**Assignment 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

**Results**

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

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

Ybus

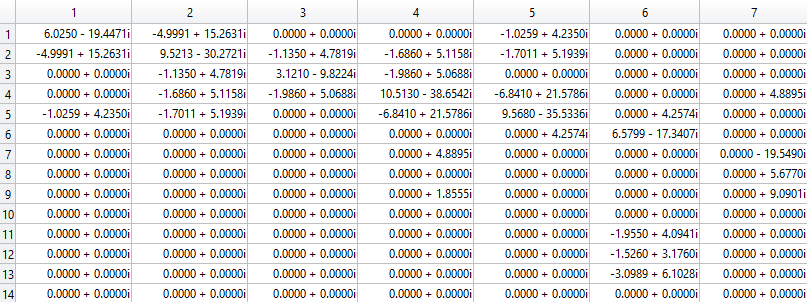
Final results

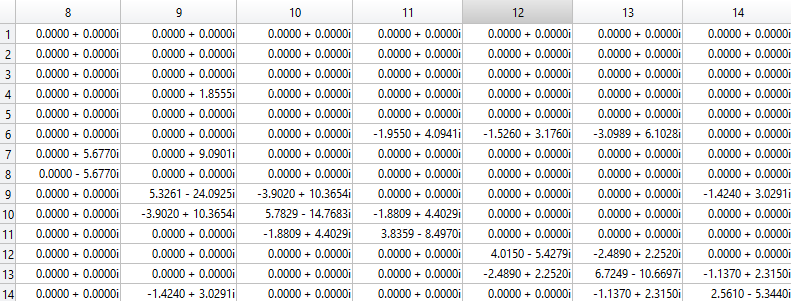
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bus Number | V(magnitude) | V(angle) | P | Q |
| 1 | 1.0600 | 0 | 2.3238 | -0.2353 |
| 2 | 1.0450 | -0.0864 | 0.1830 | 0.1471 |
| 3 | 1.0100 | -0.2202 | -0.9420 | -0.0099 |
| 4 | 1.0295 | -0.1819 | -0.4780 | 0.0390 |
| 5 | 1.0349 | -0.1563 | -0.0760 | -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





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

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 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 |

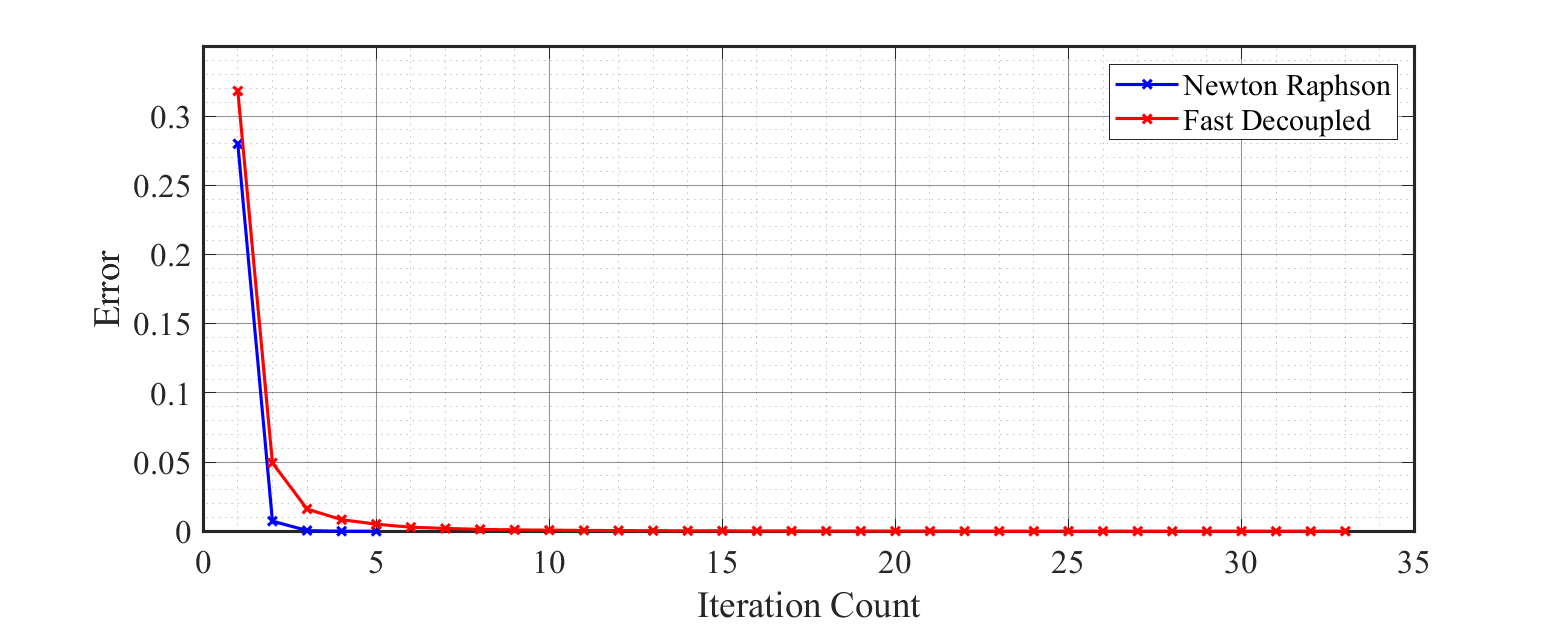
With Tap

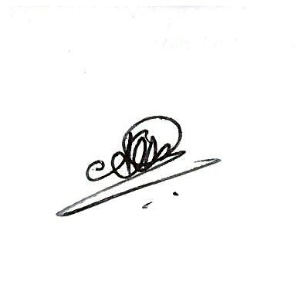
For the error tolerance of 1e-5, calculations converged at 33rd Iteration

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**