Assignment 1

Submitted by Athul Jose P
WSU ID 011867566

Executive Summary

Newton Raphson Method

Steps in main.m function

1. Initializing 14 bus 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 $\Delta\delta$
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 (ΔV and $\Delta\delta$)

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

Results

Without Tap

For the error tolerance of 1e-5, calculations converged at 5th Iteration

Ybus

	1	2	3	4	5	6	7
1	6.0250 - 19.4471i	-4.9991 + 15.2631i	0.0000 + 0.0000i	0.0000 + 0.0000i	-1.0259 + 4.2350i	0.0000 + 0.0000i	0.0000 + 0.0000i
2	-4.9991 + 15.2631i	9.5213 - 30.2721i	-1.1350 + 4.7819i	-1.6860 + 5.1158i	-1.7011 + 5.1939i	0.0000 + 0.0000i	0.0000 + 0.0000i
3	0.0000 + 0.0000i	-1.1350 + 4.7819i	3.1210 - 9.8224i	-1.9860 + 5.0688i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
4	0.0000 + 0.0000i	-1.6860 + 5.1158i	-1.9860 + 5.0688i	10.5130 - 38.3197i	-6.8410 + 21.5786i	0.0000 + 0.0000i	0.0000 + 4.7819i
5	-1.0259 + 4.2350i	-1.7011 + 5.1939i	0.0000 + 0.0000i	-6.8410 + 21.5786i	9.5680 - 34.9335i	0.0000 + 3.9679i	0.0000 + 0.0000i
6	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 3.9679i	6.5799 - 17.3407i	0.0000 + 0.0000i
7	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 4.7819i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 - 19.5490i
8	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 5.6770i
9	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 1.7980i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 9.0901i
10	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
11	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	-1.9550 + 4.0941i	0.0000 + 0.0000i
12	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	-1.5260 + 3.1760i	0.0000 + 0.0000i
13	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	-3.0989 + 6.1028i	0.0000 + 0.0000i
14	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i

	8	9	10	11	12	13	14
1	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
2	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
3	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
4	0.0000 + 0.0000i	0.0000 + 1.7980i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
5	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
6	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	-1.9550 + 4.0941i	-1.5260 + 3.1760i	-3.0989 + 6.1028i	0.0000 + 0.0000i
7	0.0000 + 5.6770i	0.0000 + 9.0901i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
8	0.0000 - 5.6770i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
9	0.0000 + 0.0000i	5.3261 - 24.0925i	-3.9020 + 10.3654i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	-1.4240 + 3.0291i
10	0.0000 + 0.0000i	-3.9020 + 10.3654i	5.7829 - 14.7683i	-1.8809 + 4.4029i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
11	0.0000 + 0.0000i	0.0000 + 0.0000i	-1.8809 + 4.4029i	3.8359 - 8.4970i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
12	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	4.0150 - 5.4279i	-2.4890 + 2.2520i	0.0000 + 0.0000i
13	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	-2.4890 + 2.2520i	6.7249 - 10.6697i	-1.1370 + 2.3150i
14	0.0000 + 0.0000i	-1.4240 + 3.0291i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	-1.1370 + 2.3150i	2.5610 - 5.3440i

Final results

Bus Number	V(magnitude)	V(angle)	P	Q
1	1.0600	0	2.3238	-0.2353
2	1.0450	-0.0864	0.1830	0.1471
3	1.0100	-0.2202	-0.9420	-0.0099
4	1.0295	-0.1819	-0.4780	0.0390
5	1.0349	-0.1563	-0.0760	-0.0160
6	1.0700	-0.2561	-0.1120	0.3286
7	1.0559	-0.2366	0.0000	0.0000
8	1.0900	-0.2366	0.0000	0.2111
9	1.0497	-0.2648	-0.2950	-0.1660
10	1.0458	-0.2682	-0.0900	-0.0580
11	1.0543	-0.2643	-0.0350	-0.0180
12	1.0547	-0.2708	-0.0610	-0.0160
13	1.0495	-0.2718	-0.1350	-0.0580
14	1.0315	-0.2854	-0.1490	-0.0500

With Tap

For the error tolerance of 1e-5, calculations converged at 5th Iteration

Ybus

	1	2	3	4	5	6	7
1	6.0250 - 19.4471i	-4.9991 + 15.2631i	0.0000 + 0.0000i	0.0000 + 0.0000i	-1.0259 + 4.2350i	0.0000 + 0.0000i	0.0000 + 0.0000i
2	-4.9991 + 15.2631i	9.5213 - 30.2721i	-1.1350 + 4.7819i	-1.6860 + 5.1158i	-1.7011 + 5.1939i	0.0000 + 0.0000i	0.0000 + 0.0000i
3	0.0000 + 0.0000i	-1.1350 + 4.7819i	3.1210 - 9.8224i	-1.9860 + 5.0688i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
4	0.0000 + 0.0000i	-1.6860 + 5.1158i	-1.9860 + 5.0688i	10.5130 - 38.6542i	-6.8410 + 21.5786i	0.0000 + 0.0000i	0.0000 + 4.8895i
5	-1.0259 + 4.2350i	-1.7011 + 5.1939i	0.0000 + 0.0000i	-6.8410 + 21.5786i	9.5680 - 35.5336i	0.0000 + 4.2574i	0.0000 + 0.0000i
6	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 4.2574i	6.5799 - 17.3407i	0.0000 + 0.0000i
7	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 4.8895i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 - 19.5490i
8	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 5.6770i
9	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 1.8555i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 9.0901i
10	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
11	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	-1.9550 + 4.0941i	0.0000 + 0.0000i
12	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	-1.5260 + 3.1760i	0.0000 + 0.0000i
13	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	-3.0989 + 6.1028i	0.0000 + 0.0000i
14	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i

	8	9	10	11	12	13	14
1	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
2	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
3	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
4	0.0000 + 0.0000i	0.0000 + 1.8555i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
5	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
6	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	-1.9550 + 4.0941i	-1.5260 + 3.1760i	-3.0989 + 6.1028i	0.0000 + 0.0000i
7	0.0000 + 5.6770i	0.0000 + 9.0901i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
8	0.0000 - 5.6770i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
9	0.0000 + 0.0000i	5.3261 - 24.0925i	-3.9020 + 10.3654i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	-1.4240 + 3.0291i
10	0.0000 + 0.0000i	-3.9020 + 10.3654i	5.7829 - 14.7683i	-1.8809 + 4.4029i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
11	0.0000 + 0.0000i	0.0000 + 0.0000i	-1.8809 + 4.4029i	3.8359 - 8.4970i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i
12	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	4.0150 - 5.4279i	-2.4890 + 2.2520i	0.0000 + 0.0000i
13	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	-2.4890 + 2.2520i	6.7249 - 10.6697i	-1.1370 + 2.3150i
14	0.0000 + 0.0000i	-1.4240 + 3.0291i	0.0000 + 0.0000i	0.0000 + 0.0000i	0.0000 + 0.0000i	-1.1370 + 2.3150i	2.5610 - 5.3440i

Final results

Bus Number	V(magnitude)	V(angle)	P	Q
1	1.0600	0	2.3239	-0.1655
2	1.0450	-0.0870	0.1830	0.3086
3	1.0100	-0.2221	-0.9420	0.0608
4	1.0177	-0.1800	-0.4780	0.0390
5	1.0195	-0.1531	-0.0760	-0.0160
6	1.0700	-0.2482	-0.1120	0.0523
7	1.0615	-0.2332	0.0000	0.0000
8	1.0900	-0.2332	-0.0000	0.1762
9	1.0559	-0.2607	-0.2950	-0.1660
10	1.0510	-0.2635	-0.0900	-0.0580
11	1.0569	-0.2581	-0.0350	-0.0180
12	1.0552	-0.2631	-0.0610	-0.0160
13	1.0504	-0.2645	-0.1350	-0.0580
14	1.0355	-0.2798	-0.1490	-0.0500

Fast Decoupled Method

Steps in main.m function

1 Initializing 14 bus 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 (ΔV and Δδ)

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

Without Tap

For the error tolerance of 1e-5, calculations converged at 33rd Iteration

Final results

Bus Number	V(magnitude)	V(angle)	P	Q
1	1.0600	0	2.3238	-0.2353
2	1.0450	-0.0864	0.1830	0.1471
3	1.0100	-0.2202	-0.9420	-0.0099
4	1.0295	-0.1819	-0.4780	0.0390
5	1.0349	-0.1563	-0.0760	-0.0160
6	1.0700	-0.2561	-0.1120	0.3286
7	1.0559	-0.2366	0.0000	0.0000
8	1.0900	-0.2366	-0.0000	0.2112
9	1.0497	-0.2648	-0.2950	-0.1660
10	1.0458	-0.2682	-0.0900	-0.0580
11	1.0543	-0.2643	-0.0350	-0.0180
12	1.0547	-0.2708	-0.0610	-0.0161
13	1.0495	-0.2718	-0.1350	-0.0579
14	1.0315	-0.2854	-0.1490	-0.0500

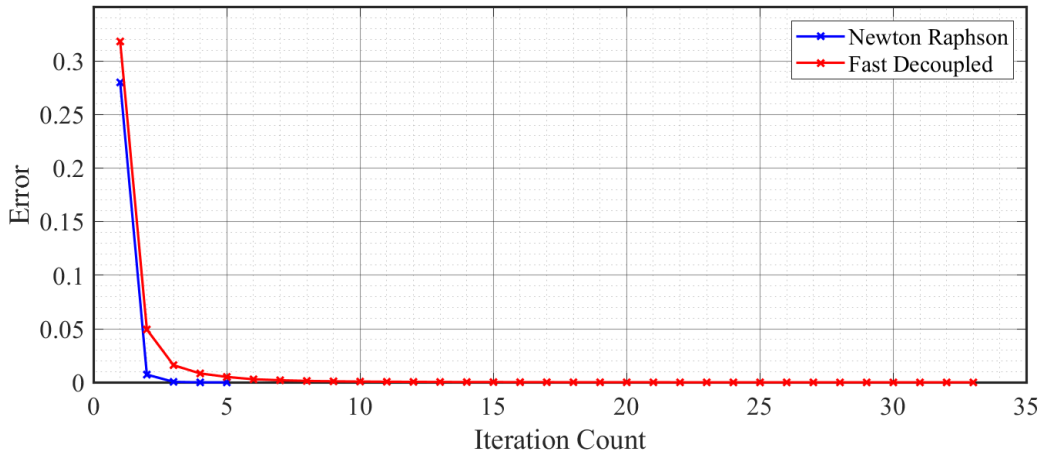
With Tap

For the error tolerance of 1e-5, calculations converged at 33rd Iteration

Final results

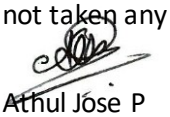
Bus Number	V(magnitude)	V(angle)	P	Q
1	1.0600	0	2.3239	-0.1655
2	1.0450	-0.0870	0.1830	0.3086
3	1.0100	-0.2221	-0.9420	0.0608
4	1.0177	-0.1800	-0.4780	0.0390
5	1.0195	-0.1531	-0.0760	-0.0160
6	1.0700	-0.2482	-0.1120	0.0523
7	1.0615	-0.2332	0.0000	0.0000
8	1.0900	-0.2332	-0.0000	0.1762
9	1.0559	-0.2607	-0.2950	-0.1660
10	1.0510	-0.2635	-0.0900	-0.0580
11	1.0569	-0.2581	-0.0350	-0.0180
12	1.0552	-0.2631	-0.0610	-0.0161
13	1.0504	-0.2645	-0.1350	-0.0579
14	1.0355	-0.2798	-0.1490	-0.0500

Convergence Curves



Statement

I, Athul Jose P, states that all the code written and submitted here is completely done by me. I have not taken any help from others or any online resources.



Athul Jose P

Appendix

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

% Initializing 14 bus and importing data
n_bus = 14;
bus_data = importdata('ieee14bus.txt');
bus_data = bus_data.data;
branch_data = importdata('ieee14branch.txt');
branch_data = branch_data.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
V = V_data(:,size(V_data,2))
T = T_data(:,size(T_data,2))
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
P
Q
```

```
% 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-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);
    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))));
        else
            J2(i-1,j) = V(i)*V(n)*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)*V(n2)*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([del_T; del_V]));
end
end
```