Massachusetts Institute of Technology

Department of Electrical Engineering and Computer Science

6.691 Seminar in Advanced Electric Power Systems

Problem Set 2 Solutions March 8, 2006

This problem set involves the example power system which was presented in class on class on February 20, reproduced in Figure 1.

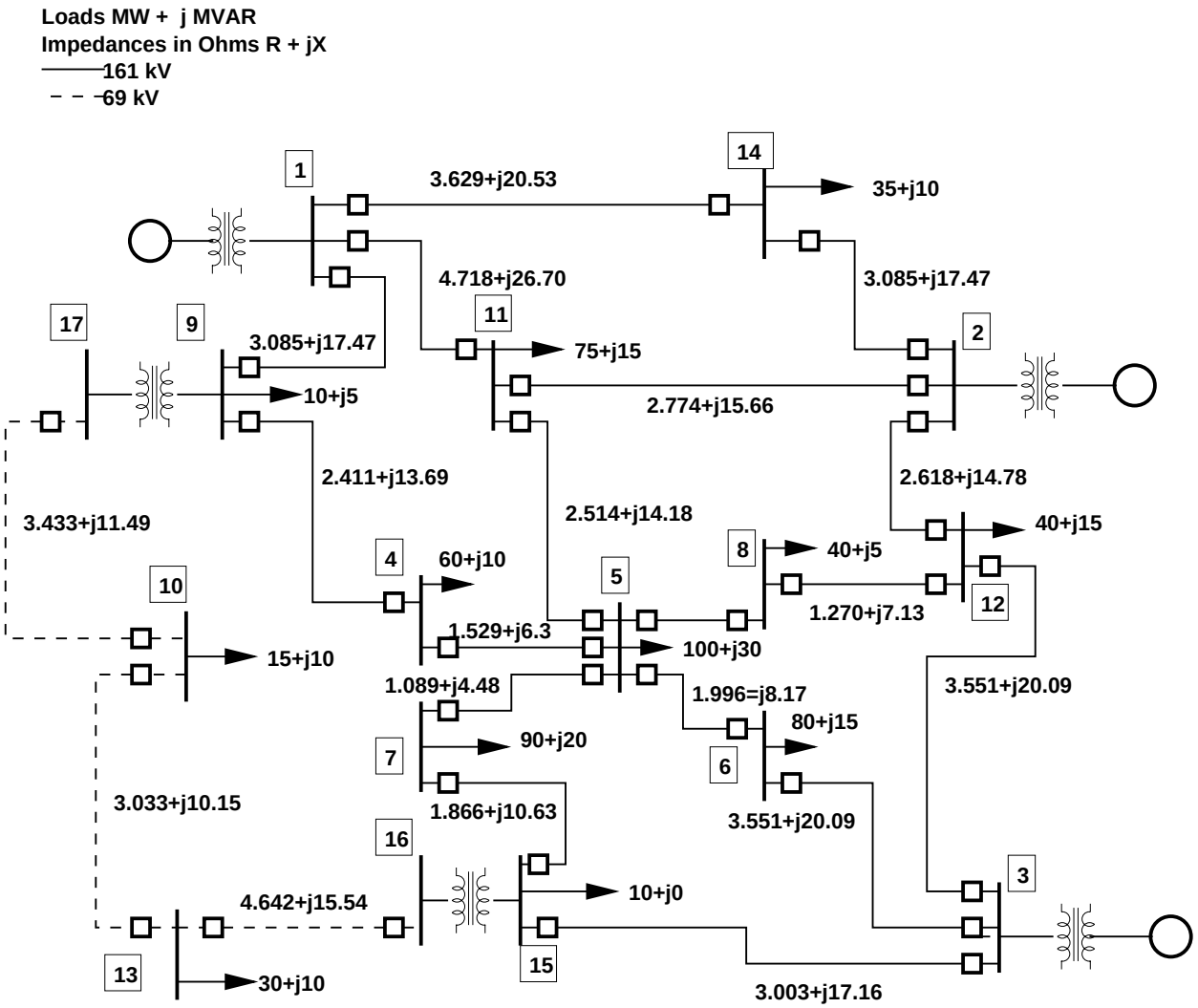

Figure 1: Example Power System

The first part of the problem consists of translating the impedances and power levels into per-unit. This is not difficult: it is necessary to divide the impedances by the base value. Here is a schedule of that calculation.

```
6.961  Problem  Set  2  Setup
Line  Impedances
Line   Buses   Ohms                      Per-Unit
  1    1  14      3.629  +  j  20.530    0.0140  +  j   0.0792
  2    1  11      4.718  +  j  26.700    0.0182  +  j   0.1030
  3   14   2      3.085  +  j  17.470    0.0119  +  j   0.0674
  4   11   2      2.774  +  j  15.660    0.0107  +  j   0.0604
  5    1   9      3.085  +  j  17.470    0.0119  +  j   0.0674
  6    9   4      2.411  +  j  13.690    0.0093  +  j   0.0528
  7   11   5      2.514  +  j  14.180    0.0097  +  j   0.0547
  8    2  12      2.618  +  j  14.780    0.0101  +  j   0.0570
  9    5   8      1.996  +  j8.170       0.0077  +  j   0.0315
 10    5   4      1.529  +  j6.300       0.0059  +  j   0.0243
 11    5   7      1.089  +  j4.460       0.0042  +  j   0.0172
 12    5   6      1.970  +  j8.090       0.0076  +  j   0.0312
 13   12   3      3.551  +  j  20.090    0.0137  +  j   0.0775
 14    6   3      3.551  +  j  20.090    0.0137  +  j   0.0775
 15    7  15      1.886  +  j  10.630    0.0073  +  j   0.0410
 16    3  15      3.003  +  j  17.160    0.0116  +  j   0.0662
 17   17  10      3.433  +  j  11.490    0.0721  +  j   0.2413
 18   10  13      3.033  +  j  10.150    0.0637  +  j   0.2132
 19   13  16      4.462  +  j  15.540    0.0937  +  j   0.3264
 20    8  12      1.270  +  j7.130       0.0049  +  j   0.0275
 21    9  17                             0.0000  +  j   0.0533
 22   15  16                             0.0000  +  j   0.0533
```

Note that we have included two 'lines' (21 and 22) that represent the per-unit impedance of the two transformers. Per the instructions, a re-drawn network diagram (one-line) is shown in Figure 2, with per-unit impedances, real and reactive power.

The scripts that do the work in this problem set will be appended to the end of this solution set.

The load flow problem can be done in a variety of ways, as described in class. Both Gauss-Seidel and Newton-Raphson methods are detailed in the scripts appended. The results below were generated from the decoupled Newton-Raphson script (the last one appended) The voltages at the various buses are:

```
Bus  Voltages  are:
Bus      Per-Unit  Angle  (Radians)     (Degrees)
  1     1.0000        0.0000           0.00
  2     1.0020        0.0072           0.41
  3     0.9896       -0.0160          -0.91
  4     0.9545       -0.0918          -5.26
  5     0.9548       -0.0909          -5.21
  6     0.9574       -0.0876          -5.02
  7     0.9516       -0.0972          -5.57
  8     0.9640       -0.0732          -4.19
```

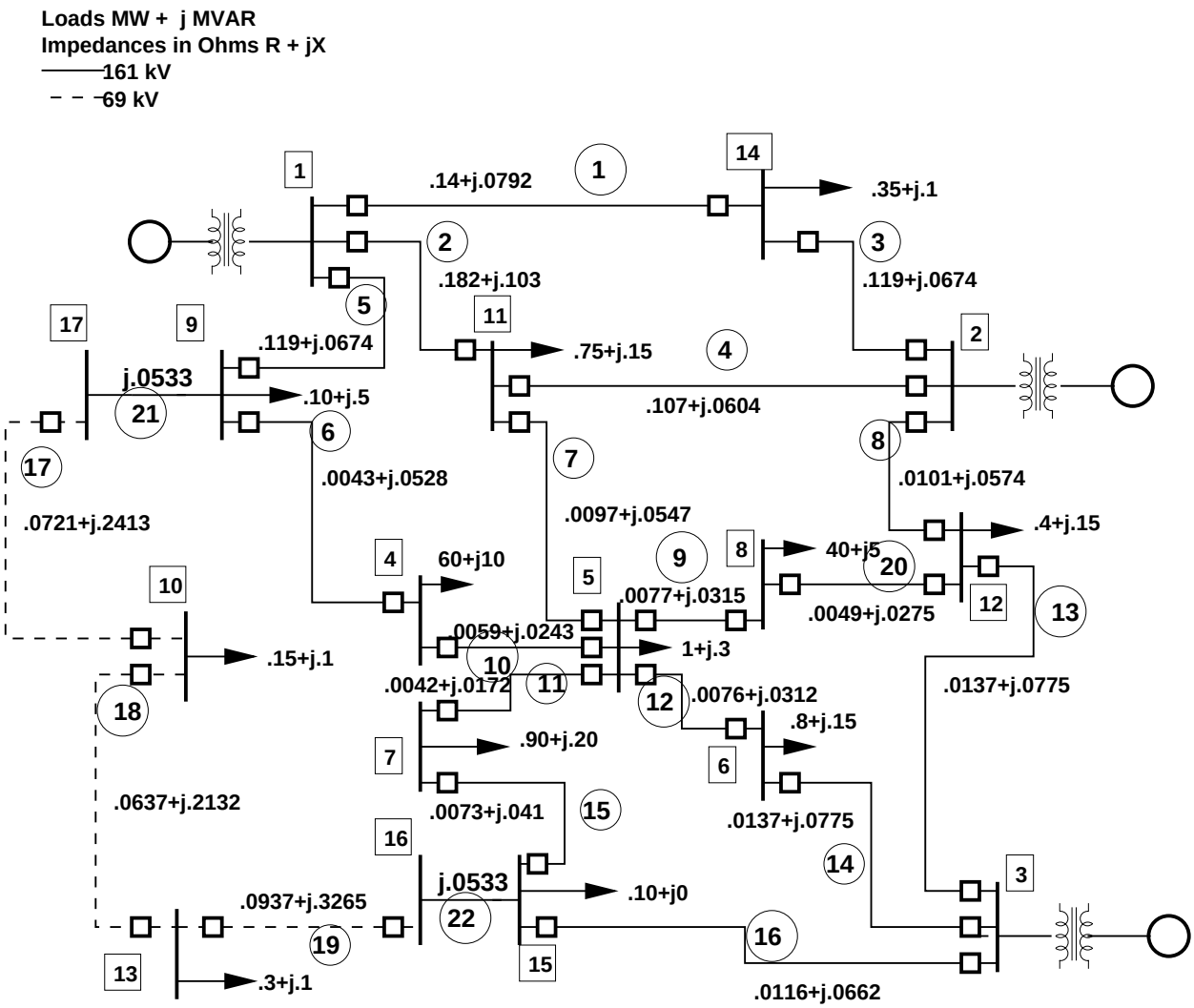

Figure 2: Per-Unit Example Power System

9	0.9657	-0.0603	-3.46
10	0.9073	-0.1325	-7.59
11	0.9745	-0.0505	-2.90
12	0.9749	-0.0463	-2.65
13	0.8981	-0.1556	-8.91
14	0.9951	-0.0083	-0.47
15	0.9609	-0.0734	-4.21
16	0.9547	-0.0859	-4.92
17	0.9582	-0.0746	-4.28

And the associated line flows are (note they are translated back into amperes):

```
Line  Current  Flows
Line     Amperes
```

1	42.8
2	192.1
3	88.6
4	370.1
5	358.8
6	215.6
7	281.7
8	367.8
9	213.7
10	12.9
11	136.1
12	46.4
13	151.4
14	349.7
15	211.4
16	335.7
17	246.3
18	85.6
19	211.5
20	362.8
21	105.5
22	90.6

1 Appendix 1: Starting Script (per-unit values)

```
% 6.691 Spring 2006 Solutions to Problem Set 2

% first, here are the line impedances
Z_l = [3.629+j*20.53 4.718+j*26.7 3.085+j*17.47 2.774+j*15.66 3.085+j*17.47...
       2.411+j*13.69 2.514+j*14.18 2.618+j*14.78...
       1.996+j*8.17 1.529+j*6.30 1.089+j*4.46 1.97+j*8.09 3.551+j*20.09...
       3.551+j*20.09 1.886+j*10.63 3.003+j*17.16 3.433+j*11.49 3.033+j*10.15...
       4.462+j*15.54 1.270+j*7.13];
Pb = 100e6;     % we are going to use this base power
Vb1 = 161e3;    % and these base voltages
Vb2 = 69e3;
Zb1 = Vb1^2/Pb;
Zb2 = Vb2^2/Pb;
% and here are the connections for each line
C_l = [1 14;
       1 11;
       14 2;
       11 2;
       1 9;
       9 4;
       11 5;
       2 12;
       5 8;
       5 4;
       5 7;
       5 6;
       12 3;
       6 3;
       7 15;
       3 15;
       17 10;
       10 13;
       13 16;
       8 12;
       9 17;
       15 16];

% here we calculate per-unit impedances: lines 1-16 and 20 are on base 1
% and lines 17 to 19 are on base 2
z_l(1:16) = Z_l(1:16) ./ Zb1;
z_l(20) = Z_l(20) / Zb1;
z_l(17:19) = Z_l(17:19) ./ Zb2;
```

```
% the last two lines are transformers
z_line = [z_l j*.08/1.5 j*.08/1.5];

fprintf('6.961 Problem Set 2 Setup\n')
fprintf('Line Impedances \n')
fprintf('Line  Buses            Ohms                    Per-Unit\n')
for i = 1:length(Z_l)
fprintf('%3.0f %3.0f %3.0f %10.3f + j %10.3f %10.4f + j %10.4f\n',...
        i, C_l(i, 1), C_l(i, 2), real(Z_l(i)), imag(Z_l(i)),...
        real(z_l(i)), imag(z_l(i)))
end
for i = 21:22
fprintf('%3.0f  %3.0f %3.0f                          %10.4f + j %10.4f\n',...
        i, C_l(i, 1), C_l(i, 2), real(z_line(i)), imag(z_line(i)))
 end
```

2 Gauss-Seidel Solution Script

```
% Solution to Problem Set 2
% Gauss-Seidel solution

tol=.0001;

% first, here are the line impedances
Z_l = [3.629+j*20.53 4.718+j*26.7 3.085+j*17.47 2.774+j*15.66 3.085+j*17.47...
       2.411+j*13.69 2.514+j*14.18 2.618+j*14.78...
       1.996+j*8.17 1.529+j*6.30 1.089+j*4.46 1.97+j*8.09 3.551+j*20.09...
       3.551+j*20.09 1.886+j*10.63 3.003+j*17.16 3.433+j*11.49 3.033+j*10.15...
       4.462+j*15.54 1.270+j*7.13];
% and here are the connections for each line
C_l = [1 14;
       1 11;
       14 2;
       11 2;
       1 9;
       9 4;
       11 5;
       2 12;
       5 8;
       5 4;
       5 7;
       5 6;
       12 3;
       6 3;
       7 15;
       3 15;
       17 10;
       10 13;
       13 16;
       8 12;
       9 17;
       15 16];

Pb = 100e6;     % we are going to use this base power
Vb1 = 161e3;    % and these base voltages
Vb2 = 69e3;

Zb1 = Vb1^2/Pb;
Zb2 = Vb2^2/Pb;
Ib1 = Pb/(sqrt(3)*Vb1);
Ib2 = Pb/(sqrt(3)*Vb2);
```

```
z_l = zeros(size(Z_l));
% unfortunately we have to cobble together the impedance vector
z_l(1:16) = Z_l(1:16) ./ Zb1;
z_l(20) = Z_l(20) / Zb1;
z_l(17:19) = Z_l(17:19) ./ Zb2;

% the last two lines are transformers
z_line = [z_l j*.08/1.5 j*.08/1.5];

fprintf('Line Impedances \n')
fprintf('Buses          Ohms                  Per-Unit\n')
for i = 1:length(Z_l)
fprintf('%3.0f %3.0f %10.3f + j %10.3f %10.4f + j %10.4f\n',...
        C_l(i, 1), C_l(i, 2), real(Z_l(i)), imag(Z_l(i)),...
        real(z_l(i)), imag(z_l(i)))
end

Nl = length(z_line);       % number of lines
Nb = 17;                % number of buses
% Now construct the node-incidence matrix:
NI = zeros(Nb, Nl);        % to start: now we fill it in
for i = 1:Nl
  NI(C_l(i, 1), i) = 1;
  NI(C_l(i, 2), i) = -1;
end

y_line = zeros(Nl);
% now the line admittance matrix is:
for i = 1:Nl
y_line(i, i) = 1 / z_line(i);
end

% and the bus admittance matrix is:
y_bus = NI * y_line * NI';

% Here are the bus power flows:

S_bus = [2.2+j*.7;
        2.2+j*.7;
        2.2+j*.7;
        -.6-j*.1;
        -1-j*.3;
        -.8-j*.15;
        -.9-j*.2;
```

```
        -.4-j*.05;
        -.1-j*.05;
        -.15-j*.1;
        -.75-j*.15;
        -.4-j*.15;
        -.3+j*.1;
        -.35-j*.1;
        -.1;
        0;  0];

P_r  =  real(S_bus);
Q_r  =  imag(S_bus);

%  now  here  is  an  initial  guess  about  voltages:
V  =  ones(Nb,  1);
%Vn  =  ones(Nb,   1);
%  now  we  go  into  a  loop:  check  error  below

n_iter  =  0;
not_done  =  1;
while  (not_done  ==  1)

    for  i  =  2:Nb             %  bus  1  is  the  swing  bus:  don't  modify  it!
        psum  =  0;
        for  k  =  1:Nb
            if  k  ~=  i,
                psum  =  psum+y_bus(i,  k)  *  V(k);
            end
        end
        V(i)  =  ((P_r(i)  - j*Q_r(i))/conj(V(i))  - psum)/y_bus(i,  i);
    end
%     V  =  Vn;
%  ok,  having  run  through  the  correction  one  cycle,  check  errors

P  =  zeros(Nb,  1);
Q  =  zeros(Nb,  1);
for  i  =  1:Nb,
    PQ  =  0;
    for  k  =  1:Nb,
        PQ  =  PQ  +  V(i)  *  conj(V(k)  *  y_bus(i,  k));
    end
    P(i)  =  real(PQ);
    Q(i)  =  imag(PQ);
end
```

```
% now find real and reactive power
PE = P-P_r;
QE = Q-Q_r;
E = [PE(2:Nb);QE(2:Nb)]; % note that bus 1 is the 'swing bus'
SE = sum(E .^2);         % this is absolute, per-unit error
if SE < tol
  not_done = 0;
end

n_iter = n_iter + 1;

%fprintf('Iteration %8.0f Error = %g\n',n_iter, SE);
%pause
end

fprintf('That took %10.0f iterations\n', n_iter)

% now generate per-unit line flows

v_bus = V;
v_line = NI' * v_bus;
i_line = y_line * v_line;

I_line = zeros(size(i_line));
I_line(1:16) = Ib1 .* i_line(1:16);
I_line(17:19) = Ib2 .* i_line(17:19);
I_line(20:22) = Ib1 .* i_line(20:22);

% now generate a report:

fprintf('Bus Voltages are:\n')
fprintf('Bus      Per-Unit    Angle (degrees)\n');
for i = 1:Nb
  fprintf('%3.0f     %3.4f         %6.2f\n',i, abs(V(i)), (180/pi)*angle(V(i)));
end

fprintf('Line Current Flows\n')
fprintf('Line     Amperes\n')
for i = 1:22
  fprintf('%3.0f    %3.1f\n',i, abs(I_line(i)));
end
```

3 Newton-Raphson Solution Script

```
% Solution to Problem Set 2
% Newton-Raphson Solution...

tol=.000001;

% first, here are the line impedances
Z_l = [3.629+j*20.53 4.718+j*26.7 3.085+j*17.47 2.774+j*15.66 3.085+j*17.47...
       2.411+j*13.69 2.514+j*14.18 2.618+j*14.78...
       1.996+j*8.17 1.529+j*6.30 1.089+j*4.46 1.97+j*8.09 3.551+j*20.09...
       3.551+j*20.09 1.886+j*10.63 3.003+j*17.16 3.433+j*11.49 3.033+j*10.15...
       4.462+j*15.54 1.270+j*7.13];
% and here are the connections for each line
C_l = [1 14;
       1 11;
       14 2;
       11 2;
       1 9;
       9 4;
       11 5;
       2 12;
       5 8;
       5 4;
       5 7;
       5 6;
       12 3;
       6 3;
       7 15;
       3 15;
       17 10;
       10 13;
       13 16;
       8 12;
       9 17;
       15 16];

Pb = 100e6;     % we are going to use this base power
Vb1 = 161e3;    % and these base voltages
Vb2 = 69e3;

Zb1 = Vb1^2/Pb;
Zb2 = Vb2^2/Pb;
Ib1 = Pb/(sqrt(3)*Vb1);
Ib2 = Pb/(sqrt(3)*Vb2);
```

```
z_l = zeros(size(Z_l));
% unfortunately we have to cobble together the impedance vector
z_l(1:16) = Z_l(1:16) ./ Zb1;
z_l(20) = Z_l(20) / Zb1;
z_l(17:19) = Z_l(17:19) ./ Zb2;

% the last two lines are transformers
z_line = [z_l j*.08/1.5 j*.08/1.5];

% now we can assemble line capacity (in per-unit)
Icap = zeros(size(Z_l));
Icap(1:9) = 907 / Ib1;
Icap(10) = 726/Ib1;
Icap(11) = 907/Ib1;
Icap(12:14) = 726/Ib1;
Icap(15:16) = 907/Ib1;
Icap(17)=907/Ib2;
Icap(18:19) = 659/Ib2;
Icap(20) = 659/Ib1;
Icap(21:22) = 1.5;

Nl = length(z_line);        % number of lines
Nb = 17;                 % number of buses
% Now construct the node-incidence matrix:
NI = zeros(Nb, Nl);         % to start: now we fill it in
for i = 1:Nl
  NI(C_l(i, 1), i) = 1;
  NI(C_l(i, 2), i) = -1;
end

y_line = zeros(Nl);
% now the line admittance matrix is:
for i = 1:Nl
y_line(i, i) = 1 / z_line(i);
end

% and the bus admittance matrix is:
y_bus = NI * y_line * NI';

% Here are the bus power flows:

S_bus = [2.2+j*.7;          % actually, bus 1 is swing bus
         2.2+j*.7;          % bus2
         2.2+j*.7;          % bus3
```

```
        -.6-j*.1;           %  bus  4:  note  this  one  is  loaded
        -1-j*.3;            %  bus  5
        -.8-j*.15;          %  bus  6
        -.9-j*.2;           %  bus  7
        -.4-j*.05;          %  bus  8
        -.1-j*.05;          %  bus  9
        -.15-j*.1;          %  bus  10
        -.75-j*.15;         %  bus  11
        -.4-j*.15;          %  bus  12
        -.3-j*.1;           %  bus  13
        -.35-j*.1;          %  bus  14
        -.1;                %  bus  15
        0;  0];             %  buses  16  and  17  are  unloaded

G  =  real(y_bus);
B  =  imag(y_bus);

P_r  =  real(S_bus);
Q_r  =  imag(S_bus);

%  now  here  is  an  initial  guess  about  voltages:
V  =  ones(Nb,  1);
th  =  zeros(Nb,  1);

%  now  we  go  into  a  loop
n_iter  =  0;
not_done  =  1;
while  (not_done  ==  1)
P  =  zeros(Nb,  1);
Q  =  zeros(Nb,  1);
for  i  =  1:Nb,
    for  k  =  1:Nb,
        P(i)  =  P(i)  +  V(i)*V(k)*(G(i,  k)*cos(th(i)-th(k))+B(i,k)*sin(th(i)-th(k)));
        Q(i)  =  Q(i)  +  V(i)*V(k)*(G(i,  k)*sin(th(i)-th(k))-B(i,k)*cos(th(i)-th(k)));
    end
end

X  =  [th(2:Nb);  V(2:Nb)];                 %  to  be  found
J11  =  zeros(Nb-1);              %  components  of  the  Jacobian
J12  =  zeros(Nb-1);
J21  =  zeros(Nb-1);
J22  =  zeros(Nb-1);

for  i  =  2:Nb
 for  k  =  2:Nb
```

```
  ii=  i-1;
  kk  =  k-1;
  if  k  ~=  i
   J11(ii,  kk)  =  V(i)*V(k)*(G(i,k)*sin(th(i)-th(k))-B(i,k)*cos(th(i)-th(k)));
   J12(ii,  kk)  =  V(i)*(G(i,k)*cos(th(i)-th(k))+  B(i,k)*sin(th(i)-th(k)));
   J21(ii,  kk)  =  -V(i)*V(k)*(G(i,k)*cos(th(i)-th(k))+B(i,k)*sin(th(i)-th(k)));
   J22(ii,  kk)  =  V(i)*(G(i,k)*sin(th(i)-th(k))-B(i,k)*cos(th(i)-th(k)));
  else
   J11(ii,  ii)  =  -V(i)^2*B(i,i)-Q(i);
   J12(ii,  ii)  =  P(i)/V(i)+V(i)*G(i,i);
   J21(ii,  ii)  =  P(i)-V(i)^2*G(i,i);
   J22(ii,  ii)  =  Q(i)/V(i)-V(i)*B(i,i);
%    pause
  end   %  end  of  if  k~=  i
 end    %  end  of  index  k  loop
end     %  end  of  index  i  loop:  Jacobian  is  constructed

J  =  [J11  J12;J21  J22];

%  now  find  real  and  reactive  power
PE  =  P-P_r;
QE  =  Q-Q_r;
E  =  [PE(2:Nb);QE(2:Nb)];
SE  =  sum(E  .^2);        %  this  is  absolute,  per-unit  error
if  SE  <  tol
  not_done  =  0;
end

X  =  X  - inv(J)*E;
th  =[0;X(1:Nb-1)];            %  angle  in  radians
V  =  [1;  X(Nb:2*Nb-2)];      %  voltage  in  per-unit
A  =  (180/pi)  .*  th;        %  angle  in  degrees

n_iter  =  n_iter  +  1;

fprintf('Error  =  %g\n',  SE);
%pause
end

fprintf('That  took  %10.0f  iterations\n',  n_iter)

%  now  generate  per-unit  line  flows

v_bus  =  V  .*  exp(j  .*  th);
v_line  =  NI'  *  v_bus;
```

```
i_line = y_line * v_line;

I_line = zeros(size(i_line));
I_line(1:16) = Ib1 .* i_line(1:16);
I_line(17:19) = Ib2 .* i_line(17:19);
I_line(20:22) = Ib1 .* i_line(20:22);

% now generate a report:

fprintf('Bus Voltages are:\n')
fprintf('Bus    Per-Unit\n');
for i = 1:Nb
  fprintf('%3.0f   %8.4f   %8.4f\n',i, V(i), A(i));
end

fprintf('Line Current Flows\n')
fprintf('Line     per-unit   Rel to capacity\n')
for i = 1:22
  fprintf('%3.0f    %5.3f    %5.3f\n',i, abs(i_line(i)), abs(i_line(i))/Icap(i));
end
```

4 Decoupled Newton-Raphson Solution Script

```
% Solution to Problem Set 2
% Newton-Raphson Solution..Decoupled.

tol=.000001;

% first, here are the line impedances
Z_l = [3.629+j*20.53 4.718+j*26.7 3.085+j*17.47 2.774+j*15.66 3.085+j*17.47...
       2.411+j*13.69 2.514+j*14.18 2.618+j*14.78...
       1.996+j*8.17 1.529+j*6.30 1.089+j*4.46 1.97+j*8.09 3.551+j*20.09...
       3.551+j*20.09 1.886+j*10.63 3.003+j*17.16 3.433+j*11.49 3.033+j*10.15...
       4.462+j*15.54 1.270+j*7.13];
% and here are the connections for each line
C_l = [1 14;
       1 11;
       14 2;
       11 2;
       1 9;
       9 4;
       11 5;
       2 12;
       5 8;
       5 4;
       5 7;
       5 6;
       12 3;
       6 3;
       7 15;
       3 15;
       17 10;
       10 13;
       13 16;
       8 12;
       9 17;
       15 16];

Pb = 100e6;     % we are going to use this base power
Vb1 = 161e3;    % and these base voltages
Vb2 = 69e3;

Zb1 = Vb1^2/Pb;
Zb2 = Vb2^2/Pb;
Ib1 = Pb/(sqrt(3)*Vb1);
Ib2 = Pb/(sqrt(3)*Vb2);
```

```
z_l = zeros(size(Z_l));
% unfortunately we have to cobble together the impedance vector
z_l(1:16) = Z_l(1:16) ./ Zb1;
z_l(20) = Z_l(20) / Zb1;
z_l(17:19) = Z_l(17:19) ./ Zb2;

% the last two lines are transformers
z_line = [z_l j*.08/1.5 j*.08/1.5];

fprintf('Line Impedances \n')
fprintf('Buses          Ohms                  Per-Unit\n')
for i = 1:length(Z_l)
fprintf('%3.0f %3.0f %10.3f + j %10.3f %10.4f + j %10.4f\n',...
        C_l(i, 1), C_l(i, 2), real(Z_l(i)), imag(Z_l(i)),...
        real(z_l(i)), imag(z_l(i)))
end

Nl = length(z_line);       % number of lines
Nb = 17;                % number of buses
% Now construct the node-incidence matrix:
NI = zeros(Nb, Nl);        % to start: now we fill it in
for i = 1:Nl
    if C_l(i, 1) ~=0,
        NI(C_l(i, 1), i) = 1;
        NI(C_l(i, 2), i) = -1;
    end
end

y_line = zeros(Nl);
% now the line admittance matrix is:
for i = 1:Nl
y_line(i, i) = 1 / z_line(i);
end

% and the bus admittance matrix is:
y_bus = NI * y_line * NI';

% Here are the bus power flows:

S_bus = [2.2+j*.7;     % but note we are going to ignore this one
        2.2+j*.7;
        2.2+j*.7;
        -.6-j*.1;
        -1-j*.3;
```

```
        -.8-j*.15;
        -.9-j*.2;
        -.4-j*.05;
        -.1-j*.05;
        -.15-j*.1;
        -.75-j*.15;
        -.4-j*.15;
        -.3-j*.1;
        -.35-j*.1;
        -.1;
        0;  0];

G  =  real(y_bus);
B  =  imag(y_bus);

P_r  =  real(S_bus);
Q_r  =  imag(S_bus);

%  now  here  is  an  initial  guess  about  voltages:
V  =  ones(Nb,  1);
th  =  zeros(Nb,  1);

%  now  we  go  into  a  loop
n_iter  =  0;
not_done  =  1;
while  (not_done  ==  1)
P  =  zeros(Nb,  1);
Q  =  zeros(Nb,  1);
for  i  =  1:Nb,
    for  k  =  1:Nb,
        P(i)  =  P(i)  +  V(i)*V(k)*(G(i,  k)*cos(th(i)-th(k))+B(i,k)*sin(th(i)-th(k)));
        Q(i)  =  Q(i)  +  V(i)*V(k)*(G(i,  k)*sin(th(i)-th(k))-B(i,k)*cos(th(i)-th(k)));
    end
end

X  =  [th(2:Nb);  V(2:Nb)];                  %  to  be  found
J11  =  zeros(Nb-1);              %  components  of  the  Jacobian
J12  =  zeros(Nb-1);
J21  =  zeros(Nb-1);
J22  =  zeros(Nb-1);

for  i  =  2:Nb
 for  k  =  2:Nb
  ii=  i-1;
  kk  =  k-1;
```

```
  if  k  ~=  i
   J11(ii,  kk)  =  V(i)*V(k)*(G(i,k)*sin(th(i)-th(k))-B(i,k)*cos(th(i)-th(k)));
   J12(ii,  kk)  =  V(i)*(G(i,k)*cos(th(i)-th(k))+  B(i,k)*sin(th(i)-th(k)));
   J21(ii,  kk)  =  -V(i)*V(k)*(G(i,k)*cos(th(i)-th(k))+B(i,k)*sin(th(i)-th(k)));
   J22(ii,  kk)  =  V(i)*(G(i,k)*sin(th(i)-th(k))-B(i,k)*cos(th(i)-th(k)));
  else
   J11(ii,  ii)  =  -V(i)^2*B(i,i)-Q(i);
   J12(ii,  ii)  =  P(i)/V(i)+V(i)*G(i,i);
   J21(ii,  ii)  =  P(i)-V(i)^2*G(i,i);
   J22(ii,  ii)  =  Q(i)/V(i)-V(i)*B(i,i);
%    pause
  end   %  end  of  if  k~=  i
 end    %  end  of  index  k  loop
end     %  end  of  index  i  loop:  Jacobian  is  constructed

J  =  [J11  zeros(Nb-1);zeros(Nb-1)  J22];

%  now  find  real  and  reactive  power
PE  =  P-P_r;
QE  =  Q-Q_r;
E  =  [PE(2:Nb);QE(2:Nb)];
SE  =  sum(E  .^2);         %  this  is  absolute,  per-unit  error
if  SE  <  tol
  not_done  =  0;
end

X  =  X  - inv(J)*E;
th  =[0;X(1:Nb-1)];
V  =  [1;  X(Nb:2*Nb-2)];

n_iter  =  n_iter  +  1;

fprintf('Error  =  %g\n',  SE);
%pause
end

fprintf('That  took  %10.0f  iterations\n',  n_iter)

%  now  generate  per-unit  line  flows

v_bus  =  V  .*  exp(j  .*  th);
v_line  =  NI'  *  v_bus;
i_line  =  y_line  *  v_line;

I_line  =  zeros(size(i_line));
```

```
I_line(1:16)  =  Ib1  .*  i_line(1:16);
I_line(17:19)  =  Ib2  .*  i_line(17:19);
I_line(20:22)  =  Ib1  .*  i_line(20:22);

%  now  generate  a  report:

fprintf('Bus  Voltages  are:\n')
fprintf('Bus     Per-Unit    Angle  (Radians)     (Degrees)\n');
for  i  =  1:Nb
  fprintf('%3.0f     %3.4f      %8.4f         %6.2f\n',i,  V(i),  th(i),  (180/pi)*th(i));
end

fprintf('Line  Current  Flows\n')
fprintf('Line     Amperes\n')
for  i  =  1:22
  fprintf('%3.0f    %3.1f\n',i,  abs(I_line(i)));
end
```