

VALIDATION REPORT

Implementation of Standard Backward Forward Sweep(B/FS) method and Branch Current Based B/FS by MATLAB programming language.

Course: Electric Power Distribution Systems

Professor: Dr. Karimi

Student: Hamed Najafi

University of Kashan

JANUARY 9, 2023



دانشگاه کاشان

Results

By performing a power flow calculation on IEEE-33 bus radial distribution system by MATPOWER7.1 we can see the same results as we had for BFS MATLAB code in initial report.

MATPOWER Output:

MATPOWER Version 7.1, 08-Oct-2020 -- AC Power Flow (Newton)

Newton's method power flow (power balance, polar) converged in 4 iterations.

Converged in 0.39 seconds

```
=====
|   System Summary                               |
=====
```

How many?		How much?	P (MW)	Q (MVar)
Buses	33	Total Gen Capacity	10.0	-10.0 to 10.0
Generators	1	On-line Capacity	10.0	-10.0 to 10.0
Committed Gens	1	Generation (actual)	3.9	2.4
Loads	32	Load	3.7	2.3
Fixed	32	Fixed	3.7	2.3
Dispatchable	0	Dispatchable	-0.0 of -0.0	-0.0
Shunts	0	Shunt (inj)	-0.0	0.0
Branches	37	Losses ($I^2 * Z$)	0.20	0.14
Transformers	0	Branch Charging (inj)	-	0.0
Inter-ties	0	Total Inter-tie Flow	0.0	0.0
Areas	1			

	Minimum	Maximum
Voltage Magnitude	0.913 p.u. @ bus 18	1.000 p.u. @ bus 1
Voltage Angle	-0.50 deg @ bus 18	0.50 deg @ bus 30
P Losses ($I^2 * R$)	-	0.05 MW @ line 2-3
Q Losses ($I^2 * X$)	-	0.03 MVar @ line 5-6

Bus Data

Bus #	Voltage Mag(pu)	Angle(deg)	Generation P (MW)	Generation Q (MVar)	Load P (MW)	Load Q (MVar)
-------	-----------------	------------	-------------------	---------------------	-------------	---------------

1	1.000	0.000*	3.92	2.44	-	-
2	0.997	0.014	-	-	0.10	0.06
3	0.983	0.096	-	-	0.09	0.04
4	0.975	0.162	-	-	0.12	0.08
5	0.968	0.228	-	-	0.06	0.03
6	0.950	0.134	-	-	0.06	0.02
7	0.946	-0.096	-	-	0.20	0.10
8	0.941	-0.060	-	-	0.20	0.10
9	0.935	-0.133	-	-	0.06	0.02
10	0.929	-0.196	-	-	0.06	0.02
11	0.928	-0.189	-	-	0.04	0.03
12	0.927	-0.177	-	-	0.06	0.04
13	0.921	-0.269	-	-	0.06	0.04
14	0.919	-0.347	-	-	0.12	0.08
15	0.917	-0.385	-	-	0.06	0.01
16	0.916	-0.408	-	-	0.06	0.02
17	0.914	-0.485	-	-	0.06	0.02
18	0.913	-0.495	-	-	0.09	0.04
19	0.997	0.004	-	-	0.09	0.04
20	0.993	-0.063	-	-	0.09	0.04
21	0.992	-0.083	-	-	0.09	0.04
22	0.992	-0.103	-	-	0.09	0.04
23	0.979	0.065	-	-	0.09	0.05
24	0.973	-0.024	-	-	0.42	0.20
25	0.969	-0.067	-	-	0.42	0.20
26	0.948	0.173	-	-	0.06	0.03
27	0.945	0.229	-	-	0.06	0.03
28	0.934	0.312	-	-	0.06	0.02
29	0.926	0.390	-	-	0.12	0.07
30	0.922	0.496	-	-	0.20	0.60
31	0.918	0.411	-	-	0.15	0.07
32	0.917	0.388	-	-	0.21	0.10
33	0.917	0.380	-	-	0.06	0.04

Total: 0.203(MW) 0.14(MVar)

BFS code Output (rounded to 3 decimal places):

Bus NO |U|(pu) $\theta(^{\circ})$

1	1	0
2	0.997	0.014
3	0.983	0.096
4	0.975	0.162
5	0.968	0.228
6	0.95	0.134
7	0.946	-0.096
8	0.941	-0.06
9	0.935	-0.133
10	0.929	-0.196
11	0.928	-0.189
12	0.927	-0.177
13	0.921	-0.268
14	0.919	-0.347
15	0.917	-0.385
16	0.916	-0.408
17	0.914	-0.485
18	0.913	-0.495
19	0.997	0.004
20	0.993	-0.063
21	0.992	-0.083
22	0.992	-0.103
23	0.979	0.065
24	0.973	-0.024
25	0.969	-0.067
26	0.948	0.173
27	0.945	0.229
28	0.934	0.312
29	0.926	0.39
30	0.922	0.495
31	0.918	0.411
32	0.917	0.388
33	0.917	0.38

Number of Iterations=3

Elapsed time is 0.025467 seconds.

Total Power Loss in lines (kW) =202.5193

Total ReactivePower Consumed by Lines (kVAr) =135.0338

So the written BFS code is working correctly.

By the same way I found a minor mistake in my previous code that is mentioned in the next section ("Correction"). After correcting two lines we can see that our BCBBFS code has no problem as well. The corrected code output is shown after MATPOWER output.

MATPOWER Output:

MATPOWER Version 7.1, 08-Oct-2020 -- AC Power Flow (Newton)
Newton's method power flow (power balance, polar) converged in 4 iterations.

Converged in 0.11 seconds

```
=====
|   System Summary                               |
=====
```

How many?		How much?	P (MW)	Q (MVar)
Buses	33	Total Gen Capacity	30.0	-10.2 to 10.3
Generators	3	On-line Capacity	30.0	-10.2 to 10.3
Committed Gens	3	Generation (actual)	3.8	2.3
Loads	32	Load	3.7	2.3
Fixed	32	Fixed	3.7	2.3
Dispatchable	0	Dispatchable	-0.0 of -0.0	-0.0
Shunts	0	Shunt (inj)	-0.0	0.0
Branches	37	Losses ($I^2 * Z$)	0.13	0.09
Transformers	0	Branch Charging (inj)	-	0.0
Inter-ties	0	Total Inter-tie Flow	0.0	0.0
Areas	1			

	Minimum	Maximum
Voltage Magnitude	0.935 p.u. @ bus 32	1.000 p.u. @ bus 1
Voltage Angle	-0.23 deg @ bus 16	0.57 deg @ bus 30
P Losses ($I^2 * R$)	-	0.04 MW @ line 2-3
Q Losses ($I^2 * X$)	-	0.02 MVar @ line 5-6

```
|   Bus Data                               |
=====
```

Bus	Voltage		Generation		Load	
#	Mag(pu)	Ang(deg)	P (MW)	Q (MVar)	P (MW)	Q (MVar)

1	1.000	0.000*	3.34	2.10	-	-
2	0.997	0.013	-	-	0.10	0.06
3	0.985	0.088	-	-	0.09	0.04
4	0.979	0.149	-	-	0.12	0.08
5	0.974	0.210	-	-	0.06	0.03
6	0.959	0.161	-	-	0.06	0.02
7	0.957	-0.013	-	-	0.20	0.10
8	0.953	0.003	-	-	0.20	0.10
9	0.950	-0.056	-	-	0.06	0.02
10	0.947	-0.106	-	-	0.06	0.02
11	0.946	-0.104	-	-	0.04	0.03
12	0.945	-0.103	-	-	0.06	0.04
13	0.943	-0.164	-	-	0.06	0.04
14	0.942	-0.199	-	-	0.12	0.08
15	0.943	-0.218	-	-	0.06	0.01
16	0.943	-0.230	-	-	0.06	0.02
17	0.945	-0.208	-	-	0.06	0.02
18	0.946	-0.203	0.25	0.15	0.09	0.04
19	0.997	0.003	-	-	0.09	0.04
20	0.994	-0.056	-	-	0.03	0.00
21	0.994	-0.075	-	-	0.09	0.04
22	0.993	-0.096	-	-	0.09	0.04
23	0.982	0.058	-	-	0.09	0.05
24	0.975	-0.031	-	-	0.42	0.20
25	0.972	-0.074	-	-	0.42	0.20
26	0.958	0.200	-	-	0.06	0.03
27	0.956	0.255	-	-	0.06	0.03
28	0.946	0.368	-	-	0.06	0.02
29	0.940	0.467	-	-	0.12	0.07
30	0.937	0.570	-	-	0.20	0.60
31	0.935	0.527	-	-	0.15	0.07
32	0.935	0.522	-	-	0.21	0.10
33	0.935	0.544	0.20	0.10	0.06	0.04

Total:	0.132 (MW)	0.09 (MVar)
--------	------------	-------------

BCBBFS code Output (rounded to 3 decimal places):

Bus NO |U|(pu) $\theta(^{\circ})$

1	1	0
2	0.997	0.013
3	0.985	0.088
4	0.979	0.149
5	0.974	0.21
6	0.959	0.161
7	0.957	-0.013
8	0.953	0.003
9	0.95	-0.056
10	0.947	-0.106
11	0.946	-0.104
12	0.945	-0.103
13	0.943	-0.164
14	0.942	-0.199
15	0.943	-0.218
16	0.943	-0.23
17	0.945	-0.208
18	0.946	-0.203
19	0.997	0.003
20	0.994	-0.056
21	0.994	-0.075
22	0.993	-0.096
23	0.982	0.058
24	0.975	-0.031
25	0.972	-0.074
26	0.958	0.2
27	0.956	0.255
28	0.946	0.368
29	0.94	0.467
30	0.937	0.57
31	0.935	0.527
32	0.935	0.522
33	0.935	0.544

Number of Iterations (outer loop) = 3

Elapsed time is 0.045709 seconds.

Total Power Loss in lines (kW) =131.6967

Total ReactivePower Consumed by Lines (kVAr) =86.3908

Correction

A correction has been made in BCBFS MATLAB code and consequently in the results as follows:

In previous report in Appendix: MATLAB code for BCBFS:” %% Updating Q Values” we had:

```
1. %% Updating Q Values
2. DV=abs(pv_bus_sp(:,2))-abs(v(pv_bus_sp(:,1))); %Calculating DeltaV Matrix
3. DQ=X_s\DV; %Calculating DeltaV Matrix
4. Q(pv_bus_no,1)=Q(pv_bus_no,1)-sign(DV).*DQ; %Updating Q values (pu)
5. for qqi=1:length(pv_bus_no) %Checking Q limits (pu)
6. qq=pv_bus_no(qqi);
7. if(Q(qq,1)<-abs(pv_qm_gen(qq)))
8. Q(qq,1)=-abs(pv_qm_gen(qq));
9. elseif(Q(qq,1)>abs(pv_qm_con(qq)))
10. Q(qq,1)=abs(pv_qm_con(qq));
11. end
12. end
```

In Lines 8 and 10 the generation value of reactive power at bus(bus number:’qq’) is set on DG reactive power generation(/consumption) limit values and it’s wrong so it must be the limit value plus Q_{Load} amount ($Q_{cons_net}=Q_{cons}+Q_{Load}$) therefore this part changed to:

```
1. %% Updating Q Values
2. DV=abs(pv_bus_sp(:,2))-abs(v(pv_bus_sp(:,1))); %Calculating DeltaV Matrix
3. DQ=X_s\DV; %Calculating DeltaV Matrix
4. Q(pv_bus_no,1)=Q(pv_bus_no,1)-sign(DV).*DQ; %Updating Q values (pu)
5. for qqi=1:length(pv_bus_no) %Checking Q limits (pu)
6. qq=pv_bus_no(qqi);
7. if(Q(qq,1)<QLoad(qq)-abs(pv_qm_gen(qq)))
8. Q(qq,1)=QLoad(qq)-abs(pv_qm_gen(qq));
9. elseif(Q(qq,1)>(abs(pv_qm_con(qq))+QLoad(qq)))
10. Q(qq,1)=abs(pv_qm_con(qq))+QLoad(qq);
11. end
12. end
```

*Notice: In MATPOWER default data for 33-bus system (case33) in branch data for branch number 7 (7to8) there is a difference in some references that mentioned as (in “case33mg.m”: in some references 0.71 and 0.23 (see case33bw.m)) and our choice was 0.7114 and 0.2351 for r and x respectively.

Appendix A

Validated MATLAB Code For B/FS:

```
1. clc
2. clear
3. close all
4. %% Reading Loads & Lines Data
5. Load_Data = readmatrix('bus33.xls');           %Reading Load Data (P & Q in kW & kVAr)
6. Line_Data = readmatrix('branch33.xls');        %Reading Line Data (R & X in Ohms)
7. %% Setting Initial Parameters
8. N=10;                                           % N: Maximum Number of Iterations
9. Sb=1;                                           % S_base (MVA)
10. Ub=12.66;                                     % U_base (kV) (line2line)
11. e=0.01;                                       % Epsilon -> Convergence criteria : max(|vnew-vold|)<e
12. %% Evaluating Zbase
13. Zb=(Ub^2)/Sb;                                % Z_base
14. %%
15. br_no=length(Line_Data);                      % Number of Branches
16. bus_no=length(Load_Data);                    % Number of Buses
17. %% Per unit Values
18. R = Line_Data(:,4)./Zb;
19. X = Line_Data(:,5)./Zb;
20. P = ((Load_Data(:,2))./(1000*Sb));           % P in kW & Sb in MVA
21. Q = ((Load_Data(:,3))./(1000*Sb));           % Q in kVAr & Sb in MVA
22. %% Forming Connection Matrix (C(branches,buses))
23. s_buses=Line_Data(:,2);                      %Sending buses
24. r_buses=Line_Data(:,3);                      %Receiving buses
25. C=zeros(br_no,bus_no);
26. for branch=1:br_no
27.     C(branch,s_buses(branch))=-1;           %Sending bus : -1
28.     C(branch,r_buses(branch))=+1;           %Receiving bus : +1
29. end
30. %% Print Values
31. % (1:no)
32. % [(1:br)',C]
33. %% Determining End Nodes
34. endnode=(find(sum(C)==1))';                  %End Buses indexes
35. %% Print Values
36. %endnode
37. %% Determining the Path of each Radius(:Routes from first Bus to each Endnode)
38. h=length(endnode);                          % h= Number of Radiuses
39. g=[0];                                       % EndBuses Path to the first Bus
40. for route_no=1:h
41.     rnode=endnode(route_no);                 %Receiving node
42.     snode=s_buses(r_buses==rnode);           %Sending Node
43.     g(route_no,1)=rnode;
44.     g(route_no,2)=snode;
45.     j=2;
46.     while(snode~=1)
47.         rnode=snode;
48.         g(route_no,j)=rnode;
49.         snode=s_buses(r_buses==rnode);
50.         g(route_no,j+1)=snode;
51.         j=j+1;
52.     end
53. end
54. %% Print Values
55. %% Sorting Radius Matrix Elements
56. gs=g; %gs is sorted form of g
57. for i=1:length(endnode)
58.     rout=sort(nonzeros(g(i,:)));
59.     gs(i,1:length(rout))=rout;
60. end
61. %% print Values
62. %gs;
63. %% Forming Route Matrices for applications
64. gb=g;
65. mr=1;                                       %MainRoute_Row_index in g
66. for i=1:size(gb,1)
67.     if length(nonzeros(gb(i,:)))>length(nonzeros(gb(mr,:)))
68.         mr=i;
```

```

69.     end
70. end
71. temp=gb(1,:);           %Main Route placed in at the 1st row
72. gb(1,:)=gb(mr,:);
73. gb(mr,:)=temp;
74. for i=1:size(gb,1)
75.     for j=1:size(gb,2)
76.         n=gb(i,j);
77.         for ii=((i+1):size(gb,1))
78.             for jj=1:(size(gb,2)-1)
79.                 if gb(ii,jj)==n
80.                     gb(ii,jj+1)=0;
81.                 end
82.             end
83.         end
84.     end
85. end
86. gv=gb;                  %gv matrix will be used for KVL in Forward steps
87. for i=1:length(endnode)
88.     rout=sort(nonzeros(gb(i,:)));
89.     gv(i,1:length(rout))=rout;
90. end
91. sc=zeros(size(gb,1),1);
92. for j=1:size(gb,1)
93.     for i=1:size(gb,1)-1
94.         a=length(nonzeros(gb(i,:)));
95.         b=length(nonzeros(gb(i+1,:)));
96.         if a>b
97.             t=gb(i,:);
98.             gb(i,:)=gb(i+1,:);
99.             gb(i+1,:)=t;
100.        end
101.    end
102. end
103. g=gb;
104. %% initial guess
105. v = ones(bus_no,1);      %Bus Voltages vector Initialization (complex value) (flat initial guess)
106. I = zeros(br_no,1);      %Branches' Current vector Initialization %C:(br,bus) '
107. %% Iteration Loop
108. for ni=1:N
109.     %% Backward Step
110.     vold=v;
111.     LC = conj(complex(P,Q)./v);    %Bus Load Currents vector
112.     for r=1:route_no
113.         for i=1:size(g,2)-1
114.             b=g(r,i);
115.             if b==0
116.                 break;
117.             end
118.             if sum(C(:,b))==1
119.                 I(C(:,b)==1)=LC(b);
120.                 LC(g(r,i+1))=LC(g(r,i+1))+LC(b);
121.             else
122.                 if g(r,i+1)==1
123.                     I(C(:,b)==1)=LC(b);
124.                     break;
125.                 else
126.                     I(C(:,b)==1)=LC(b);
127.                     if g(r,i+1)~=0
128.                         LC(g(r,i+1))=LC(g(r,i+1))+LC(b);
129.                     end
130.                 end
131.             end
132.         end
133.     end
134.     %% Forward Step
135.     for r=1:route_no
136.         for i=1:size(gv,2)-1
137.             if gv(r,i+1)==0
138.                 continue;
139.             end
140.             b= find(C(:,gv(r,i+1))==1);
141.             v(gv(r,i+1))=v(gv(r,i))-complex(R(b),X(b))*I(b);
142.             fprintf("V("+num2str(gv(r,i+1))+")=V("+num2str(gv(r,i))+")-zI("+num2str(b)+")\n");
143.         end

```

```

144.     end
145.     vnew=v;
146.     if max(abs(vnew-vold))<e
147.         fprintf("Algorithm Converged!\nNumber of Iterations="+num2str(ni)+"\n-----\n")
148.         break;
149.     end
150. end

151. %% Print Calculated Values
152. vbp=[abs(v),angle(v).*(180/pi)];
153. vbp2=[(1:bus_no'),abs(v),angle(v).*(180/pi)]; %'
154. h={'Bus NO','|U|(pu)','?(°)'};
155. T = array2table(vbp2,'VariableNames',h);
156. T2 = array2table(round(vbp2,2),'VariableNames',h);
157. %% Print
158. fprintf("Final Voltages:\n")
159. disp(T2)
160. f=figure;
161. t=uitable(f,'data',vbp2,'columnname',h);
162. %% Plot Voltage Profile
163. f2=figure;
164. p=plot(vbp2(:,1),vbp2(:,2),'-b','LineWidth',2);
165. hold on
166. plot([0 33],[0.95 0.95],'-r','LineWidth',1);
167. plot([0 33],[0.9 0.9],'-r','LineWidth',1);
168. hold off
169. xlim([0 33])
170. ylim([0.85 1.02])
171. yaxess=[0.86:0.02:0.95],[0.95:0.01:1.2];
172. yticks(yaxes)
173. xticks(0:33)
174. xlabel("BUS Number")
175. ylabel("V_{Line} pu")
176. grid on
177. %%
178. Ibrpu=[abs(I) angle(I)*180/pi]; % Branches' Currents in pu magnitude and angle '
179. %% Line Consumed Power Calculation
180. PL = R.*(abs(I).^2); % Active Power (pu) Consumed by each Line
181. QL = X.*(abs(I).^2); %Reactive Power (pu) Consumed by each Line
182. PLkW=(PL)*Sb*1000;
183. QLkVAR=(QL)*Sb*1000;
184. PLt=sum(PL)*Sb*1000; %Total kW consumed by lines (total power loss)
185. QLt=sum(QL)*Sb*1000; %Total kVAR consumed by lines
186. %% Print
187. fprintf('-----\n')
188. fprintf("Total Power Loss in lines (kW) =" +num2str(PLt)+'\n');
189. fprintf("Total ReactivePower Consumed by Lines (kVAR) =" +num2str(QLt)+'\n');

```

Appendix B

Validated MATLAB Code for BCBB/FS:

```
1. clc
2. clear
3. close all
4. %% Reading Loads & Lines Data
5. Load_Data = readmatrix('bus33.xls');           %Reading Load Data (P & Q in kW & kVAr)
6. Line_Data = readmatrix('branch33.xls');        %Reading Line Data (R & X in Ohms)
7. DG_Data = readmatrix('DG_BUS.xlsx');
8. %% Setting Initial Parameters
9. N1=10;                                         % N: Maximum Number of Iterations of Inner loop
10. N2=10;                                       % N2: Maximum Number of Iterations
11. Sb=1;                                       % S_base (MVA)
12. Ub=12.66;                                  % U_base (kV) (line2line)
13. e=0.001;                                   % Epsilon -> Convergence criteria : max(||vsp_pu|-|vcal_pu||)<e
14. e1=0.001;                                  % Inner loop Convergence criteria: max(|vsp_old-vcal_new|)<e1
15. %% Processing const.P-const.Q DG Data in Load_Data Matrix
16. pq_dg=find(DG_Data(:,4)==0);
17. Load_Data([DG_Data(pq_dg,1)], [2 3])=Load_Data([DG_Data(pq_dg,1)], [2 3])-DG_Data(pq_dg, [2 3]);
18. %% Processing const.P-const.V DG Data in Load_Data Matrix and Extracting data
19. pv_dg=find(DG_Data(:,4)==1); % pv buses indices in DG_Data
20. Load_Data([DG_Data(pv_dg,1)], [2])=Load_Data([DG_Data(pv_dg,1)], [2])-DG_Data(pv_dg,2);
21. pv_bus_sp=DG_Data(pv_dg, [1 5]); %DG Voltage set points (pu)
22. pv_bus_no=DG_Data(pv_dg,1);
23. pv_bus_maxgen_q=DG_Data(pv_dg, [1 6]); % Maximum Generated Q (pu)
24. pv_bus_maxcon_q=DG_Data(pv_dg, [1 7]); % Maximum Consumed Q (pu)
25. pv_bus_maxgen_q(:,2)=pv_bus_maxgen_q(:,2)./(1000*Sb);
26. pv_bus_maxcon_q(:,2)=pv_bus_maxcon_q(:,2)./(1000*Sb);
27. pv_qm_gen=containers.Map(pv_bus_maxgen_q(:,1),pv_bus_maxgen_q(:,2));
28. pv_qm_con=containers.Map(pv_bus_maxcon_q(:,1),pv_bus_maxcon_q(:,2));
29. %% Evaluating Zbase
30. Zb=(Ub^2)/Sb; % Z_base
31. %%
32. br_no=length(Line_Data); % Number of Branches
33. bus_no=length(Load_Data); % Number of Buses
34. %% Per unit Values
35. R = Line_Data(:,4)./Zb;
36. X = Line_Data(:,5)./Zb;
37. P = ((Load_Data(:,2))./(1000*Sb)); % P in kW & Sb in MVA
38. Q = ((Load_Data(:,3))./(1000*Sb)); % Q in kVAr & Sb in MVA
39. QLoad=Q;
40. %% Forming Connection Matrix (C(branches,buses))
41. s_buses=Line_Data(:,2); %Sending buses
42. r_buses=Line_Data(:,3); %Receiving buses
43. C=zeros(br_no,bus_no);
44. for branch=1:br_no
45.     C(branch,s_buses(branch))=-1; %Sending bus : -1
46.     C(branch,r_buses(branch))=+1; %Receiving bus : +1
47. end
48. %% Forming Reactance Sensitivity Matrix
49. s=Line_Data(:,2); % Sending Buses
50. r=Line_Data(:,3); % Receiving Buses
51. wx=[Line_Data(:,1),X]; % line number line X(pu)
52. NG=graph(s,r,wx(:,2));
53. figure
54.
55. gp=plot(NG, 'EdgeLabel',wx(:,1), 'Layout', 'layered', 'Direction', 'right', 'LineWidth', 4, 'EdgeColor', 'c',
'MarkerSize', 8, 'Marker', 'o', 'NodeFontSize', 14, 'EdgeFontSize', 10);
55. highlight(gp,pv_bus_no, 'Marker', 's', 'MarkerSize', 12)
```

```

56. X_s=zeros(length(pv_bus_no));
57. pv_path_bus=zeros(length(pv_dg),1);
58. pv_d_s=zeros(length(pv_dg),1);
59. pv_path_line=zeros(length(pv_dg),1);
60. for i=1:length(pv_dg)
61.     [a,b,c] = shortestpath(NG,1,pv_bus_no(i),'Method','unweighted');
62.     pv_path_bus(i,1:length(a))=a;
63.     pv_d_s(i)=b;
64.     pv_path_line(i,1:length(c))=c;
65.     singlepath=nonzeros(pv_path_line(i,:));
66.     X_s(i,i)=sum(wx(singlepath,2));
67.     colora=['c','y','r','b','g','m'];
68.     highlight(gp,a,'EdgeColor',colora(mode(i,4)+1))
69. end
70. for i=1:length(pv_bus_no)
71.     for j=i+1:length(pv_bus_no)
72.         fprintf("i="+num2str(i)+' '+'j='+num2str(j)+'\n');
73.         temp=pv_path_line(i,:)-pv_path_line(j,:);
74.         for ii=1:length(temp)
75.             if temp(ii)~=0
76.                 break;
77.             end
78.         end
79.         ii=ii-1;
80.         commonpath=pv_path_line(i,1:ii);
81.         X_s(i,j)=sum(wx(commonpath,2));
82.         X_s(j,i)=X_s(i,j);
83.     end
84. end
85. %% Determining the Path of each Radius(:Routes from first Bus to each Endnode)
86. h=length(endnode); % h= Number of Radiuses
87. g=[0]; % EndBuses Path
to the first Bus
88. for route_no=1:h
89.     rnode=endnode(route_no); % Recieving node
90.     snode=s_buses(r_buses==rnode); %Sending Node
91.     g(route_no,1)=rnode;
92.     g(route_no,2)=snode;
93.     j=2;
94.     while(snode~=1)
95.         rnode=snode;
96.         g(route_no,j)=rnode;
97.         snode=s_buses(r_buses==rnode);
98.         g(route_no,j+1)=snode;
99.         j=j+1;
100.    end
101. end
102. %% Sorting Radius Matrix Elements
103. gs=g; %gs is sorted form of g
104. for i=1:length(endnode)
105.     rout=sort(nonzeros(g(i,:)));
106.     gs(i,1:length(rout))=rout;
107. end
108. %% Forming Route Matrices for applications
109. gb=g;
110. mr=1; %MainRoute_Row_index in g
111. for i=1:size(gb,1)
112.     if length(nonzeros(gb(i,:)))>length(nonzeros(gb(mr,:)))
113.         mr=i;
114.     end
115. end
116. temp=gb(1,:); %Main Route placed in at the 1st row
117. gb(1,:)=gb(mr,:);

```

```

118. gb(mr,:)=temp;
119. for i=1:size(gb,1)
120.     for j=1:size(gb,2)
121.         n=gb(i,j);
122.         for ii=((i+1):size(gb,1))
123.             for jj=1:(size(gb,2)-1)
124.                 if gb(ii,jj)==n
125.                     gb(ii,jj+1)=0;
126.                 end
127.             end
128.         end
129.     end
130. end
131. gv=gb; %gv matrix will be used for KVL in Forward steps
132. for i=1:length(endnode)
133.     rout=sort(nonzeros(gb(i,:)));
134.     gv(i,1:length(rout))=rout;
135. end
136. sc=zeros(size(gb,1),1);
137. for j=1:size(gb,1)
138.     for i=1:size(gb,1)-1
139.         a=length(nonzeros(gb(i,:)));
140.         b=length(nonzeros(gb(i+1,:)));
141.         if a>b
142.             t=gb(i,:);
143.             gb(i,:)=gb(i+1,:);
144.             gb(i+1,:)=t;
145.         end
146.     end
147. end
148. g=gb;
149. %% initial guess
150. v = ones(bus_no,1); %Bus Voltages vector Initialization (complex value) (flat initial guess)
151. v(pv_bus_sp(:,1))=pv_bus_sp(:,2); % PV Buses initial guess according to their set points
152. I = zeros(br_no,1); %Branches' Current vector Initialization % C(br,bus)'

153. %% outer Iteration Loop
154. for oc=1:N2
155.     v_pv_old_abs=abs(v(pv_bus_no));
156.     %% Inner Iteration Loop (BFS)
157.     for ni=1:N1
158.         %% Backward Step
159.         vold=v;
160.         LC = conj(complex(P,Q)./v); %Bus Load Currents vector
161.         for r=1:route_no
162.             for i=1:size(g,2)-1
163.                 b=g(r,i);
164.                 if b==0
165.                     break;
166.                 end
167.                 if sum(C(:,b))==1
168.                     I(C(:,b)==1)=LC(b);
169.                     LC(g(r,i+1))=LC(g(r,i+1))+LC(b);
170.                 else
171.                     if g(r,i+1)==1
172.                         I(C(:,b)==1)=LC(b);
173.                         break;
174.                     else
175.                         I(C(:,b)==1)=LC(b);
176.                         if g(r,i+1)~=0
177.                             LC(g(r,i+1))=LC(g(r,i+1))+LC(b);
178.                         end
179.                     end

```

```

180.         end
181.     end
182. end
183. %% Forward Step
184. for r=1:route_no
185.     for i=1:size(gv,2)-1
186.         if gv(r,i+1)==0
187.             continue;
188.         end
189.         b= find(C(:,gv(r,i+1))==1);
190.         v(gv(r,i+1))=v(gv(r,i))-complex(R(b),X(b))*I(b);
191.         %fprintf("V("+num2str(gv(r,i+1))+")=V("+num2str(gv(r,i))+")-
zI("+num2str(b)+")\n");
192.     end
193. end
194. vnew=v;
195. if max(abs(vnew-vold))<e1
196.     %fprintf("Algorithm Converged!\nNumber of Iterations="+num2str(ni)+"\n---\n")
197.     break;
198. end
199. end
200. v_pv_new_abs=abs(v(pv_bus_no));
201. if max(abs(v_pv_new_abs-v_pv_old_abs))<e
202.     fprintf("Algorithm Converged!\nNumber of Iterations(outer loop)="+num2str(oc)+"\n--\n")
203.     break;
204. end
205. %% Updating Q Values
206. DV=abs(pv_bus_sp(:,2))-abs(v(pv_bus_sp(:,1))); %Calculating DeltaV Matrix
207. DQ=X_s\DV; %Calculating DeltaV Matrix
208. Q(pv_bus_no,1)=Q(pv_bus_no,1)-sign(DV).*DQ; %Updating Q values (pu)
209. for qq=1:length(pv_bus_no) %Checking Q limits (pu)
210.     qq=pv_bus_no(qq);
211.     if(Q(qq,1)<QLoad(qq)-abs(pv_qm_gen(qq)))
212.         Q(qq,1)=QLoad(qq)-abs(pv_qm_gen(qq));
213.     elseif(Q(qq,1)>(abs(pv_qm_con(qq))+QLoad(qq)))
214.         Q(qq,1)=abs(pv_qm_con(qq))+(QLoad(qq));
215.     end
216. end
217. end
218. %% Print Calculated Values
219. vbp=[abs(v),angle(v).*(180/pi)]; 220. vbp2=[((1:bus_no)'),abs(v),angle(v).*(180/pi)]; %'
221. h={'Bus NO','|U|(pu)','?(°)'};
222. T = array2table(vbp2,'VariableNames',h);
223. T2 = array2table(round(vbp2,2),'VariableNames',h);
224. %% Print
225. fprintf("Final Voltages:\n")
226. disp(T2)
227. f=figure;
228. t=uitable(f,'data',vbp2,'columnname',h);
229. %% Plot Voltage Profile
230. f2=figure;
231. p=plot(vbp2(:,1),vbp2(:,2),'-b','LineWidth',2);
232. hold on
233. plot([0 bus_no],[0.95 0.95],'-r','LineWidth',1);
234. plot([0 bus_no],[0.9 0.9],'-r','LineWidth',1);
235. hold off
236. xlim([0 bus_no])
237. ylim([0.85 1.02])
238. yaxes=[[0.86:0.02:0.95],[0.95:0.01:1.2]];
239. yticks(yaxes)
240. xticks(0:bus_no)
241. xlabel("BUS Number")

```

```

242. ylabel("V_{Line} pu")
243. grid on
244. %%

245. Ibrpu=[abs(I) angle(I)*180/pi];    % Branches' Currents in pu magnitude and angle '
246. %% Line Consumed Power Calculation
247. PL = R.*(abs(I).^2);                % Active Power (pu) Consumed by each Line
248. QL = X.*(abs(I).^2);                % Reactive Power (pu) Consumed by each Line
249. PLkW=(PL)*Sb*1000;
250. QLkVar=(QL)*Sb*1000;
251. PLt=sum(PL)*Sb*1000;                %Total kW consumed by lines (total power loss)
252. QLt=sum(QL)*Sb*1000;                %Total kVar consumed by lines
253. %% Print
254. fprintf('-----\n')
255. fprintf("Total Power Loss in lines (kW) =" + num2str(PLt) + '\n');
256. fprintf("Total ReactivePower Consumed by Lines (kVar) =" + num2str(QLt) + '\n');

```

Thank you for your time.