EE 523 | Spring 2023

Assignment 01

Submitted by

Sajjad Uddin Mahmud

WSU ID: 011789534

Code:

Platform: MATLAB

Main Code :

```
%% EE 523 Assignment 01 - Sajjad Uddin Mahmud - Spring 2023 - WSU

%% Basic Initialization
clc;
clear all;
close all;

%% Adding Folder Path
Data_Path = [fileparts(pwd),'/Data'];
addpath(Data_Path);

Function_Path = [fileparts(pwd),'/Functions'];
addpath(Function_Path);

%% Setting Up The Input Data As Per Assignment
Problem = 'A';
if Problem == 'A'
    Excel_Worksheet = 'Problem_A';
elseif Problem == 'B'
    Excel_Worksheet = 'Problem_B';
elseif Problem == 'C'
    Excel_Worksheet = 'Problem_C';
end

%% Reading From Bus Data
%% Bus Number
All_Bus_Number = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'A4:A14'); % Reading All Bus
ID Data
Total_Bus = length(All_Bus_Number); % Calculating Total Bus Number

%% Bus Type
All_Bus_Type = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'G4:G14'); % Reading All Bus
Type Data
PQ_Bus_Type = 0;
PQ_Bus_Type_1 = 1;
PV_Bus_Type = 2;
Slack_Bus_Type = 3;
Bus = All_Bus_Number(find(All_Bus_Type ~= Slack_Bus_Type)); % Bus Type Data Except the Slack Bus
Bus_Type = All_Bus_Type(Bus); % Bus Type Data Except the Slack Bus

%% Bus Information
% Slack_Bus_Number = 1

Base_MVA = 100;
V_Desired = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'O4:O14'); % Given Desired Voltage
Delta_in_Rad = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'I4:I14'); % Given Voltage Angle
P_Load = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'J4:J14')/Base_MVA; % Load MW pu
Q_Load = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'K4:K14')/Base_MVA; % Load MVAR pu
```

```
P_Gen = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'L4:L14')/Base_MVA; % Generator MW pu
Q_Gen = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'M4:M14')/Base_MVA; % Generator MVAR pu

%% Reading from Branch Data
%% Branch Number
From_Bus = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'A18:A27');
To_Bus = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'B18:B27');

%% Bus Shunt Conductance and Shunt Susceptance
G_Shunt_Bus = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'R4:R14');
B_Shunt_Bus = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'S4:S14');

%% Calculating Bus Shunt Admittance
Y_Shunt_Bus = G_Shunt_Bus + j.*B_Shunt_Bus;

%% Branch Resistance Per Unit
R_Branch = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'G18:G27');

%% Branch Reactance Per Unit
X_Branch = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'H18:H27');

%% Line Charging B Per Unit
B_Branch = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'I18:I27');

%% Transformer Turns Ratio
XFR_TurnRatio = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'O18:O27');

%% Calculating Branch Impedence and Admittance
for i=1:length(From_Bus)
    Z_Branch(i) = R_Branch(i) + j * X_Branch(i); % Per Unit Impedance
    Y_Branch(i) = 1 / Z_Branch(i); % Per Unit Admittance
end

%% Tap Consideration
Tap_Consideration = 1; % 0 = Without Taps, 1 = With Taps
if (Tap_Consideration == 0)
    for i = 1:length(XFR_TurnRatio)
        XFR_TurnRatio(i) = 0; % If We Do Not Consider Tap, All the Turn Ratio of Transformer are 0
    end
end

%% Calculating Y Bus Matrix:

% Initialization
Y_Bus = zeros(Total_Bus,Total_Bus);

% LOOP: Computing Off-Diagonal Elements
for i=1:length(Y_Branch)
    if (XFR_TurnRatio(i)==0)
        Y_Bus(From_Bus(i),To_Bus(i)) = - Y_Branch(i);
        Y_Bus(To_Bus(i),From_Bus(i)) = - Y_Branch(i);
        Y_Bus_Diag(From_Bus(i),To_Bus(i)) = - Y_Branch(i);
        Y_Bus_Diag(To_Bus(i),From_Bus(i)) = - Y_Branch(i);
    else
        T = (1/(XFR_TurnRatio(i)));
        Y_Bus(From_Bus(i),To_Bus(i)) = - Y_Branch(i) * (T);
        Y_Bus(To_Bus(i),From_Bus(i)) = - Y_Branch(i) * (T);
        Y_Bus_Diag(From_Bus(i),To_Bus(i)) = - Y_Branch(i);
        Y_Bus_Diag(To_Bus(i),From_Bus(i)) = - Y_Branch(i) * (T^2);
    end
end
```

```
% LOOP: Computing Diagonal Elements
Y_Bus_Sum = sum(Y_Bus_Diag);
for i=1:Total_Bus
Y_Bus(i,i) = -Y_Bus_Sum(i) + Y_Shunt_Bus(i); % Adding Shunt Capacitance
end

% LOOP: Adding Line Charaging Capacitance
for i=1:length(From_Bus)
    Y_Bus(From_Bus(i),From_Bus(i)) = Y_Bus(From_Bus(i),From_Bus(i)) + j * (B_Branch(i) / 2);
    Y_Bus(To_Bus(i),To_Bus(i)) = Y_Bus(To_Bus(i),To_Bus(i)) + j * (B_Branch(i) / 2);
end

% Converting Y Bus Data into Polar Form
Rho = abs(Y_Bus); % Magnitude of Y Bus Entries
Theta = angle(Y_Bus); % Angle of Y Bus Entries in radian
B = imag(Y_Bus); % Imaginary Part of Y Bus Entries
G = real(Y_Bus); % Real Part of Y Bus Entries

% End of Y Bus Formation. Y Bus is Ready

%% Power Flow

%% Method
    Task = 1; % Newton-Raphson = 1; Fast Decoupled = 2

    %% Power Flow - Newton-Raphson Method
    if Task == 1
        [V, Delta_in_Rad, Iteration] = Newton_Raphson_Function(Y_Bus, V_Desired, Delta_in_Rad,
P_Gen, P_Load, Q_Gen, Q_Load, All_Bus_Number, All_Bus_Type);


    %%  Power Flow - Fast Decoupled
    elseif Task == 2
        [V_FD, Delta_in_Rad_FD, Iteration_FD] = Newton_Raphson_Function_1(Y_Bus, V_Desired,
Delta_in_Rad, P_Gen, P_Load, Q_Gen, Q_Load, All_Bus_Number, All_Bus_Type); % Putting Values after
4 Iterations of NR as Input for Fast Decoupled

        V_FD = transpose(V_FD);
        Delta_in_Rad_FD = transpose(Delta_in_Rad_FD);

        [V, Delta_in_Rad, Iteration] = Fast_Decoupled_Function(Y_Bus, V_FD, Delta_in_Rad_FD,
P_Gen, P_Load, Q_Gen, Q_Load, All_Bus_Number, All_Bus_Type);

    end
```

Function - Newton-Raphson:

```
%% EE 523 Assignment 01 - Sajjad Uddin Mahmud - Spring 2023 - WSU

%% Basic Initialization
clc;
clear all;
close all;


%% Setting Up The Input Data As Per Assignment
Problem = 'A';
if Problem == 'A'
    Excel_Worksheet = 'Problem_A';
elseif Problem == 'B'
    Excel_Worksheet = 'Problem_B';
elseif Problem == 'C'
    Excel_Worksheet = 'Problem_C';
end

%% Reading From Bus Data
%% Bus Number
All_Bus_Number = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'A4:A14'); % Reading All Bus
ID Data
Total_Bus = length(All_Bus_Number); % Calculating Total Bus Number

%% Bus Type
All_Bus_Type = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'G4:G14'); % Reading All Bus
Type Data
PQ_Bus_Type = 0;
PQ_Bus_Type_1 = 1;
PV_Bus_Type = 2;
Slack_Bus_Type = 3;
Bus = All_Bus_Number(find(All_Bus_Type ~= Slack_Bus_Type)); % Bus Type Data Except the Slack Bus
Bus_Type = All_Bus_Type(Bus); % Bus Type Data Except the Slack Bus

%% Bus Information
% Slack_Bus_Number = 1

Base_MVA = 100;
V_Desired = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'O4:O14'); % Given Desired Voltage
Delta_in_Rad = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'I4:I14'); % Given Voltage Angle
P_Load = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'J4:J14')/Base_MVA; % Load MW pu
Q_Load = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'K4:K14')/Base_MVA; % Load MVAR pu
P_Gen = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'L4:L14')/Base_MVA; % Generator MW pu
Q_Gen = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'M4:M14')/Base_MVA; % Generator MVAR pu

%% Reading from Branch Data
%% Branch Number
From_Bus = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'A18:A27');
To_Bus = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'B18:B27');

%% Bus Shunt Conductance and Shunt Susceptance
G_Shunt_Bus = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'R4:R14');
B_Shunt_Bus = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'S4:S14');

%% Calculating Bus Shunt Admittance
Y_Shunt_Bus = G_Shunt_Bus + j.*B_Shunt_Bus;
```

```
%% Branch Resistance Per Unit
R_Branch = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'G18:G27');

%% Branch Reactance Per Unit
X_Branch = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'H18:H27');

%% Line Charging B Per Unit
B_Branch = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'I18:I27');

%% Transformer Turns Ratio
XFR_TurnRatio = xlsread('Kundur_11Bus_System.xlsx',Excel_Worksheet,'O18:O27');

%% Calculating Branch Impedence and Admittance
for i=1:length(From_Bus)
    Z_Branch(i) = R_Branch(i) + j * X_Branch(i); % Per Unit Impedance
    Y_Branch(i) = 1 / Z_Branch(i); % Per Unit Admittance
end

%% Tap Consideration
Tap_Consideration = 1; % 0 = Without Taps, 1 = With Taps
if (Tap_Consideration == 0)
    for i = 1:length(XFR_TurnRatio)
        XFR_TurnRatio(i) = 0; % If We Do Not Consider Tap, All the Turn Ratio of Transformer are 0
    end
end

%% Calculating Y Bus Matrix:

% Initialization
Y_Bus = zeros(Total_Bus,Total_Bus);

% LOOP: Computing Off-Diagonal Elements
for i=1:length(Y_Branch)
    if (XFR_TurnRatio(i)==0)
        Y_Bus(From_Bus(i),To_Bus(i)) = - Y_Branch(i);
        Y_Bus(To_Bus(i),From_Bus(i)) = - Y_Branch(i);
        Y_Bus_Diag(From_Bus(i),To_Bus(i)) = - Y_Branch(i);
        Y_Bus_Diag(To_Bus(i),From_Bus(i)) = - Y_Branch(i);
    else
        T = (1/(XFR_TurnRatio(i)));
        Y_Bus(From_Bus(i),To_Bus(i)) = - Y_Branch(i) * (T);
        Y_Bus(To_Bus(i),From_Bus(i)) = - Y_Branch(i) * (T);
        Y_Bus_Diag(From_Bus(i),To_Bus(i)) = - Y_Branch(i);
        Y_Bus_Diag(To_Bus(i),From_Bus(i)) = - Y_Branch(i) * (T^2);
    end
end

% LOOP: Computing Diagonal Elements
Y_Bus_Sum = sum(Y_Bus_Diag);
for i=1:Total_Bus
Y_Bus(i,i) = -Y_Bus_Sum(i) + Y_Shunt_Bus(i); % Adding Shunt Capacitance
end

% LOOP: Adding Line Charaging Capacitance
for i=1:length(From_Bus)
    Y_Bus(From_Bus(i),From_Bus(i)) = Y_Bus(From_Bus(i),From_Bus(i)) + j * (B_Branch(i) / 2);
    Y_Bus(To_Bus(i),To_Bus(i)) = Y_Bus(To_Bus(i),To_Bus(i)) + j * (B_Branch(i) / 2);
end

% Converting Y Bus Data into Polar Form
Rho = abs(Y_Bus); % Magnitude of Y Bus Entries
```

```
Theta = angle(Y_Bus); % Angle of Y Bus Entries in radian
B = imag(Y_Bus); % Imaginary Part of Y Bus Entries
G = real(Y_Bus); % Real Part of Y Bus Entries

% End of Y Bus Formation. Y Bus is Ready

%% Power Flow

%% Method
    Task = 1; % Newton-Raphson = 1; Fast Decoupled = 2

    %% Power Flow - Newton-Raphson Method
    if Task == 1
        [V, Delta_in_Rad, Iteration] = Newton_Raphson_Function(Y_Bus, V_Desired, Delta_in_Rad,
P_Gen, P_Load, Q_Gen, Q_Load, All_Bus_Number, All_Bus_Type);


    %%  Power Flow - Fast Decoupled
    elseif Task == 2
        [V_FD, Delta_in_Rad_FD, Iteration_FD] = Newton_Raphson_Function_1(Y_Bus, V_Desired,
Delta_in_Rad, P_Gen, P_Load, Q_Gen, Q_Load, All_Bus_Number, All_Bus_Type); % Putting Values after
4 Iterations of NR as Input for Fast Decoupled

        V_FD = transpose(V_FD);
        Delta_in_Rad_FD = transpose(Delta_in_Rad_FD);

        [V, Delta_in_Rad, Iteration] = Fast_Decoupled_Function(Y_Bus, V_FD, Delta_in_Rad_FD,
P_Gen, P_Load, Q_Gen, Q_Load, All_Bus_Number, All_Bus_Type);

    end
```

Function - Newton-Raphson (For Getting Input of Fast Decoupled):

```
%% Power Flow Function: Newton-Raphson

function [V, Delta_in_Rad, Iteration] = Newton_Raphson_Function_1(Y_Bus, V_Desired, Delta_in_Rad,
P_Gen, P_Load, Q_Gen, Q_Load, All_Bus_Number, All_Bus_Type)

%% Basic Initialization
Total_Bus = length(All_Bus_Number);
PQ_Bus_Type = 0;
PQ_Bus_Type_1 = 1;
PV_Bus_Type = 2;
Slack_Bus_Type = 3;
Bus = All_Bus_Number(find(All_Bus_Type ~= Slack_Bus_Type)); % Bus Type Data Except the Slack Bus
Bus_Type = All_Bus_Type(Bus); % Bus Type Data Except the Slack Bus
Base_MVA = 100;

%% Power Flow

% Schedule Real and Reactive Power
P_Scheduled = transpose(P_Gen - P_Load);
Q_Scheduled = transpose(Q_Gen - Q_Load);

% Initial Voltage Magnitude
V = ones(1,length(All_Bus_Number));
V(1,find(V_Desired)) = V_Desired(find(V_Desired),1);

% Initialization
Iteration = 0;
Tolerance = 0.01;
while 1

    %% Calculating Real Power

    % Initialization
    P_Calculated = zeros(1,Total_Bus);

    % LOOP: Computing Real Power
    for i=1:Total_Bus
        for n=1:Total_Bus
            P_Calculated(i) = P_Calculated(i) + (abs(abs(Y_Bus(i,n)) * V(i) * V(n))) *
(cos(angle(Y_Bus(i,n)) + Delta_in_Rad(n) - Delta_in_Rad(i)));
        end
    end

    %% Calculating Reactive Power

    % Initialization
    Q_Calculated = zeros(1,Total_Bus);

    % LOOP: Computing Reactive Power
    for i=1:Total_Bus
        for n=1:Total_Bus
            Q_Calculated(i) = Q_Calculated(i) + (abs(abs(Y_Bus(i,n)) * V(i) * V(n))) *
(sin(angle(Y_Bus(i,n)) + Delta_in_Rad(n) - Delta_in_Rad(i)));
        end
        Q_Calculated(i) = - Q_Calculated(i);
    end

    %% Calculating Mismatch
```

```
    Delta_P = P_Scheduled - P_Calculated;
    Delta_Q = Q_Scheduled - Q_Calculated;

% Initializating Jacobian Matrix
        J11 = zeros(length(Bus));
        J12 = zeros(length(Bus));
        J21 = zeros(length(Bus));
        J22 = zeros(length(Bus));

        % LOOP: Computing Jacobian Matrix for All the Buses Except Slack Bus
        for i=1:length(Bus)
            for j=1:length(Bus)
                if (i==j)
                    J11(i,j) = - Q_Calculated(Bus(i)) - ((V(Bus(i)))^2) *
(imag(Y_Bus(Bus(i),Bus(i))));
                    J21(i,j) = P_Calculated(Bus(i)) - ((V(Bus(i)))^2) *
(real(Y_Bus(Bus(i),Bus(i))));
                    J12(i,j) = P_Calculated(Bus(i)) + ((V(Bus(i)))^2) *
(real(Y_Bus(Bus(i),Bus(i))));
                    J22(i,j) = Q_Calculated(Bus(i)) - ((V(Bus(i)))^2) *
(imag(Y_Bus(Bus(i),Bus(i))));
                else
                    J11(i,j) = - abs(V(Bus(i)) * V(Bus(j)) * abs(Y_Bus(Bus(i),Bus(j)))) *
sin(angle(Y_Bus(Bus(i),Bus(j))) + Delta_in_Rad(Bus(j)) - Delta_in_Rad(Bus(i)));
                    J21(i,j) = - abs(V(Bus(i)) * V(Bus(j)) * abs(Y_Bus(Bus(i),Bus(j)))) *
cos(angle(Y_Bus(Bus(i),Bus(j))) + Delta_in_Rad(Bus(j)) - Delta_in_Rad(Bus(i)));
                    J12(i,j) = - J21(i,j);
                    J22(i,j) = J11(i,j);
                end
            end
        end

        % Removing Rows and Columns from Jacobian for PV Bus
        PV = find(Bus_Type==PV_Bus_Type);
        J12(:,PV) = [];
        J21(PV,:) = [];
        J22(:,PV) = [];
        J22(PV,:) = [];
        J = [J11 J12; J21 J22];

        % Delta
        Delta_J = Delta_in_Rad(find(All_Bus_Type ~= Slack_Bus_Type));
        V_J = V(find((All_Bus_Type == PQ_Bus_Type)));
        Delta_P_J = Delta_P(find(All_Bus_Type ~= Slack_Bus_Type));
        Delta_Q_J = Delta_Q(find((All_Bus_Type == PQ_Bus_Type)));
        Delta_P_Q = [transpose(Delta_P_J);transpose(Delta_Q_J)];

        %% Updating V and Delta through LU Factorization

        % Function Calling: LU Factorization Using Dolittle's Method
        [V_Delta_Corrected] = LU_Factorization_Dolittle_Function(J,Delta_P_Q);

        % LOOP: Sorting the Voltages and Angles after LU Factorization
        for i=1:length(V_Delta_Corrected)
            if (i <= length(Delta_P_J))
                Delta_Corrected(i) = V_Delta_Corrected(i);
            else
                V_Corrected(i-length(Delta_P_J)) = V_Delta_Corrected(i);
            end
        end
```

```
        % Updating Voltages and Angles
        Delta_Updated = Delta_J + Delta_Corrected;
        V_Updated = V_J .* (1 + V_Corrected);

        % Preparing for Next Iteration
        V_i = (find((All_Bus_Type == PQ_Bus_Type)));
        Delta_i = find(All_Bus_Type ~= Slack_Bus_Type);

        for i=1:length(Delta_i)
            Delta_New(Delta_i(i)) = Delta_Updated(i);
        end

        for i=1:length(V_i)
            V_Desired(V_i(i)) = V_Updated(i);
        end

        V = transpose(V_Desired);
        Delta_in_Rad = Delta_New;
        Delta_in_Degree = (180 / pi) * Delta_in_Rad;

        Iteration = Iteration + 1;

        %% Output

        %fprintf("YBus: \n")
        %Y_Bus

        %fprintf("Number of Iteration: \n");
        %Iteration

        %fprintf("Voltage Magnitude: \n")
        %V

        %fprintf("Voltage Angles in Degree: \n")
        %Delta_in_Degree

        %fprintf("Real Power in MW: \n")
        %P_Calculated * Base_MVA

        %fprintf("Reactive Power in MVAR: \n")
        %Q_Calculated * Base_MVA

        % Checking Iteration Limit
        if (Iteration == 4)
            break;
        end

end

% This does not converge; this is just to get a certain iterated values which are used as a input
of Fast Decoupled Function to make that converge.
```

Function - Fast Decoupled:

```
%% Power Flow Function: Fast Decoupled

function [V, Delta_in_Rad, Iteration] = Fast_Decoupled_Function(Y_Bus, V_Desired, Delta_in_Rad,
P_Gen, P_Load, Q_Gen, Q_Load, All_Bus_Number, All_Bus_Type)

%% Basic Initialization
Total_Bus = length(All_Bus_Number);
PQ_Bus_Type = 0;
PQ_Bus_Type_1 = 1;
PV_Bus_Type = 2;
Slack_Bus_Type = 3;
Bus = All_Bus_Number(find(All_Bus_Type ~= Slack_Bus_Type)); % Bus Type Data Except the Slack Bus
Bus_Type = All_Bus_Type(Bus); % Bus Type Data Except the Slack Bus
Base_MVA = 100;

%% Power Flow

% Schedule Real and Reactive Power
P_Scheduled = transpose(P_Gen - P_Load);
Q_Scheduled = transpose(Q_Gen - Q_Load);

% Initial Voltage Magnitude
V = ones(1,length(All_Bus_Number));
V(1,find(V_Desired)) = V_Desired(find(V_Desired),1);

% Initialization
Iteration = 0;
Tolerance = 0.01;
while 1
    %% Calculating Real Power

    % Initialization
    P_Calculated = zeros(1,Total_Bus);

    % LOOP: Computing Real Power
    for i=1:Total_Bus
        for n=1:Total_Bus
            P_Calculated(i) = P_Calculated(i) + (abs(abs(Y_Bus(i,n)) * V(i) * V(n))) *
(cos(angle(Y_Bus(i,n)) + Delta_in_Rad(n) - Delta_in_Rad(i)));
        end
    end

    %% Calculating Reactive Power

    % Initialization
    Q_Calculated = zeros(1,Total_Bus);

    % LOOP: Computing Reactive Power
    for i=1:Total_Bus
        for n=1:Total_Bus
            Q_Calculated(i) = Q_Calculated(i) + (abs(abs(Y_Bus(i,n)) * V(i) * V(n))) *
(sin(angle(Y_Bus(i,n)) + Delta_in_Rad(n) - Delta_in_Rad(i)));
        end
        Q_Calculated(i) = - Q_Calculated(i);
    end

    %% Calculating Mismatch
```

```
    Delta_P = P_Scheduled - P_Calculated;
    Delta_Q = Q_Scheduled - Q_Calculated;

% Initialization of Jacobian in Fast Decoupled Method; J12=J21=0
        J11 = zeros(length(Bus));
        J22 = zeros(length(Bus));

        % LOOP: Computing Jacobian Matrix for All the Buses Except Slack Bus
        for i=1:length(Bus)
            for j=1:length(Bus)
                    J11(i,j) = - (imag(Y_Bus(Bus(i),Bus(j))));
                    J22(i,j) = - (imag(Y_Bus(Bus(i),Bus(j))));
            end
        end

        % Removing Rows and Columns from Jacobian for PV Bus
        PV = find(Bus_Type == PV_Bus_Type);
        J22(:,PV) = [];
        J22(PV,:) = [];

        % Delta
        Delta_J = Delta_in_Rad(find(All_Bus_Type ~= Slack_Bus_Type));
        V_J = V(find((All_Bus_Type == PQ_Bus_Type)));
        Delta_P_J = Delta_P(find(All_Bus_Type ~= Slack_Bus_Type));
        Delta_Q_J = Delta_Q(find((All_Bus_Type == PQ_Bus_Type)));

        %% Updating V and Delta through LU Factorization

        % Function Calling: LU Factorization Using Dolittle's Method
        [Delta_Corrected] = LU_Factorization_Dolittle_Function(J11,Delta_P_J);
        [V_Corrected] = LU_Factorization_Dolittle_Function(J22,Delta_Q_J);

        % Updating Voltages and Angles
        Delta_Updated = Delta_J + transpose(Delta_Corrected);
        V_Updated = V_J.*(1  + transpose(V_Corrected));

        % Preparing for Next Iteration
        V_i = (find((All_Bus_Type == PQ_Bus_Type)));
        Delta_i = find(All_Bus_Type ~= Slack_Bus_Type);

        for i=1:length(Delta_i)
            Delta_New(Delta_i(i)) = Delta_Updated(i);
        end

        for i=1:length(V_i)
            V_Desired(V_i(i)) = V_Updated(i);
        end

        V = transpose(V_Desired);
        Delta_in_Rad = Delta_New;
        Delta_in_Degree = (180 / pi) * Delta_in_Rad;

        Iteration = Iteration + 1;

        %% Output

        %fprintf("YBus: \n")
        %Y_Bus

        fprintf("Number of Iteration: \n")
        Iteration
```

```
        fprintf("Voltage Magnitude: \n")
        V

        fprintf("Voltage Angles in Degree: \n")
        Delta_in_Degree

        fprintf("Real Power in MW: \n")
        P_Calculated * Base_MVA

        fprintf("Reactive Power in MVAR: \n")
        Q_Calculated * Base_MVA

        % Checking Tolerance Limit
        if (max(abs(Delta_P_J)) < Tolerance & max(abs(Delta_Q_J)) < Tolerance)
            break;
        end

end

end
```

Function – LU Factorization:

```
%% LU Factorization Function: Dolittle's Algorithm

function [ X_Matrix ] = LU_Factorization_Dolittle_Function(A_Matrix,B_Matrix)

% Getting the Size of Input Matrix
Length_A = length(A_Matrix);

% Initializing The Lower and Upper Triangular Matrices
Lower_Triangular_Matrix = zeros(Length_A,Length_A);
Upper_Triangular_Matrix = zeros(Length_A,Length_A);

% LOOP: Assigning 1 into All Diagonal Elements of Lower Traingular Matrix
for j = 1:Length_A
    Lower_Triangular_Matrix(j,j) = 1;
end

% Computing 1st Row of Upper Traingular Matrix
Upper_Triangular_Matrix(1,:) = A_Matrix(1,:);

% Computing 1st Column of Lower Traingular Matrix
Lower_Triangular_Matrix(:,1) = A_Matrix(:,1)/Upper_Triangular_Matrix(1,1);

% LOOP: Computing All Other Rows and Column of Upper and Lower Traingular Matrix
for j = 2:Length_A
    for k = j:Length_A
        Upper_Triangular_Matrix(j,k) = A_Matrix(j,k) - Lower_Triangular_Matrix(j,1:j-1) *
Upper_Triangular_Matrix(1:j-1,k);
    end

    for l = j+1:Length_A
        Lower_Triangular_Matrix(l,j) = (A_Matrix(l,j) - Lower_Triangular_Matrix(l,1:j-1) *
Upper_Triangular_Matrix(1:j-1,j)) / Upper_Triangular_Matrix(j,j);
    end

end

% Output
% A_Matrix
% Lower_Triangular_Matrix
% Upper_Triangular_Matrix

% Verification
% A_Matrix - (Lower_Triangular_Matrix * Upper_Triangular_Matrix)


%% Forward Substitution

% Initialization of Y Matrix
Y_Matrix = zeros(Length_A,1);

% Computing First Value of Y Matrix
Y_Matrix(1) = B_Matrix(1) / Lower_Triangular_Matrix(1,1);

% LOOP: Computing Rest of the Entries of Y Matrix
for j = 2:Length_A
    Y_Matrix(j) = (B_Matrix(j) - Lower_Triangular_Matrix(j,1:j-1) * Y_Matrix(1:j-1)) /
Lower_Triangular_Matrix(j,j);
```

```
end

% Output
% Y_Matrix


%% Backward Substitution

% Initialization of X Matrix
X_Matrix = zeros(Length_A,1);

% Computing Last Value of X Matrix
X_Matrix(Length_A) = Y_Matrix(Length_A) / Upper_Triangular_Matrix(Length_A,Length_A);

% LOOP: Computing Rest of the Entries of X Matrix
for j = Length_A-1:-1:1
    X_Matrix(j) = (Y_Matrix(j) - Upper_Triangular_Matrix(j,j+1:Length_A) * X_Matrix(j+1:Length_A))
/ Upper_Triangular_Matrix(j,j);
end

% Output
% X_Matrix

end
```

Results:

Problem A:

Newton-Raphson Method

Number of Iteration:

Iteration =

1

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0401 1.0376 1.0455 1.0651 1.0608 1.0462 1.0447

Voltage Angles in Degree:

Delta_in_Degree =

0 -7.9641 -7.9641 -7.9641 -7.9641 -7.9641 -7.9641 -7.9641 -7.9641 -7.9641 -7.9641

Real Power in MW:

ans =

1.0e-11 *

0.0767 0.0744 0.0767 0.0744 0.0577 -0.1599 -0.2109 -0.0133 -0.2842 -0.1599 0.0577

Reactive Power in MVAR:

ans =

185.0299 60.4790 185.0299 60.4790 -181.8282 -62.9427 -220.1250 -38.5000 -370.1250 -62.9427 -181.8282

Number of Iteration:

Iteration =

2

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0359 1.0349 1.0424 1.0612 1.0574 1.0438 1.0427

Voltage Angles in Degree:

Delta_in_Degree =

0 -7.3821 -17.2845 -26.8624 -5.5302 -13.7736 -20.8314 -30.8859 -40.6743 -33.2014 -23.6779

Real Power in MW:

ans =

888.8507 0.0000 0.0000 0.0000 -887.7857 -9.1911 4.4969 4.5886 14.5387 -14.5481 -0.6001

Reactive Power in MVAR:

ans =

-0.6797 -166.6613 -90.7605 -218.7274 133.3290 76.0008 -195.6295 2.2105 -271.1066 77.7264 83.6692

Number of Iteration:

Iteration =

3

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0096 0.9847 0.9711 0.9641 0.9813 0.9897 1.0111

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.4562 -24.9306 -35.0354 -6.3806 -16.1891 -24.3288 -37.2777 -49.9540 -41.7316 -31.5442

Real Power in MW:

ans =

1.0e+03 *

0.6157 0.6968 0.7161 0.6970 -0.0022 0.0114 -0.9495 0.0061 -1.7389 0.0128 0.0025

Reactive Power in MVAR:

ans =

-6.4035 -111.9341 -38.0511 -165.7143 50.4980 164.1559 10.8884 59.9932 20.6614 191.3774 100.0547

Number of Iteration:

Iteration =

4

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0065 0.9782 0.9613 0.9491 0.9716 0.9836 1.0083

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.7581 -26.9988 -37.1876 -6.4733 -16.5527 -24.9547 -38.7910 -52.3550 -43.9452 -33.6381

Real Power in MW:

ans =

1.0e+03 *

0.6920 0.6982 0.7183 0.6980 -0.0056 -0.0096 -0.9608 0.0003 -1.7401 -0.0077 -0.0025

Reactive Power in MVAR:

ans =

164.3868 194.2299 157.9330 163.5236 3.1602 9.8272 -91.3397 7.0366 -92.4274 9.9196 3.0792

Number of Iteration:

Iteration =

5

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0064 0.9780 0.9609 0.9485 0.9713 0.9834 1.0082

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.7682 -27.0919 -37.2828 -6.4765 -16.5647 -24.9765 -38.8539 -52.4584 -44.0422 -33.7318

Real Power in MW:

ans =

1.0e+03 *

0.6998 0.7000 0.7190 0.7000 -0.0001 -0.0004 -0.9670 0.0000 -1.7659 -0.0003 -0.0001

Reactive Power in MVAR:

ans =

184.6540 233.5748 175.7289 201.1635 0.0528 0.1733 -99.7298 0.3787 -99.8528 0.1037 0.0138

Number of Iteration:

Iteration =

6

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0064 0.9780 0.9609 0.9485 0.9713 0.9834 1.0082

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.7682 -27.0921 -37.2830 -6.4765 -16.5647 -24.9765 -38.8540 -52.4586 -44.0423 -33.7320

Real Power in MW:

ans =

1.0e+03 *

0.7001 0.7000 0.7190 0.7000 -0.0000 -0.0000 -0.9670 0.0000 -1.7670 -0.0000 -0.0000

Reactive Power in MVAR:

ans =

185.3269 234.9460 176.2398 202.3625 0.0001 0.0002 -99.9995 0.0009 -99.9998 0.0001 0.0000

Problem A:

Fast Decoupled Method

As Fast Decoupled method was not converging with initial condition, first I have run Newton-Raphson and took the 4th iteration results as the input of the Fast Decoupled function.

Number of Iteration:

Iteration =

1

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0064 0.9782 0.9611 0.9489 0.9715 0.9835 1.0082

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.7926 -27.0332 -37.2220 -6.5077 -16.5871 -24.9892 -38.8254 -52.3895 -43.9796 -33.6725

Real Power in MW:

ans =

1.0e+03 *

0.6998 0.7000 0.7190 0.7000 -0.0001 -0.0004 -0.9670 0.0000 -1.7659 -0.0003 -0.0001

Reactive Power in MVAR:

ans =

184.6540 233.5748 175.7289 201.1635 0.0528 0.1733 -99.7298 0.3787 -99.8528 0.1037 0.0138

Number of Iteration:

Iteration =

2

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0064 0.9782 0.9611 0.9489 0.9715 0.9835 1.0082

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.7708 -27.0930 -37.2829 -6.4770 -16.5663 -24.9761 -38.8520 -52.4548 -44.0413 -33.7326

Real Power in MW:

ans =

1.0e+03 *

0.7035 0.6999 0.7190 0.6999 -0.0039 -0.0006 -0.9669 0.0000 -1.7656 -0.0004 -0.0001

Reactive Power in MVAR:

ans =

185.3031 234.0405 175.8874 201.5237 0.4127 -0.0095 -99.9792 0.0324 -100.0168 -0.0057 -0.0058

Number of Iteration:

Iteration =

3

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0064 0.9780 0.9609 0.9485 0.9713 0.9834 1.0082

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.7579 -27.0413 -37.2313 -6.4723 -16.5535 -24.9603 -38.8189 -52.4033 -43.9897 -33.6809

Real Power in MW:

ans =

1.0e+03 *

0.7002 0.7000 0.7190 0.7000 0.0001 -0.0000 -0.9669 0.0001 -1.7672 -0.0000 -0.0000

Reactive Power in MVAR:

ans =

185.2359 234.1776 175.8904 201.5415 -0.3317 0.2503 -99.5279 0.5382 -99.4844 0.0685 0.0121

Problem B:

Newton-Raphson Method

Number of Iteration:

Iteration =

1

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0529 1.0687 1.1025 1.1085 1.0853 1.0596 1.0502

Voltage Angles in Degree:

Delta_in_Degree =

0 -8.0698 -8.0698 -8.0698 -8.0698 -8.0698 -8.0698 -8.0698 -8.0698 -8.0698 -8.0698

Real Power in MW:

ans =

1.0e-11 *

0.0767 0.0744 0.0767 0.0744 0.0577 -0.1599 -0.2842 -0.0155 -0.2132 -0.1599 0.0577

Reactive Power in MVAR:

ans =

185.0299 60.4790 185.0299 60.4790 -181.8282 -62.9427 -429.7500 -48.1250 -420.1250 -62.9427 -181.8282

Number of Iteration:

Iteration =

2

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0469 1.0624 1.0930 1.0989 1.0786 1.0553 1.0472

Voltage Angles in Degree:

Delta_in_Degree =

```
        0   -7.2930  -12.3140  -21.8067   -5.4874  -13.4981  -20.1224  -26.2115  -35.3291  -28.0655  -18.6741

Real Power in MW:

ans =

  911.5820    0.0000    0.0000    0.0000 -918.1976  -28.9979   35.0530    6.4278   23.1118  -23.0541   -3.9005

Reactive Power in MVAR:

ans =

  -76.6819 -355.1600 -124.4594 -299.7196  139.8339   82.3335 -171.8105    5.1446 -263.7234   80.4464   85.4798

Number of Iteration:

Iteration =

     3

Voltage Magnitude:

V =

    1.0300    1.0100    1.0300    1.0100    1.0226    1.0164    1.0282    1.0188    1.0094    1.0053    1.0175

Voltage Angles in Degree:

Delta_in_Degree =

        0   -9.1288  -17.8981  -27.8995   -6.2424  -15.6529  -23.2235  -30.8826  -42.4449  -34.4940  -24.4710

Real Power in MW:

ans =

    1.0e+03 *

    0.6175    0.6945    0.7155    0.6958   -0.0057    0.0041   -0.9416    0.0047   -1.7302    0.0088    0.0012

Reactive Power in MVAR:

ans =

  -74.7544 -279.0003  -66.4655 -235.6717   50.8425  158.7622   -6.3054   45.2724   14.9952  188.2070   99.2678

Number of Iteration:

Iteration =

     4
```

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0203 1.0119 1.0214 1.0097 1.0026 1.0008 1.0153

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.3387 -18.9954 -29.0640 -6.3057 -15.9065 -23.6544 -31.6656 -43.7938 -35.7046 -25.5884

Real Power in MW:

ans =

1.0e+03 *

0.6858 0.6985 0.7183 0.6982 -0.0047 -0.0080 -0.9604 -0.0002 -1.7450 -0.0066 -0.0022

Reactive Power in MVAR:

ans =

83.1321 0.9693 118.6110 68.6800 2.7488 8.5346 -93.6781 4.4650 -93.2986 8.9904 2.8879

Number of Iteration:

Iteration =

5

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0203 1.0118 1.0213 1.0095 1.0025 1.0008 1.0153

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.3425 -19.0218 -29.0914 -6.3069 -15.9110 -23.6625 -31.6823 -43.8245 -35.7327 -25.6150

Real Power in MW:

ans =

1.0e+03 *

0.6912 0.7000 0.7190 0.7000 -0.0001 -0.0002 -0.9669 -0.0000 -1.7665 -0.0002 -0.0000

Reactive Power in MVAR:

```
ans =

  97.6241   28.7114  131.9440   95.9812    0.0285    0.0960  -99.8913    0.1405  -99.9105    0.0656    0.0096

>>
```

Problem B:

Fast Decoupled Method

As Fast Decoupled method was not converging with initial condition, first I have run Newton-Raphson and took the 4th iteration results as the input of the Fast Decoupled function.

Number of Iteration:

Iteration =

1

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0203 1.0119 1.0214 1.0096 1.0026 1.0008 1.0153

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.3588 -19.0154 -29.0841 -6.3257 -15.9266 -23.6745 -31.6857 -43.8139 -35.7247 -25.6084

Real Power in MW:

ans =

1.0e+03 *

0.6912 0.7000 0.7190 0.7000 -0.0001 -0.0002 -0.9669 -0.0000 -1.7665 -0.0002 -0.0000

Reactive Power in MVAR:

ans =

97.6241 28.7114 131.9440 95.9812 0.0285 0.0960 -99.8913 0.1405 -99.9105 0.0656 0.0096

Problem C :

Newton-Raphson Method

Number of Iteration:

Iteration =

1

Voltage Magnitude:

V =

1.0300 1.0100 1.0305 1.0105 1.0203 1.0119 1.0214 1.0098 1.0031 1.0013 1.0158

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.3407 -19.0201 -29.0897 -6.3051 -15.9093 -23.6608 -31.6806 -43.8228 -35.7310 -25.6133

Real Power in MW:

ans =

1.0e+03 *

0.6913 0.7000 0.7190 0.7000 -0.0000 0.0000 -0.9670 -0.0000 -1.7671 0.0000 0.0000

Reactive Power in MVAR:

ans =

97.9126 29.2820 132.0794 96.2569 -0.0152 -0.2815 -99.7948 -0.0139 -100.1817 0.4484 -0.0194

>>

Fast Decoupled Method

Here, it is not necessary to run Newton-Raphson first. Fast decoupled is converged with direct input.

Number of Iteration:

Iteration =

1

Voltage Magnitude:

V =

1.0300 1.0081 1.0251 1.0052 1.0196 1.0100 1.0191 1.0063 0.9978 0.9961 1.0105

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.3428 -19.0221 -29.0917 -6.3071 -15.9113 -23.6628 -31.6826 -43.8248 -35.7330 -25.6153

Real Power in MW:

ans =

1.0e+03 *

0.6913 0.7000 0.7190 0.7000 -0.0000 0.0000 -0.9670 -0.0000 -1.7671 0.0000 0.0000

Reactive Power in MVAR:

ans =

97.9126 29.2820 132.0794 96.2569 -0.0152 -0.2815 -99.7948 -0.0139 -100.1817 0.4484 -0.0194

>>

Input Data:

Problem A:

BUS DATA FOLLOWS

BUS				LOAD FLOW AREA	LOSS ZONE	TYPE	V_MAG	V_ANG in RAD	LOAD_MW	LOAD_MVA	G_MW	G_MVAR	BASE_KV	V_DESIRED	MAX MVAR/VOLT LIMIT	MIN MVAR/VOLT LIMIT	SHUNT_G	SHUNT_B	REMOTE CONTROLLED BUS
1	Bus	1	HV	1	1	3	1.06	0	0	0	700	185	0	1.03	0	0	0	0	0
2	Bus	2	HV	1	1	2	1.045	0	0	0	700	235	0	1.01	0	0	0	0	0
3	Bus	3	HV	1	1	2	1.01	0	0	0	719	176	0	1.03	0	0	0	0	0
4	Bus	4	HV	1	1	2	1.019	0	0	0	700	202	0	1.01	0	0	0	0	0
5	Bus	5	HV	1	1	0	1.02	0	0	0	0	0	0	1	0	0	0	0	0
6	Bus	6	LV	1	1	0	1.07	0	0	0	0	0	0	1	0	0	0	0	0
7	Bus	7	ZV	1	1	0	1.062	0	967	100	0	0	0	1	0	0	0	2	0
8	Bus	8	TV	1	1	0	1.09	0	0	0	0	0	0	1	0	0	0	0	0
9	Bus	9	LV	1	1	0	1.056	0	1767	100	0	0	0	1	0	0	0	3.5	0
10	Bus	10	LV	1	1	0	1.051	0	0	0	0	0	0	1	0	0	0	0	0
11	Bus	11	LV	1	1	0	1.057	0	0	0	0	0	0	1	0	0	0	0	0

BRANCH DATA
FOLLOWS

FROM	TO	LOAD FLOW AREA	LOSS ZONE	CIRCUIT	TYPE	R	X	B	LINE MVA RATING 1	LINE MVA RATING 2	LINE MVA RATING 3	CONTROL BUS	SIDE	XFR FINAL TURN RATIO	XFR FINAL ANGLE	MIN TAP.PHASE SHIFT	MAX TAP/PHASE SHIFT	STEP SIZE	MIN VOLT/MVAR MW LIMIT	MAX VOLT/MVAR MW LIMIT
1	5	1	1	1	0	0	0.0167	0	0	0	0	0	0	1	0	0	0	0	0	0
2	6	1	1	1	0	0	0.0167	0	0	0	0	0	0	1	0	0	0	0	0	0
3	11	1	1	1	0	0	0.0167	0	0	0	0	0	0	1	0	0	0	0	0	0
4	10	1	1	1	0	0	0.0167	0	0	0	0	0	0	1	0	0	0	0	0	0
5	6	1	1	1	0	0.0025	0.025	0.04375	0	0	0	0	0	0	0	0	0	0	0	0
6	7	1	1	1	0	0.001	0.01	0.0175	0	0	0	0	0	0	0	0	0	0	0	0
7	8	1	1	1	0	0.0055	0.055	0.385	0	0	0	0	0	0	0	0	0	0	0	0
8	9	1	1	1	0	0.0055	0.055	0.385	0	0	0	0	0	0	0	0	0	0	0	0
9	10	1	1	1	0	0.001	0.01	0.0175	0	0	0	0	0	0	0	0	0	0	0	0
10	11	1	1	1	0	0.0025	0.025	0.04375	0	0	0	0	0	0	0	0	0	0	0	0

Problem B:

BUS DATA FOLLOWS

BUS				LOAD FLOW AREA	LOSS ZONE	TYPE	V_MAG	V_ANG in RAD	LOAD_MW	LOAD_MVA	G_MW	G_MVAR	BASE_KV	V_DESIRED	MAX MVAR/VOLT LIMIT	MIN MVAR/VOLT LIMIT	SHUNT_G	SHUNT_B	REMOTE CONTROLLED BUS
1	Bus	1	HV	1	1	3	1.06	0	0	0	700	185	0	1.03	0	0	0	0	0
2	Bus	2	HV	1	1	2	1.045	0	0	0	700	235	0	1.01	0	0	0	0	0
3	Bus	3	HV	1	1	2	1.01	0	0	0	719	176	0	1.03	0	0	0	0	0
4	Bus	4	HV	1	1	2	1.019	0	0	0	700	202	0	1.01	0	0	0	0	0
5	Bus	5	HV	1	1	0	1.02	0	0	0	0	0	0	1	0	0	0	0	0
6	Bus	6	LV	1	1	0	1.07	0	0	0	0	0	0	1	0	0	0	0	0
7	Bus	7	ZV	1	1	0	1.062	0	967	100	0	0	0	1	0	0	0	4	0
8	Bus	8	TV	1	1	0	1.09	0	0	0	0	0	0	1	0	0	0	0	0
9	Bus	9	LV	1	1	0	1.056	0	1767	100	0	0	0	1	0	0	0	4	0
10	Bus	10	LV	1	1	0	1.051	0	0	0	0	0	0	1	0	0	0	0	0
11	Bus	11	LV	1	1	0	1.057	0	0	0	0	0	0	1	0	0	0	0	0

BRANCH DATA
FOLLOWS

FROM	TO	LOAD FLOW AREA	LOSS ZONE	CIRCUIT	TYPE	R	X	B	LINE MVA RATING 1	LINE MVA RATING 2	LINE MVA RATING 3	CONTROL BUS	SIDE	XFR FINAL TURN RATIO	XFR FINAL ANGLE	MIN TAP.PHASE SHIFT	MAX TAP/PHASE SHIFT	STEP SIZE	MIN VOLT/MVAR MW LIMIT	MAX VOLT/MVAR MW LIMIT
1	5	1	1	1	0	0	0.0167	0	0	0	0	0	0	1	0	0	0	0	0	0
2	6	1	1	1	0	0	0.0167	0	0	0	0	0	0	1	0	0	0	0	0	0
3	11	1	1	1	0	0	0.0167	0	0	0	0	0	0	1	0	0	0	0	0	0
4	10	1	1	1	0	0	0.0167	0	0	0	0	0	0	1	0	0	0	0	0	0
5	6	1	1	1	0	0.0025	0.025	0.04375	0	0	0	0	0	0	0	0	0	0	0	0
6	7	1	1	1	0	0.001	0.01	0.0175	0	0	0	0	0	0	0	0	0	0	0	0
7	8	1	1	1	0	0.00367	0.03667	0.57755	0	0	0	0	0	0	0	0	0	0	0	0
8	9	1	1	1	0	0.0055	0.055	0.385	0	0	0	0	0	0	0	0	0	0	0	0
9	10	1	1	1	0	0.001	0.01	0.0175	0	0	0	0	0	0	0	0	0	0	0	0
10	11	1	1	1	0	0.0025	0.025	0.04375	0	0	0	0	0	0	0	0	0	0	0	0

Problem C:

Problem B:

BUS DATA FOLLOWS

BUS				LOAD FLOW AREA	LOSS ZONE	TYPE	V_MAG	V_ANG in RAD	LOAD_MW	LOAD_MVA	G_MW	G_MVAR	BASE_KV	V_DESIRED	MAX MVAR/VOLT LIMIT	MIN MVAR/VOLT LIMIT	SHUNT_G	SHUNT_B	REMOTE CONTROLLED BUS
1	Bus	1	HV	1	1	3	1.06	20.2	0	0	700	185	0	1.03	0	0	0	0	0
2	Bus	2	HV	1	1	2	1.045	0	-700	-28.37	0	0	0	1.01	0	0	0	0	0
3	Bus	3	HV	1	1	2	1.01	0	-719	-131.883	0	0	0	1.03	0	0	0	0	0
4	Bus	4	HV	1	1	2	1.019	0	-700	-95.8365	0	0	0	1.01	0	0	0	0	0
5	Bus	5	HV	1	1	0	1.02	0	0	0	0	0	0	1	0	0	0	0	0
6	Bus	6	LV	1	1	0	1.07	0	0	0	0	0	0	1	0	0	0	0	0
7	Bus	7	ZV	1	1	0	1.062	0	967	100	0	0	0	1	0	0	0	4	0
8	Bus	8	TV	1	1	0	1.09	0	0	0	0	0	0	1	0	0	0	0	0
9	Bus	9	LV	1	1	0	1.056	0	1767	100	0	0	0	1	0	0	0	4	0
10	Bus	10	LV	1	1	0	1.051	0	0	0	0	0	0	1	0	0	0	0	0
11	Bus	11	LV	1	1	0	1.057	0	0	0	0	0	0	1	0	0	0	0	0

BRANCH DATA
FOLLOWS

FROM	TO	LOAD FLOW AREA	LOSS ZONE	CIRCUIT	TYPE	R	X	B	LINE MVA RATING 1	LINE MVA RATING 2	LINE MVA RATING 3	CONTROL BUS	SIDE	XFR FINAL TURN RATIO	XFR FINAL ANGLE	MIN TAP.PHASE SHIFT	MAX TAP/PHASE SHIFT	STEP SIZE	MIN VOLT/MVAR MW LIMIT	MAX VOLT/MVAR MW LIMIT
1	5	1	1	1	0	0	0.0167	0	0	0	0	0	0	1	0	0	0	0	0	0
2	6	1	1	1	0	0	0.0167	0	0	0	0	0	0	1	0	0	0	0	0	0
3	11	1	1	1	0	0	0.0167	0	0	0	0	0	0	1	0	0	0	0	0	0
4	10	1	1	1	0	0	0.0167	0	0	0	0	0	0	1	0	0	0	0	0	0
5	6	1	1	1	0	0.0025	0.025	0.04375	0	0	0	0	0	0	0	0	0	0	0	0
6	7	1	1	1	0	0.001	0.01	0.0175	0	0	0	0	0	0	0	0	0	0	0	0
7	8	1	1	1	0	0.00367	0.0367	0.5775	0	0	0	0	0	0	0	0	0	0	0	0
8	9	1	1	1	0	0.0055	0.055	0.385	0	0	0	0	0	0	0	0	0	0	0	0
9	10	1	1	1	0	0.001	0.01	0.0175	0	0	0	0	0	0	0	0	0	0	0	0
10	11	1	1	1	0	0.0025	0.025	0.04375	0	0	0	0	0	0	0	0	0	0	0	0