Homework 1

Submitted by Athul Jose P
WSU ID 011867566

Executive Summary

Newton Raphson Method

Steps in main.m function

1. Initializing Kundur 2 area system and importing data
2. Making Ybus by calling y_bus_calc.m (with taps or without taps should be mentioned)
3. Calculating the scheduled active power (P) and reactive power (Q)
4. Finding bus types and assigning to vectors
5. Initializing Voltage magnitude and angles
6. Calling Newton Raphson Function (NR.m)
7. Calculating P & Q after convergence

Steps in y_bus_calc.m function

1. Initializing Ybus with zeros
2. Calculating diagonal and off diagonal elements
3. Changing terms if tap is present

Steps in NR.m function

1. Initializing indexes and deltas
2. Starting iteration loop which will terminate either if converged or 100 iterations
3. Calling dpdq.m for calculating mismatch
4. Calling J_calc.m for calculating Jacobian
5. Calling fwd_bwd.m for calculating the ΔV and Δδ
6. Updating del_V and del_T (magnitude and angle) for next iteration
7. Updating the error. Here the error is taken as the maximum of absolute of deltas (ΔP and ΔQ)

Steps in dpdq.m function

1. Initializing P & Q as zeros
2. Calculating P for PV bus and P & Q for PQ bus

Steps in J_calc.m function

1. Calculating J1 with loops according to limits (n_bus-1, n_bus-1)
2. Calculating J2 with loops according to limits (n_bus-1, n_pq)
3. Calculating J3 with loops according to limits (n_pq, n_bus-1)
4. Calculating J4 with loops according to limits (n_pq, n_pq)
5. Combining all to make J

Steps in fwd_bwd.m function

1. Calling LU.m for calculating Lower and Upper elements
2. Doing the backward substitution
3. Doing the forward substitution

Steps in LU.m function

1. Making the Q matrix
2. Dividing it into L & U matrices

Fast Decoupled Method

Steps in main.m function

1 Initializing Kundur 2 area system and importing data
2 Making Ybus by calling y_bus_calc.m (with taps or without taps should be mentioned)
3 Calculating the scheduled active power (P) and reactive power (Q)
4 Finding bus types and assigning to vectors
5 Initializing Voltage magnitude and angles
6 Calling Newton Raphson Function (FD.m)
7 Calculating P & Q after convergence

Steps in y_bus_calc.m function

1 Initializing Ybus with zeros
2 Calculating diagonal and off diagonal elements
3 Changing terms if tap is present

Steps in FD.m function

1. Initializing indexes, deltas and B
2. Starting iteration loop which will terminate either if converged or 100 iterations
3. Calling dpdq.m for calculating mismatch
4. Calling fwd_bwd.m for calculating the ΔV and Δδ
5. Updating del_V and del_T (magnitude and angle) for next iteration
6. Updating the error. Here the error is taken as the maximum of absolute of deltas (ΔP and ΔQ)

Steps in dpdq.m function

1 Initializing P & Q as zeros
2 Calculating P for PV bus and P & Q for PQ bus

Steps in fwd_bwd.m function

1 Calling LU.m for calculating Lower and Upper elements
2 Doing the backward substitution
3 Doing the forward substitution

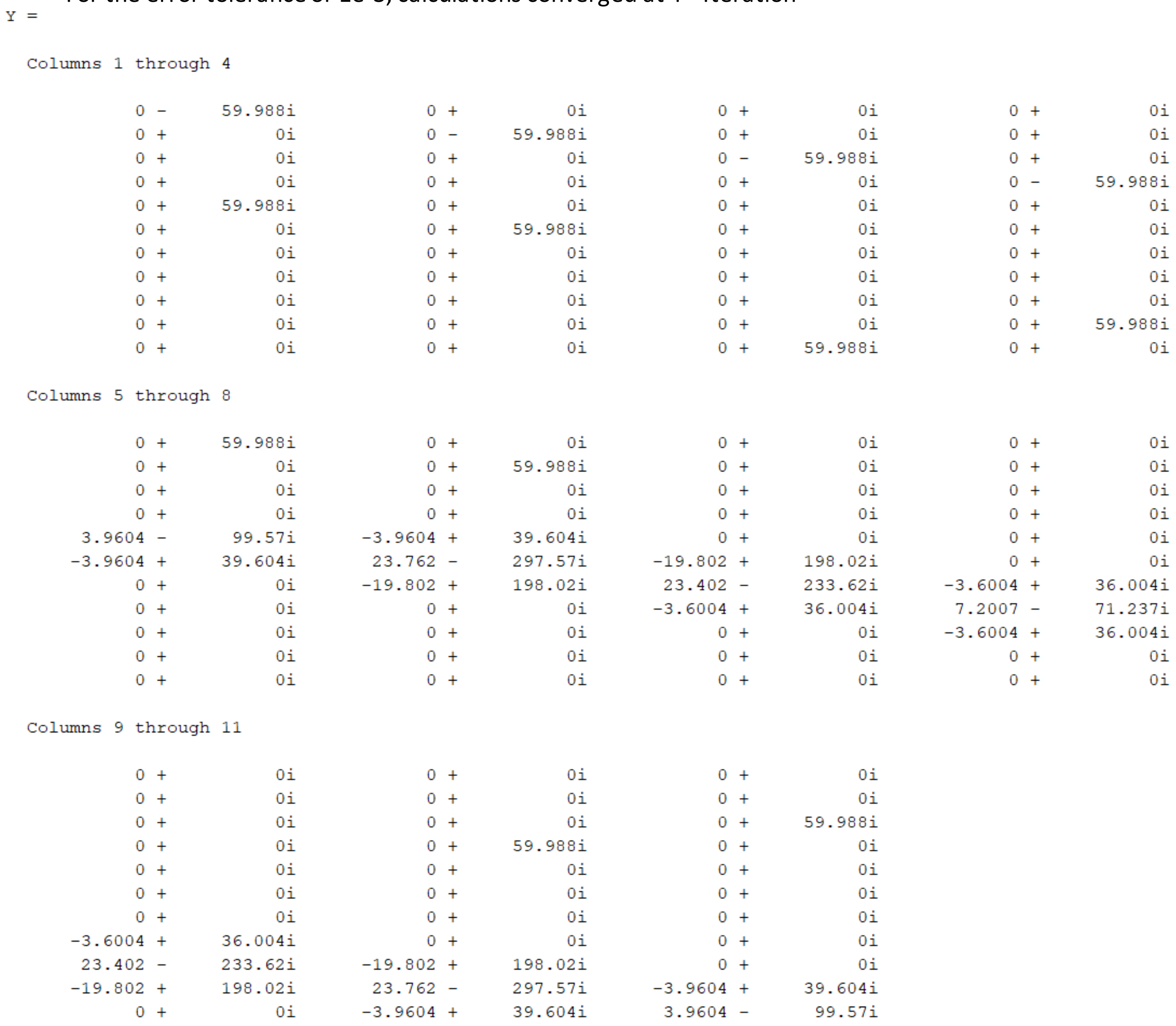
Steps in LU.m function

3. Making the Q matrix
4. Dividing it into L & U matrices

Results

Case 1a: Newton Raphson with Slack, PV, PQ buses

For the error tolerance of 1e-3, calculations converged at 4th Iteration



```
Y =

  Columns 1 through 4

           0 -     59.988i            0 +          0i            0 +          0i            0 +          0i
           0 +          0i            0 -     59.988i            0 +          0i            0 +          0i
           0 +          0i            0 +          0i            0 -     59.988i            0 +          0i
           0 +          0i            0 +          0i            0 +          0i            0 -     59.988i
           0 +     59.988i            0 +          0i            0 +          0i            0 +          0i
           0 +          0i            0 +     59.988i            0 +          0i            0 +          0i
           0 +          0i            0 +          0i            0 +          0i            0 +          0i
           0 +          0i            0 +          0i            0 +          0i            0 +          0i
           0 +          0i            0 +          0i            0 +          0i            0 +          0i
           0 +          0i            0 +          0i            0 +          0i            0 +     59.988i
           0 +          0i            0 +          0i            0 +     59.988i            0 +          0i

  Columns 5 through 8

           0 +     59.988i            0 +          0i            0 +          0i            0 +          0i
           0 +          0i            0 +     59.988i            0 +          0i            0 +          0i
           0 +          0i            0 +          0i            0 +          0i            0 +          0i
           0 +          0i            0 +          0i            0 +          0i            0 +          0i
      3.9604 -      99.57i      -3.9604 +     39.604i            0 +          0i            0 +          0i
     -3.9604 +     39.604i       23.762 -     297.57i      -19.802 +     198.02i            0 +          0i
           0 +          0i      -19.802 +     198.02i       23.402 -     233.62i      -3.6004 +     36.004i
           0 +          0i            0 +          0i      -3.6004 +     36.004i       7.2007 -     71.237i
           0 +          0i            0 +          0i            0 +          0i      -3.6004 +     36.004i
           0 +          0i            0 +          0i            0 +          0i            0 +          0i
           0 +          0i            0 +          0i            0 +          0i            0 +          0i

  Columns 9 through 11

           0 +          0i            0 +          0i            0 +          0i
           0 +          0i            0 +          0i            0 +          0i
           0 +          0i            0 +          0i            0 +     59.988i
           0 +          0i            0 +     59.988i            0 +          0i
           0 +          0i            0 +          0i            0 +          0i
           0 +          0i            0 +          0i            0 +          0i
           0 +          0i            0 +          0i            0 +          0i
     -3.6004 +     36.004i            0 +          0i            0 +          0i
      23.402 -     233.62i      -19.802 +     198.02i            0 +          0i
     -19.802 +     198.02i       23.762 -     297.57i      -3.9604 +     39.604i
           0 +          0i      -3.9604 +     39.604i       3.9604 -      99.57i
```

Final results

Bus Number	V(magnitude)	V(angle)
1	1.0300	0
2	1.0100	-8.7346
3	1.0300	-10.2582
4	1.0100	-20.2529
5	1.0193	-6.0570
6	1.0082	-15.3147
7	1.0087	-19.1595
8	1.0172	-25.0478
9	1.0146	-30.7797
10	1.0118	-26.8098
11	1.0198	-16.8103

Case 1b: Fast Decoupled with Slack, PV, PQ buses

For the error tolerance of 1e-3, calculations converged at 10th Iteration

Bus Number	V(magnitude)	V(angle)
1	1.0300	0
2	1.0100	-8.7345
3	1.0300	-10.2584
4	1.0100	-20.2532
5	1.0193	-6.0570
6	1.0082	-15.3148
7	1.0087	-19.1596
8	1.0172	-25.0481
9	1.0146	-30.7803
10	1.0118	-26.8103
11	1.0198	-16.8106

Case 2a: Newton Raphson with Slack & PQ buses

For the error tolerance of 1e-3, calculations converged at 9th Iteration

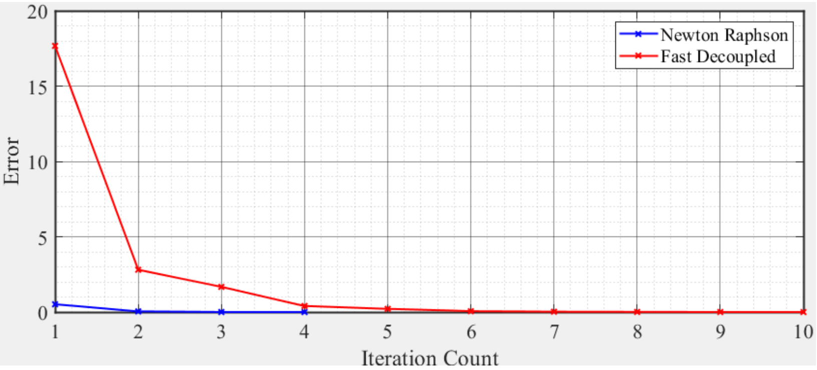
Bus Number	V(magnitude)	V(angle)
1	1.03	0
2	1.0541	-8.6594
3	1.1263	-11.364
4	1.1025	-19.681
5	1.0371	-5.907
6	1.0519	-14.7
7	1.0568	-18.23
8	1.0872	-23.551
9	1.1032	-28.527
10	1.1032	-25.186
11	1.116	-16.835

Case 2b: Fast Decoupled with Slack & PQ buses

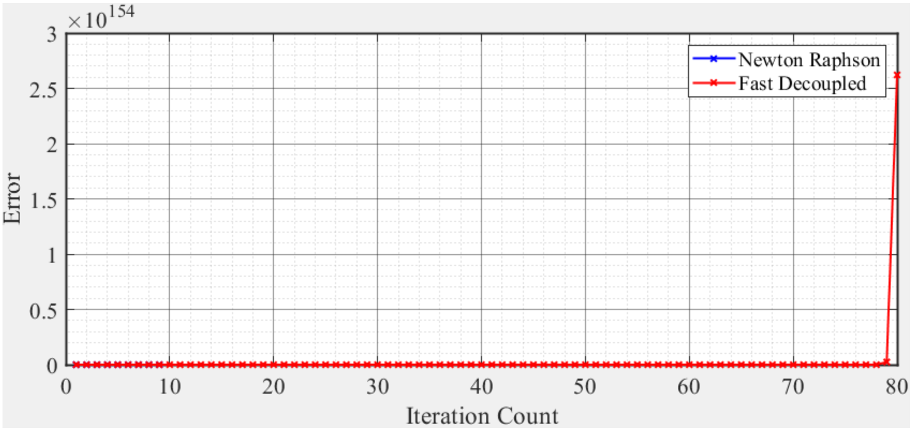
For the error tolerance of 1e-3, calculations are not converging

Convergence Curves

For case 1



For case 2


×10^154
3
2.5
2
1.5
1
0.5
0
Error
0
10
20
30
40
50
60
70
80
Iteration Count
Newton Raphson
Fast Decoupled

Appendix

```
% main.m
clc
clear all; close all;

% Initializing Kundur 2 area system and importing data
n_bus = 11;
bus_data = importdata('ieee11bus.txt').data;
% bus_data = importdata('ieee11bus_allPV.txt').data;
branch_data = importdata('ieee11branch.txt').data;

% Ybus formation
t = 1; % 0 for without tap, 1 for with tap
Y = y_bus_calc(n_bus,bus_data,branch_data,t);

% Scheduled power calculation
base_MVA = 100;
P_inj = (bus_data(:,8) - bus_data(:,6)) / base_MVA;
Q_inj = (bus_data(:,9) - bus_data(:,7)) / base_MVA;

% Finding bus types
pv_i = find(bus_data(:,3) == 2);
pq_i = find(bus_data(:,3) == 0);
n_pv = length(pv_i);
n_pq = length(pq_i);

% Initializing Voltage magnitude and angles
V = bus_data(:,11);
V(find(V(:)==0)) = 1;
T = zeros(n_bus,1);

[V_data,T_data] = NR(bus_data,V,T,P_inj,Q_inj,n_bus,Y,n_pq,pq_i);
% [V_data,T_data] = FD(bus_data,V,T,P_inj,Q_inj,n_bus,Y,n_pq,pq_i);

% P,Q calculation after convergence
[P,Q] = PQ_calc(V1_data(:,size(V1_data,2)),T1_data(:,size(T1_data,2)),Y)

% plotting convergence curves
mplot([1:size(V1_data,2)],T1,[1:size(V2_data,2)],T2)
```

```
% y_bus_calc.m

function Y = y_bus_calc(N_bs,D_bs,D_br,t)
Y = zeros(N_bs);

% Calculating elements of Ybus
for k = 1:size(D_br,1)
    Y(D_br(k,1),D_br(k,1)) = Y(D_br(k,1),D_br(k,1)) + 1/(D_br(k,7) +
i*D_br(k,8)) + i*D_br(k,9)/2;
    Y(D_br(k,2),D_br(k,2)) = Y(D_br(k,2),D_br(k,2)) + 1/(D_br(k,7) +
i*D_br(k,8)) + i*D_br(k,9)/2;
    Y(D_br(k,1),D_br(k,2)) = -1/(D_br(k,7) + i*D_br(k,8));
    Y(D_br(k,2),D_br(k,1)) = Y(D_br(k,1),D_br(k,2));
end
for k = 1:N_bs
    Y(k,k) = Y(k,k) + D_bs(k,14) + i*D_bs(k,15);
end

% adjusting for taps
```

```
if(t == 1)
    for k = 1:size(D_br,1)
        if(D_br(k,15) ~= 0)
            t = D_br(k,15);
            ((t^2) / i*D_br(k,8));
            Y(D_br(k,1),D_br(k,1)) = Y(D_br(k,1),D_br(k,1)) +
Y(D_br(k,1),D_br(k,2)) - (Y(D_br(k,1),D_br(k,2)))/(t^2);
            Y(D_br(k,1),D_br(k,1));
            Y(D_br(k,1),D_br(k,2)) = Y(D_br(k,1),D_br(k,2))/t;
            Y(D_br(k,2),D_br(k,1)) = Y(D_br(k,1),D_br(k,2));
        end
    end
end
end
```

```
% NR.m

function [V_data,T_data] = NR(bus_data,V,T,P_inj,Q_inj,n_bus,Y,n_pq,pq_i)
% Initializing index
i = 0;
Tol = 1;
del_T = zeros(n_bus,1);
del_V = zeros(n_bus,1);

% Iteration loop
while(Tol > 1e-3 & i < 100)
    i = i+1
    V = V+del_V;
    T = T+del_T;
    T_data(:,i) = T;
    V_data(:,i) = V;
    [del_P, del_Q] = dpdq_calc(bus_data,V,T,P_inj,Q_inj,n_bus,Y);
    dpdq = [del_P, del_Q]; % mismatch calculation
    J = J_calc(bus_data,V,T,Y,n_bus,n_pq,pq_i); % Jacobian calculation
    delta = fwd_bwd(J,dpdq); % finding errors
    del_T = [0 delta(1:n_bus-1)]';
    for j = 1:n_pq
        del_V(pq_i(j)) = delta(n_bus+j-1);
    end
    Tol = max(abs(delta)) % updating error for convergence
end
end
```

```
% dpdq.m

function [del_P, del_Q] = dpdq_calc(bus_data,V,T,P_inj,Q_inj,n_bus,Y)
P = zeros(n_bus,1);
Q = zeros(n_bus,1);
Pi = 1;
Qi = 1;
for i = 1:n_bus
    if(bus_data(i,3) ~= 3)
        for j = 1:n_bus
            P(i) = P(i) + V(i)*V(j)*abs(Y(i,j))*cos(T(i)-T(j)-
angle(Y(i,j)));
            Q(i) = Q(i) + V(i)*V(j)*abs(Y(i,j))*sin(T(i)-T(j)-
angle(Y(i,j)));
        end
        del_P(Pi) = P_inj(i) - P(i);
        Pi = Pi+1;
        if(bus_data(i,3) == 0)
```

```
            del_Q(Qi) = Q_inj(i) - Q(i);
            Qi = Qi+1;
        end
    end
end
end
```

```
% J_calc.m

function J = J_calc(bus_data,V,T,Y,n_bus,n_pq,pq_i)

% J1 calculation
J1 = zeros(n_bus-1);
for i = 1:n_bus
    for j = 1:n_bus
        if(bus_data(i,3) ~=3 & bus_data(j,3) ~=3)
            if(i==j)
                for k = 1:n_bus
                    J1(i-1,j-1) = J1(i-1,j-
1)+(V(i)*V(k)*abs(Y(i,k))*sin(angle(Y(i,k))-T(i)+T(k)));
                end
                J1(i-1,j-1) = J1(i-1,j-1) - ((V(i)^2) * (imag(Y(i,i))));
            else
                J1(i-1,j-1) = -V(i)*V(j)*abs(Y(i,j))*sin(angle(Y(i,j))-
T(i)+T(j));
            end
        end
    end
end
J1;

% J2 calculation
J2 = zeros(n_bus-1,n_pq);
for i = 2:n_bus
    for j = 1:n_pq
        n = pq_i(j);
        if(n == i)
            for k = 1:n_bus
                J2(i-1,j) = J2(i-
1,j)+(V(i)*V(k)*abs(Y(i,k))*cos(angle(Y(i,k))-T(i)+T(k)));
            end
            J2(i-1,j) = (J2(i-1,j) + ((V(i)^2) * (real(Y(i,i)))))/V(i);
        else
            J2(i-1,j) = V(i)*abs(Y(i,n))*cos(angle(Y(i,n))-T(i)+T(n));
        end
    end
end
J2;

% J3 calculation
J3 = zeros(n_pq,n_bus-1);
for i = 1:n_pq
    n = pq_i(i);
    for j = 2:n_bus
        if(n==j)
            for k = 1:n_bus
                J3(i,j-1) = J3(i,j-
1)+(V(n)*V(k)*abs(Y(n,k))*cos(angle(Y(n,k))-T(n)+T(k)));
            end
            J3(i,j-1) = J3(i,j-1) - ((V(n)^2) * (real(Y(n,n))));
        else
```

```
            J3(i,j-1) = -V(n)*V(j)*abs(Y(n,j))*cos(angle(Y(n,j))-
T(n)+T(j));
        end
    end
end
J3;

% J4 calculation
J4 = zeros(n_pq);
for i = 1:n_pq
    n1 = pq_i(i);
    for j = 1:n_pq
        n2 = pq_i(j);
        if(n1==n2)
            for k = 1:n_bus
                J4(i,j) =
J4(i,j)+(V(n1)*V(k)*abs(Y(n1,k))*sin(angle(Y(n1,k))-T(n1)+T(k)));
            end
            J4(i,j) = - J4(i,j) - ((V(n1)^2) * (imag(Y(n1,n1))));
        else
            J4(i,j) = -V(n1)*abs(Y(n1,n2))*sin(angle(Y(n1,n2))-
T(n1)+T(n2));
        end
    end
end
J4;
J = [J1, J2; J3, J4];
end
```

```
% fwd_bwd.m
function x = fwd_bwd(A,b)
[L, U] = LU(A);

% Forward Substitution
for k = 1:length(A)
    s = 0;
    for j = 1:k-1
        s = s + (L(k,j)*y(j));
    end
    y(k) = (b(k) - s) / L(k,k);
end

% Backward Substitution
for k = length(A):-1:1
    s = 0;
    for j = k+1:length(A)
        s = s + (U(k,j)*x(j));
    end
    x(k) = y(k) - s;
end
end
```

```
% LU.m

function [L, U] = LU(a)
Q = zeros(length(a));
for j = 1:length(a)
    for k = j:length(a)
        s = 0;
        for m = 1:j-1
```

```
            s = s + (Q(k,m)*Q(m,j));
        end
        Q(k,j) = a(k,j) - s;
    end
    if j < length(a)
        for k = j+1:length(a)
            s = 0;
            for m = 1:j-1
                s = s + (Q(j,m)*Q(m,k));
            end
            Q(j,k) = (a(j,k) - s) / Q(j,j);
        end
    end
end
L = tril(Q);
U = Q - L + eye(length(a));
end
```

```
% FD.m
function [V_data,T_data] = FD(bus_data,V,T,P_inj,Q_inj,n_bus,Y,n_pq,pq_i)
% Initializing index
B = imag(Y);
B_T =  - B(2:n_bus,2:n_bus);
B_V = - B(pq_i,pq_i);
i = 0;
Tol = 1;
del_T = zeros(n_bus,1);
del_V = zeros(n_bus,1);

% Iteration loop
while(Tol > 1e-5 & i < 100)
    i = i+1
    V = V+del_V;
    T = T+del_T;
    T_data(:,i) = T;
    V_data(:,i) = V;
    [del_P, del_Q] = dpdq_calc(bus_data,V,T,P_inj,Q_inj,n_bus,Y);
    P_T = del_P'./V(2:n_bus);
    d_T = fwd_bwd(B_T,P_T);
    Q_V = del_Q'./V(pq_i);
    d_V = fwd_bwd(B_V,Q_V);
    del_T = [0 d_T]'; % angle calculation
    for j = 1:n_pq
        del_V(pq_i(j)) = d_V(j); % magnitude calculation
    end
    Tol = max(abs([P_T; Q_V]));
end
end
```

```
% PQ_calc.m
function [P,Q] = PQ_calc(V,T,Y)
n_bus = size(V,1);
P = zeros(n_bus,1);
Q = zeros(n_bus,1);
for i = 1:n_bus
    for j = 1:n_bus
        P(i) = P(i) + V(i)*V(j)*abs(Y(i,j))*cos(T(i)-T(j)-angle(Y(i,j)));
        Q(i) = Q(i) + V(i)*V(j)*abs(Y(i,j))*sin(T(i)-T(j)-angle(Y(i,j)));
    end
end
end
```

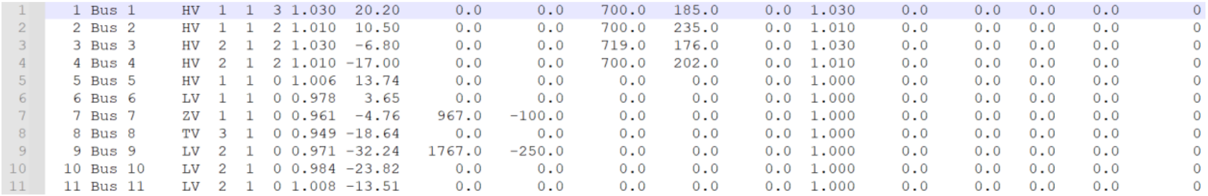
```matlab
% mplot.m
function mplot(x1,y1,x2,y2)
x_label = 'Iteration Count'; % x axis label
y_label = 'Error'; % y axis label
legend_name = {'Newton Raphson','Fast Decoupled'}; % legend names

figure('Renderer', 'painters', 'Position', [10 10 1000 400])
plot(x1,y1,'-xb','LineWidth',1.5)
hold on
plot(x2,y2,'-xr','LineWidth',1.5)
xlabel(x_label,'FontSize',18,'FontName','Times New Roman')
ylabel(y_label,'FontSize',18,'FontName','Times New Roman')
legend (legend_name,'Location','northeast')
set(gca,'fontsize',16,'Fontname','Times New Roman','GridAlpha',0.5)
ax = gca

ax.XRuler.Axle.LineWidth = 1.5;
ax.YRuler.Axle.LineWidth = 1.5;
grid
grid minor
% legend (legend_name,'Location','southeast')
saveas(gca,'plot.png')
end
```

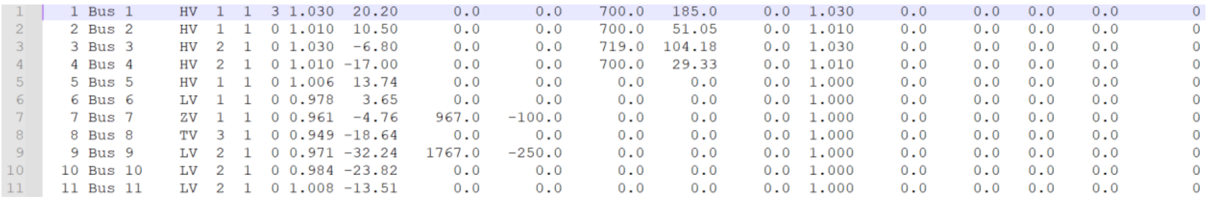
Input

Bus data for Case 1



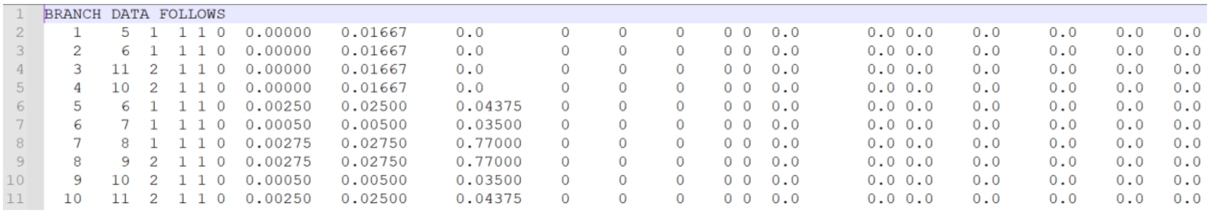
```
 1 Bus 1     HV  1  1  3 1.030  20.20      0.0      0.0    700.0   185.0    0.0 1.030    0.0    0.0  0.0   0.0        0
 2 Bus 2     HV  1  1  2 1.010  10.50      0.0      0.0    700.0   235.0    0.0 1.010    0.0    0.0  0.0   0.0        0
 3 Bus 3     HV  2  1  2 1.030  -6.80      0.0      0.0    719.0   176.0    0.0 1.030    0.0    0.0  0.0   0.0        0
 4 Bus 4     HV  2  1  2 1.010 -17.00      0.0      0.0    700.0   202.0    0.0 1.010    0.0    0.0  0.0   0.0        0
 5 Bus 5     HV  1  1  0 1.006  13.74      0.0      0.0      0.0     0.0    0.0 1.000    0.0    0.0  0.0   0.0        0
 6 Bus 6     LV  1  1  0 0.978   3.65      0.0      0.0      0.0     0.0    0.0 1.000    0.0    0.0  0.0   0.0        0
 7 Bus 7     ZV  1  1  0 0.961  -4.76    967.0   -100.0      0.0     0.0    0.0 1.000    0.0    0.0  0.0   0.0        0
 8 Bus 8     TV  3  1  0 0.949 -18.64      0.0      0.0      0.0     0.0    0.0 1.000    0.0    0.0  0.0   0.0        0
 9 Bus 9     LV  2  1  0 0.971 -32.24   1767.0   -250.0      0.0     0.0    0.0 1.000    0.0    0.0  0.0   0.0        0
10 Bus 10    LV  2  1  0 0.984 -23.82      0.0      0.0      0.0     0.0    0.0 1.000    0.0    0.0  0.0   0.0        0
11 Bus 11    LV  2  1  0 1.008 -13.51      0.0      0.0      0.0     0.0    0.0 1.000    0.0    0.0  0.0   0.0        0
```

Bus data for Case 2



```
 1 Bus 1     HV  1  1  3 1.030  20.20      0.0      0.0    700.0   185.0    0.0 1.030    0.0    0.0  0.0   0.0        0
 2 Bus 2     HV  1  1  0 1.010  10.50      0.0      0.0    700.0   51.05    0.0 1.010    0.0    0.0  0.0   0.0        0
 3 Bus 3     HV  2  1  0 1.030  -6.80      0.0      0.0    719.0  104.18    0.0 1.030    0.0    0.0  0.0   0.0        0
 4 Bus 4     HV  2  1  0 1.010 -17.00      0.0      0.0    700.0   29.33    0.0 1.010    0.0    0.0  0.0   0.0        0
 5 Bus 5     HV  1  1  0 1.006  13.74      0.0      0.0      0.0     0.0    0.0 1.000    0.0    0.0  0.0   0.0        0
 6 Bus 6     LV  1  1  0 0.978   3.65      0.0      0.0      0.0     0.0    0.0 1.000    0.0    0.0  0.0   0.0        0
 7 Bus 7     ZV  1  1  0 0.961  -4.76    967.0   -100.0      0.0     0.0    0.0 1.000    0.0    0.0  0.0   0.0        0
 8 Bus 8     TV  3  1  0 0.949 -18.64      0.0      0.0      0.0     0.0    0.0 1.000    0.0    0.0  0.0   0.0        0
 9 Bus 9     LV  2  1  0 0.971 -32.24   1767.0   -250.0      0.0     0.0    0.0 1.000    0.0    0.0  0.0   0.0        0
10 Bus 10    LV  2  1  0 0.984 -23.82      0.0      0.0      0.0     0.0    0.0 1.000    0.0    0.0  0.0   0.0        0
11 Bus 11    LV  2  1  0 1.008 -13.51      0.0      0.0      0.0     0.0    0.0 1.000    0.0    0.0  0.0   0.0        0
```

Branch data



```
BRANCH DATA FOLLOWS
   1    5  1  1 1 0  0.00000   0.01667   0.0        0    0    0   0 0  0.0       0.0 0.0   0.0    0.0   0.0   0.0
   2    6  1  1 1 0  0.00000   0.01667   0.0        0    0    0   0 0  0.0       0.0 0.0   0.0    0.0   0.0   0.0
   3   11  2  1 1 0  0.00000   0.01667   0.0        0    0    0   0 0  0.0       0.0 0.0   0.0    0.0   0.0   0.0
   4   10  2  1 1 0  0.00000   0.01667   0.0        0    0    0   0 0  0.0       0.0 0.0   0.0    0.0   0.0   0.0
   5    6  1  1 1 0  0.00250   0.02500   0.04375    0    0    0   0 0  0.0       0.0 0.0   0.0    0.0   0.0   0.0
   6    7  1  1 1 0  0.00050   0.00500   0.03500    0    0    0   0 0  0.0       0.0 0.0   0.0    0.0   0.0   0.0
   7    8  1  1 1 0  0.00275   0.02750   0.77000    0    0    0   0 0  0.0       0.0 0.0   0.0    0.0   0.0   0.0
   8    9  2  1 1 0  0.00275   0.02750   0.77000    0    0    0   0 0  0.0       0.0 0.0   0.0    0.0   0.0   0.0
   9   10  2  1 1 0  0.00050   0.00500   0.03500    0    0    0   0 0  0.0       0.0 0.0   0.0    0.0   0.0   0.0
  10   11  2  1 1 0  0.00250   0.02500   0.04375    0    0    0   0 0  0.0       0.0 0.0   0.0    0.0   0.0   0.0
```