N_{im} &= \frac{\partial P_i}{\partial |V_m|} \\
J_{im} &= \frac{\partial Q_i}{\partial \delta_m} \\
L_{im} &= \frac{\partial Q_i}{\partial |V_m|}
\end{aligned}
\qquad (6.63a)
$$

Equation 7

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), P_1 and Q_1 are unspecified and $|V_1|$, δ_1 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, Q_i 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 ΔP_i , i.e.

$$
i\text{th bus}\ \begin{bmatrix} \Delta P_i \end{bmatrix} = \begin{bmatrix} \cdots & \overbrace{H_{im} \;\; N_{im}}^{m\text{th bus}} & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ \left.\begin{matrix} \Delta \delta_m \\ \Delta |V_m| \end{matrix}\right\} m\text{th bus} \\ \vdots \end{bmatrix} \qquad (6.62b)
$$

Equation 8

If the mth bus is also a PV bus, $|V_m|$ becomes fixed so that $\Delta|V_m| = 0$. We can now write

$$\text{ith bus}\;\begin{bmatrix}\Delta P_i\end{bmatrix} = \begin{array}{c}\hspace{4em}\text{mth bus}\\ \begin{bmatrix} \quad & H_{im} & \quad \end{bmatrix}\end{array} \begin{bmatrix} \\ \\ \Delta\delta_m \\ \\ \end{bmatrix}\text{mth bus} \qquad (6.62c)$$

Equation 9

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

$$\text{ith bus}\;\begin{bmatrix}\Delta P_i\\ \Delta Q_i\end{bmatrix} = \begin{array}{c}\hspace{4em}\text{mth bus}\\ \begin{bmatrix} \quad & H_{im} & \quad \\ \quad & J_{im} & \quad \end{bmatrix}\end{array} \begin{bmatrix} \\ \\ \Delta\delta_m \\ \\ \end{bmatrix}\text{mth bus} \qquad (6.62d)$$

Equation 10

It is convenient for numerical solution to normalize the voltage corrections

$$\frac{\Delta |V_m|}{|V_m|}$$

Equation 11

as a consequence of which, the corresponding Jacobian elements become

$$N_{im} = \frac{\partial P_i}{\partial |V_m|} |V_m|$$

$$L_{im} = \frac{\partial Q_i}{\partial |V_m|} |V_m| \tag{6.63b}$$

Equation 12

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

Case 1

$$m \neq i$$

$$H_{im} = L_{im} = a_m f_i - b_m e_i \tag{6.64}$$

$$N_{im} = - J_{im} = a_m e_i + b_m f_i$$

Equation 13

where

$$Y_{im} = G_{im} + jB_{im}$$

$$V_i = e_i + jf_i$$

$$(a_m + jb_m) = (G_{im} + jB_{im})\ (e_m + jf_m)$$

Equation 14

Case 2

$$m = i$$

$$H_{ii} = - Q_i - B_{ii}|V_i|^2$$

$$N_{ii} = P_i + G_{ii}|V_i|^2 \tag{6.65}$$

Equation 15

$$J_{ii} = P_i - G_{ii}|V_i|^2$$

$$L_{ii} = Q_i - B_{ii}|V_i|^2$$

Equation 16

An important observation can be made in respect of the Jacobian by examination of the Y_{BUS} matrix. If buses i and m are not connected, $Y_{im} = 0$ ($G_{im} = B_{im} = 0$). Hence from Eqs. (6.63) and (6.64), we can write

$$
\begin{aligned}
H_{im} &= H_{mi} = 0 \\
N_{im} &= N_{mi} = 0 \\
J_{im} &= J_{mi} = 0 \\
L_{im} &= L_{mi} = 0
\end{aligned}
\tag{6.66}
$$

Equation 17

Thus the Jacobian is as sparse as the Y_{BUS} matrix.

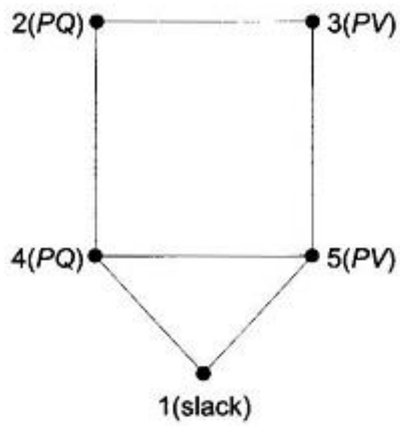


Fig. 6.10 Sample five-bus network

Figure 1

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 $[\delta_2 |V_2| \delta_3 \delta_4 |V_4| \delta_5]^T$, the vector of residuals $[\Delta P_2\ \Delta Q_2\ \Delta P_3\ \Delta P_4\ \Delta Q_4\ \Delta P_5]^T$ and 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

$$
\left[\Delta\delta_2 \, \frac{\Delta |V_2|}{|V_2|} \, \Delta\delta_3 \Delta\delta_4 \, \frac{\Delta |V_4|}{|V_4|} \, \Delta\delta_5 \right]^T
$$

Equation 18

Corrections are then added to update the vector of variables.

$$
\underbrace{\begin{array}{c|c}
 & \\
2 & \Delta P_2 \\
 & \Delta Q_2 \\
3 & \Delta P_3 \\
4 & \Delta P_4 \\
 & \Delta Q_4 \\
5 & \Delta P_5
\end{array}}_{\text{Residuals}}
=
\underbrace{\begin{array}{cccccc}
2 & & 3 & 4 & & 5 \\
\hline
H_{22} & N_{22} & H_{23} & H_{24} & N_{24} & \\
J_{22} & L_{22} & J_{23} & J_{24} & L_{24} & \\
H_{32} & N_{32} & H_{33} & & & H_{35} \\
H_{42} & N_{42} & & H_{44} & N_{44} & H_{45} \\
J_{42} & L_{42} & & J_{44} & L_{44} & J_{45} \\
 & & H_{53} & H_{54} & N_{54} & H_{55}
\end{array}}_{\substack{\text{Jacobian} \\ \text{(Evaluated at trial values of variables)}}}
\underbrace{\begin{array}{c}
 \\
\Delta\delta_2 \\
\dfrac{\Delta|V_2|}{|V_2|} \\
\Delta\delta_3 \\
\Delta\delta_4 \\
\dfrac{\Delta|V_4|}{|V_4|} \\
\Delta\delta_5
\end{array}}_{\substack{\text{Corrections} \\ \text{in variables}}}
\qquad (6.67)
$$

Equation 19

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 $\delta = 0$) 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 ΔP_i (for PV and PQ buses) and ΔQ_i, (for all PQ buses) from (6.60a and b). If all the values are less than the prescribed tolerance, stop the iterations, calculate P_1 and Q_1 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


```
>> PS3_project
Enter the desired method to solve the problem (NR - GS)
Method: NR

ans =

  4×2 table

    BUS_NUMBER      VOLTAGE_Kv
    __________    ______________

     "Bus 1"         230+0i
     "Bus 2"      225.92-3.8648i
     "Bus 3"      222.78-7.2733i
     "Bus 4"      234.52+6.2078i


ans =

  2×2 table

        SLACK_BUS_POWER              S
    _______________________    ______________

    "Injected power (MVA)"     136.79+83.23i
    "Generated power (MVA)"    186.79+114.22i


ans =

  4×4 table

    LINE_NUMBER    LINE_FLOW_MVA     DIRECTION_OF_FLOW    LINE_LOSSES_MVA
    ___________    ______________    _________________    _______________

    "Line 1-2"      38.82+22.274i    "Bus 1 to Bus 2"     0.22757-8.9335i
    "Line 1-3"     97.974+60.956i    "Bus 1 to Bus 3"      1.0269-2.3799i
    "Line 4-3"     104.88+57.158i    "Bus 4 to Bus 3"      1.8268-3.4455i
    "Line 4-2"     133.12+74.937i    "Bus 4 to Bus 2"      1.7132+0.7945i

>>
```


Figure 2

Example 2 using GS method

MATLAB Command Window   Page 1

```
>> PS3_project
Enter the desired method to solve the problem (NR - GS)
Method: GS

ans =

  4×2 table

    Bus_number    Bus_voltage_in_kV
    __________    _________________

     "Bus 1"           230+0i
     "Bus 2"       225.92-3.8498i
     "Bus 3"       222.76-7.2824i
     "Bus 4"       234.52+6.235i


ans =

  2×2 table

         Slack_bus_data          Slack_bus_power
    ________________________    _______________

    "Injected power in MVA"     136.81+83.347i
    "Generated power in MVA"    186.81+114.34i


ans =

  4×4 table

       Line          Line_flow          Direction          Line_losses
    __________    ______________    ________________    ______________

    "Line 1-2"    38.695+22.298i    "Bus 1 to Bus 2"    0.22673-8.9377i
    "Line 1-3"    98.119+61.049i    "Bus 1 to Bus 3"     1.0309-2.5533i
    "Line 2-4"    133.25+74.92i     "Bus 4 to Bus 3"     1.7155+0.8059i
    "Line 3-4"    104.75+56.893i    "Bus 4 to Bus 2"     1.8349-3.4445i

>>
```


Figure 3

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 4

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.