EE 523 | Spring 2023

Assignment 01

Submitted by

Sajjad Uddin Mahmud

WSU ID: 011789534

WASHINGTON STATE
UNIVERSITY

Code:

Platform: MATLAB

Main Code :

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

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

%% Bus Shunt Conductance and Shunt Susceptance
G_Shunt_Bus = xlsread('Kundur_Two_Area_System.xlsx',Excel_Worksheet,'R4:R14');
B_Shunt_Bus = xlsread('Kundur_Two_Area_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_Two_Area_System.xlsx',Excel_Worksheet,'G18:G27');

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

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

%% Transformer Turns Ratio
XFR_TurnRatio = xlsread('Kundur_Two_Area_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 1.0649 1.0473 1.0550 1.0709 1.0627 1.0472 1.0451

Voltage Angles in Degree:

Delta_in_Degree =

0 14.1978 14.1978 14.1978 14.1978 14.1978 14.1978 14.1978 14.1978 14.1978 14.1978

Real Power in MW:

ans =

1.0e+03 *

2.1297 0.0000 0.0000 0.0000 -2.1297 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 0.0000

Reactive Power in MVAR:

ans =

564.3918 60.4790 185.0299 60.4790 197.5336 -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 1.0786 1.0514 1.0584 1.0708 1.0604 1.0454 1.0433

Voltage Angles in Degree:

Delta_in_Degree =

0 -7.5255 -16.9800 -26.5538 -6.0226 -13.8575 -20.7846 -30.6625 -40.3464 -32.8865 -23.3708

Real Power in MW:

ans =

1.0e+03 *

-1.6110 0.0000 0.0000 0.0000 1.6184 -0.0153 0.0050 0.0047 0.0147 -0.0152 -0.0009

Reactive Power in MVAR:

ans =

-14.9118 -225.6061 -93.3520 -224.9557 495.4177 77.9557 -195.2449 2.4574 -270.8618 77.9340 83.8079

Number of Iteration:

Iteration =

3

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0130 0.9869 0.9737 0.9662 0.9822 0.9902 1.0113

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.2934 -24.4828 -34.5837 -6.3573 -16.0038 -24.0732 -36.8897 -49.4874 -41.2760 -31.0950

Real Power in MW:

ans =

1.0e+03 *

0.6980 0.7013 0.7163 0.6974 -0.0699 0.0089 -0.9629 0.0042 -1.7429 0.0141 0.0029

Reactive Power in MVAR:

ans =

-263.0013 -211.7455 -42.0610 -175.5967 444.7931 161.3176 11.3574 59.5464 20.8568 191.2985 100.0204

Number of Iteration:

Iteration =

4

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0065 0.9783 0.9614 0.9493 0.9717 0.9836 1.0083

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.7515 -26.9668 -37.1552 -6.4717 -16.5450 -24.9426 -38.7659 -52.3211 -43.9124 -33.6059

Real Power in MW:

ans =

1.0e+03 *

0.6918 0.6974 0.7182 0.6979 -0.0123 -0.0077 -0.9575 0.0012 -1.7389 -0.0079 -0.0025

Reactive Power in MVAR:

ans =

143.4928 180.8507 156.6653 160.5378 27.2058 10.9940 -90.4928 7.0786 -92.3724 10.0843 3.1147

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 -27.0918 -37.2827 -6.4765 -16.5647 -24.9764 -38.8538 -52.4583 -44.0420 -33.7317

Real Power in MW:

ans =

```
   1.0e+03 *

    0.6997    0.6999    0.7190    0.7000   -0.0003   -0.0005   -0.9668    0.0001   -1.7657   -0.0004   -0.0001

Reactive Power in MVAR:

ans =

  184.2368  232.9942  175.6173  200.8981    0.2747    0.2809  -99.6393    0.4408  -99.8420    0.1133    0.0149

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 =

  185.3260  234.9443  176.2393  202.3613    0.0002    0.0004  -99.9991    0.0013  -99.9997    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 1.0065 0.9782 0.9612 0.9490 0.9716 0.9835 1.0083

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.8011 -27.0164 -37.2048 -6.5214 -16.5946 -24.9922 -38.8156 -52.3707 -43.9621 -33.6556

Real Power in MW:

ans =

1.0e+03 *

0.6997 0.6999 0.7190 0.7000 -0.0003 -0.0005 -0.9668 0.0001 -1.7657 -0.0004 -0.0001

Reactive Power in MVAR:

ans =

184.2368 232.9942 175.6173 200.8981 0.2747 0.2809 -99.6393 0.4408 -99.8420 0.1133 0.0149

Number of Iteration:

Iteration =

2

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0064 0.9782 0.9612 0.9490 0.9716 0.9835 1.0083

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.7698 -27.0880 -37.2776 -6.4764 -16.5648 -24.9734 -38.8471 -52.4486 -44.0358 -33.7274

Real Power in MW:

ans =

1.0e+03 *

0.7050 0.6998 0.7190 0.6999 -0.0057 -0.0007 -0.9666 0.0001 -1.7653 -0.0005 -0.0001

Reactive Power in MVAR:

ans =

185.2990 233.6724 175.8017 201.3188 0.5927 -0.0116 -99.9719 0.0358 -100.0220 -0.0069 -0.0068

Number of Iteration:

Iteration =

3

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0064 0.9781 0.9610 0.9486 0.9713 0.9834 1.0082

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.7572 -27.0336 -37.2232 -6.4720 -16.5525 -24.9582 -38.8134 -52.3945 -43.9814 -33.6730

Real Power in MW:

ans =

1.0e+03 *

0.7001 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 =

185.1895 233.8677 175.8054 201.3400 -0.4829 0.3788 -99.3538 0.6714 -99.3918 0.0781 0.0138

Problem B:

Newton-Raphson Method

Number of Iteration:

Iteration =

1

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0785 1.0791 1.1129 1.1159 1.0877 1.0609 1.0507

Voltage Angles in Degree:

Delta_in_Degree =

0 14.3786 14.3786 14.3786 14.3786 14.3786 14.3786 14.3786 14.3786 14.3786 14.3786

Real Power in MW:

ans =

1.0e+03 *

2.1297 0.0000 0.0000 0.0000 -2.1297 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 0.0000

Reactive Power in MVAR:

ans =

564.3918 60.4790 185.0299 60.4790 197.5336 -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 1.0902 1.0792 1.1094 1.1103 1.0822 1.0572 1.0480

Voltage Angles in Degree:

Delta_in_Degree =

0 -7.4147 -12.0427 -21.5297 -5.9529 -13.5601 -20.0598 -26.0288 -35.0276 -27.7806 -18.3995

Real Power in MW:

ans =

1.0e+03 *

-1.6518 0.0000 0.0000 0.0000 1.6515 -0.0359 0.0364 0.0065 0.0234 -0.0239 -0.0042

Reactive Power in MVAR:

ans =

-90.7027 -418.0637 -127.8255 -307.8099 516.2356 84.4993 -168.5341 5.5693 -263.2220 80.7205 85.6615

Number of Iteration:

Iteration =

3

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0258 1.0185 1.0306 1.0208 1.0103 1.0058 1.0177

Voltage Angles in Degree:

Delta_in_Degree =

0 -8.9805 -17.5408 -27.5382 -6.2213 -15.4844 -22.9949 -30.5796 -42.0688 -34.1288 -24.1122

Real Power in MW:

ans =

1.0e+03 *

0.6974 0.6987 0.7157 0.6962 -0.0717 0.0010 -0.9539 0.0031 -1.7343 0.0102 0.0016

Reactive Power in MVAR:

ans =

-335.1018 -380.9067 -71.2751 -247.5230 453.6693 155.9044 -6.0536 44.8067 15.1052 188.0764 99.2187

Number of Iteration:

Iteration =

4

Voltage Magnitude:

V =

1.0300 1.0100 1.0300 1.0100 1.0204 1.0119 1.0215 1.0098 1.0027 1.0009 1.0153

Voltage Angles in Degree:

Delta_in_Degree =

0 -9.3345 -18.9802 -29.0487 -6.3048 -15.9017 -23.6471 -31.6541 -43.7776 -35.6891 -25.5732

Real Power in MW:

ans =

1.0e+03 *

0.6857 0.6977 0.7183 0.6982 -0.0112 -0.0063 -0.9572 0.0006 -1.7435 -0.0069 -0.0023

Reactive Power in MVAR:

ans =

62.8989 -11.5286 117.3564 65.7442 26.7709 9.7786 -92.9328 4.6109 -93.2053 9.2082 2.9352

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 -29.0913 -6.3069 -15.9110 -23.6625 -31.6823 -43.8245 -35.7327 -25.6150

Real Power in MW:

ans =

1.0e+03 *

0.6911 0.7000 0.7190 0.7000 -0.0002 -0.0003 -0.9668 0.0000 -1.7663 -0.0002 -0.0000

Reactive Power in MVAR:

ans =

```
   97.3500   28.3703  131.8829   95.8365    0.2083    0.1747  -99.8397    0.1769  -99.8990    0.0743    0.0106
```

>>

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   -9.3685  -19.0143  -29.0827   -6.3388  -15.9358  -23.6811  -31.6881  -43.8116  -35.7231  -25.6072
```

Real Power in MW:

```
ans =

   1.0e+03 *

   0.6911    0.7000    0.7190    0.7000   -0.0002   -0.0003   -0.9668    0.0000   -1.7663   -0.0002   -0.0000
```

Reactive Power in MVAR:

```
ans =

   97.3500   28.3703  131.8829   95.8365    0.2083    0.1747  -99.8397    0.1769  -99.8990    0.0743    0.0106

>>
```

Problem C :

Newton-Raphson Method

Both Newton-Raphson and Fast Decoupled is not converging.

Input Data:

Problem A:

BUS DATA FOLLOWS

BUS				LOAD FLOW AREA	LOSS ZONE	TYPE	V_MAG	V_ANG	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	10.5	0	0	700	235	0	1.01	0	0	0	0	0
3	Bus	3	HV	1	1	2	1.01	-6.8	0	0	719	176	0	1.03	0	0	0	0	0
4	Bus	4	HV	1	1	2	1.019	-17	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	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	10.5	0	0	700	235	0	1.01	0	0	0	0	0
3	Bus	3	HV	1	1	2	1.01	-6.8	0	0	719	176	0	1.03	0	0	0	0	0
4	Bus	4	HV	1	1	2	1.019	-17	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	0.0367	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

Problem C:

Problem B:

BUS DATA FOLLOWS

BUS				LOAD FLOW AREA	LOSS ZONE	TYPE	V_MAG	V_ANG	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