10 92

```
%{
Written by Tianyu Gao
Born on Sept 16, 2015
%}
```

$$\frac{dVC_A}{dt} = vC_{A,0} - vC_A - k_1C_AV$$

$$\frac{dVC_B}{dt} = vC_{B,0} - vC_B + k_1C_AV - k_2C_BV + k_3C_DV - k_4C_BV$$

$$\frac{dVC_C}{dt} = vC_{C,0} - vC_C + k_4C_BV$$

$$\frac{dVC_D}{dt} = vC_{D,0} - vC_D + k_2C_BV - k_3C_DV$$

Steady State and $\tau = \frac{V}{v}$

$$C_{A,0} = C_A + k_1C_A\tau$$

$$C_{B,0} = C_B - k_1C_A\tau + k_2C_B\tau - k_3C_D\tau + k_4C_B\tau$$

$$C_{C,0} = C_C - k_4C_B\tau$$

$$C_{D,0} = C_D - k_2C_B\tau + k_3C_D\tau$$

```
clc
clear all
close all

k = [0.1, 0.2, 0.1, 0.8]; % /sec
t = 10/1; % sec
Ci = [5; 0; 0; 1]; %feed
A = [1+k(1)*t, 0, 0, 0; -k(1)*t, 1+k(2)*t+k(4)*t, 0,-k(3)*t; 0, -
k(4)*t, 1, 0; 0, -k(2)*t, 0, 1+k(3)*t,];

[L, U, P]=lu(A); % P is permutation matrix, which L * U = P * A
disp('Solution is')
Css = U\(L\(P*Ci))
disp('Triangular matrix mutiple the feed vector is')
U*Css
```

```
Solution is

Css =

    2.5000
    0.3000
    2.4000
```

```
    0.8000

Triangular matrix mutiple the feed vector is

ans =

    5.0000
    2.5000
    1.8182
    1.4545
```

Published with MATLAB® R2015a

```
%{
Written by Tianyu Gao
Born on Sept, 20
%}
clc
clear all
```

Problem a

8

```
A = [3 2 2 1; 2 3 1 2; -1 1 2 0; 2 4 3 5];
[V,D] = eig(A);
V
D
sprintf('Norms of these  four eignevectors are %.2f, %.2f, %.2f and
 %.2f ', ...
 (norm(V(:,1))),(norm(V(:,2))),(norm(V(:,3))),(norm(V(:,4))))
% So we can see that Matlab uses 2-norm to normalize eigenvectors.
```

but how?

```
V =

   0.3446 + 0.0000i  -0.1195 - 0.3317i  -0.1195 + 0.3317i  -0.5000 +
 0.0000i
   0.4569 + 0.0000i  -0.5295 + 0.2518i  -0.5295 - 0.2518i  -0.5000 +
 0.0000i
   0.0183 + 0.0000i   0.7213 + 0.0000i   0.7213 + 0.0000i   0.5000 +
 0.0000i
   0.8198 + 0.0000i   0.0723 - 0.0799i   0.0723 + 0.0799i   0.5000 +
 0.0000i


D =

   8.1370 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 +
 0.0000i
   0.0000 + 0.0000i   1.4315 + 0.8090i   0.0000 + 0.0000i   0.0000 +
 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.4315 - 0.8090i   0.0000 +
 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   2.0000 +
 0.0000i


ans =

Norms of these  four eignevectors are 1.00, 1.00, 1.00 and 1.00
```

Problem b

```
A = sym(A);
```

```
[Ve,De]=eig(A);
Ve
De
% When use sym function. Matlab will output the exact result.
```

```
Ve =

[ -1, (43/(9*(454^(1/2)/3 + 341/27)^(1/3)) + (454^(1/2)/3 +
 341/27)^(1/3) + 11/3)^2/2 - 215/(9*(454^(1/2)/3 + 341/27)^(1/3))
 - 5*(454^(1/2)/3 + 341/27)^(1/3) - 31/3, (43/(18*(454^(1/2)/3 +
 341/27)^(1/3)) + (454^(1/2)/3 + 341/27)^(1/3)/2 + (3^(1/2)*(43/
(9*(454^(1/2)/3 + 341/27)^(1/3)) - (454^(1/2)/3 + 341/27)^(1/3))*1i)/2
 - 11/3)^2/2 + 215/(18*(454^(1/2)/3 + 341/27)^(1/3)) + (5*(454^(1/2)/3
 + 341/27)^(1/3))/2 + (3^(1/2)*(43/(9*(454^(1/2)/3 + 341/27)^(1/3))
 - (454^(1/2)/3 + 341/27)^(1/3))*5i)/2 - 31/3, (43/(18*(454^(1/2)/3
 + 341/27)^(1/3)) + (454^(1/2)/3 + 341/27)^(1/3)/2 - (3^(1/2)*(43/
(9*(454^(1/2)/3 + 341/27)^(1/3)) - (454^(1/2)/3 + 341/27)^(1/3))*1i)/2
 - 11/3)^2/2 + 215/(18*(454^(1/2)/3 + 341/27)^(1/3)) + (5*(454^(1/2)/3
 + 341/27)^(1/3))/2 - (3^(1/2)*(43/(9*(454^(1/2)/3 + 341/27)^(1/3)) -
 (454^(1/2)/3 + 341/27)^(1/3))*5i)/2 - 31/3]
[ -1, (43/(9*(454^(1/2)/3 + 341/27)^(1/3)) + (454^(1/2)/3 +
 341/27)^(1/3) + 11/3)^2/2 - 172/(9*(454^(1/2)/3 + 341/27)^(1/3)) -
 4*(454^(1/2)/3 + 341/27)^(1/3) - 44/3,           (43/(18*(454^(1/2)/3
 + 341/27)^(1/3)) + (454^(1/2)/3 + 341/27)^(1/3)/2 + (3^(1/2)*(43/
(9*(454^(1/2)/3 + 341/27)^(1/3)) - (454^(1/2)/3 + 341/27)^(1/3))*1i)/2
 - 11/3)^2/2 + 86/(9*(454^(1/2)/3 + 341/27)^(1/3)) + 2*(454^(1/2)/3
 + 341/27)^(1/3) + 3^(1/2)*(43/(9*(454^(1/2)/3 + 341/27)^(1/3))
 - (454^(1/2)/3 + 341/27)^(1/3))*2i - 44/3,           (43/
(18*(454^(1/2)/3 + 341/27)^(1/3)) + (454^(1/2)/3 + 341/27)^(1/3)/2
 - (3^(1/2)*(43/(9*(454^(1/2)/3 + 341/27)^(1/3)) - (454^(1/2)/3
 + 341/27)^(1/3))*1i)/2 - 11/3)^2/2 + 86/(9*(454^(1/2)/3 +
 341/27)^(1/3)) + 2*(454^(1/2)/3 + 341/27)^(1/3) - 3^(1/2)*(43/
(9*(454^(1/2)/3 + 341/27)^(1/3)) - (454^(1/2)/3 + 341/27)^(1/3))*2i -
 44/3]
[  1,          43/(454^(1/2)/3 + 341/27)^(1/3) - (43/(9*(454^(1/2)/3
 + 341/27)^(1/3)) + (454^(1/2)/3 + 341/27)^(1/3) + 11/3)^2 +
 9*(454^(1/2)/3 + 341/27)^(1/3) + 26,       26 - 43/(2*(454^(1/2)/3 +
 341/27)^(1/3)) - (9*(454^(1/2)/3 + 341/27)^(1/3))/2 - (3^(1/2)*(43/
(9*(454^(1/2)/3 + 341/27)^(1/3)) - (454^(1/2)/3 + 341/27)^(1/3))*9i)/2
 - (43/(18*(454^(1/2)/3 + 341/27)^(1/3)) + (454^(1/2)/3 +
 341/27)^(1/3)/2 + (3^(1/2)*(43/(9*(454^(1/2)/3 + 341/27)^(1/3))
 - (454^(1/2)/3 + 341/27)^(1/3))*1i)/2 - 11/3)^2,       26 - 43/
(2*(454^(1/2)/3 + 341/27)^(1/3)) - (9*(454^(1/2)/3 + 341/27)^(1/3))/2
 + (3^(1/2)*(43/(9*(454^(1/2)/3 + 341/27)^(1/3)) - (454^(1/2)/3
 + 341/27)^(1/3))*9i)/2 - (43/(18*(454^(1/2)/3 + 341/27)^(1/3)) +
 (454^(1/2)/3 + 341/27)^(1/3)/2 - (3^(1/2)*(43/(9*(454^(1/2)/3 +
 341/27)^(1/3)) - (454^(1/2)/3 + 341/27)^(1/3))*1i)/2 - 11/3)^2]
[  1,

                         1,
```

```
                  1,




     1]


De =

[ 2,
       0,

                               0,

                                                       0]
[ 0, 43/(9*(454^(1/2)/3 + 341/27)^(1/3)) + (454^(1/2)/3 +
 341/27)^(1/3) + 11/3,

                                            0,

                                                                    0]
[ 0,
           0, 11/3 - (454^(1/2)/3 + 341/27)^(1/3)/2 - (3^(1/2)*(43/
(9*(454^(1/2)/3 + 341/27)^(1/3)) - (454^(1/2)/3 + 341/27)^(1/3))*1i)/2
 - 43/(18*(454^(1/2)/3 + 341/27)^(1/3)),

                                                              0]
[ 0,
           0,

                                            0, 11/3 - (454^(1/2)/3
 + 341/27)^(1/3)/2 + (3^(1/2)*(43/(9*(454^(1/2)/3 + 341/27)^(1/3))
 - (454^(1/2)/3 + 341/27)^(1/3))*1i)/2 - 43/(18*(454^(1/2)/3 +
 341/27)^(1/3))]
```

Published with MATLAB® R2015a

```
%{
Written by Tianyu Gao
Born on Sept, 21, 2015
Power method for eigenvalues
%}
clc
clear all

A=[2,-1,0;-1,2,-1;0,-1,3];
x0 = [1, 0, 0]';
N = 1000; % maximum iteration times.
eps = 10^-6; % error
i = 1;
x = A * x0;
x = x/norm(x,Inf);
s = x0;

while norm(s-x)>=eps && i<N % if p-q less than error, we find the
 eigenvalue
    %and if i more than maximun times, stop iteration.
    s = x;
    y= A*x;
    x=y/norm(y,Inf);% normalize vector x to converge;
    i = i + 1;
end
if i == N
    sprintf('Fail: reaching maximum iteration times.')
else
    value = norm(y,Inf)
    sprintf('The max eigenvalue is %f',(value))
    disp('The eigenvector is')
    x/norm(x)
end


value =

    3.8019


ans =

The max eigenvalue is 3.801939

The eigenvector is

ans =

    0.3280
   -0.5910
    0.7370
```

10

Published with MATLAB® R2015a

10

Table of Contents


```
%{
Written by Tianyu Gao
%}
function Assignmeng_2_4()
```

Problem a:

$$f(x) = x^3 - 5x^2 + 7x - 3$$

$$f(x) = (x-1)^2(x-3)$$

The roots of $f(x)$ is 1,1,3

Problem b,c:

$$f(x) = x^3 - 5x^2 + 7x - 3$$

$$f^{(1)}(x) = 3x^2 - 10x$$

$$f^{(2)}(x) = 6x$$

```
clc
clear all
x1 = 0; % initial guess for standard update function
x2 = 0; % initial guess for modified update function
X1 = zeros(5,1);
X2 = zeros(5,1);

for i = 1:5
    y1 = NR(x1);
    x1 = x1 + y1;
    X1(i) = x1;

end
for i = 1:5
```

```
    y2 = mNR(x2);
    x2 = x2 + y2;
    X2(i) = x2;
end
disp('Problem b: Standard function:')
vpa(X1,8)
disp('Problem C: Modified function:')
vpa(X2,8)
```

Proble d:

```
x1 = 4; % initial guess for standard update function
x2 = 4; % initial guess for modified update function
X1 = zeros(5,1);
X2 = zeros(5,1);

for i = 1:5
    y1 = NR(x1);
    x1 = x1 + y1;
    X1(i) = x1;

end
for i = 1:5
    y2 = mNR(x2);
    x2 = x2 + y2;
    X2(i) = x2;
end
disp('Problem d: Standard function:')
vpa(X1,8)
disp('Problem d: Modified function:')
vpa(X2,8)
```

```
Problem d: Standard function:

ans =

       3.4
       3.1
 3.0086957
 3.0000746
       3.0

Problem d: Modified function:

ans =

 2.6363636
 2.8202247
 2.9617282
 2.9984787
 2.9999977
```

```
end
```

Function

```
function [y] = fx(x)    % f(x)
y = x^3 - 5*x^2 + 7*x -3;
end
function [y] = f1x(x)   % first derivative function
y = 3*x^2 - 10 * x + 7;
end
function [y] = f2x(x)  % second deritivative function
y = 6*x - 10;
end
function [y] = NR(x)   % standard update function
y = -fx(x)/f1x(x);
end
function [y]= mNR(x)   % modified update function
y = -fx(x)*f1x(x)/(f1x(x)^2 - fx(x)*f2x(x));
end
% The result is different from that of b,c, which means different
initial
% guesses may yeild different roots.
```

```
Problem b: Standard function:

ans =

 0.42857143
 0.68571429
  0.8328654
 0.91332989
 0.95578329

Problem C: Modified function:

ans =

 1.1052632
 1.0030817
 1.0000024
       1.0
       1.0
```

Published with MATLAB® R2015a

8

```
function Assignment_2_5
m=1;  % initial guess 1
n=2;  % initial guess 2
N = 1000;  % maxiumn iteration times
eps = 10^-6; % tolerance

for i = 1:1:N
    p = myfun(m);
    q = myfun(n);
    delta = -q*(n-m)/(q-p); % I use Secant methods.
    m = n;
    n = n + delta;
    if abs(n-m)<eps  % stop iteration when |x(k)-x(k-1)|<eps
        break;
    end
end
if i == N
    disp('Fail: reaching maxiumn iteration times');
else
sprintf('When h = %g, velocity requirements is satisfied.',n)
end
end



function F = myfun(x)
V = 5; % m/s
t = 3; % s
L = 5; % m
g = 9.81; %m/s^2
F = V/(2*g*x)^0.5-tanh(t*(2*g*x)^0.5/2/L);
end
```

tolerance

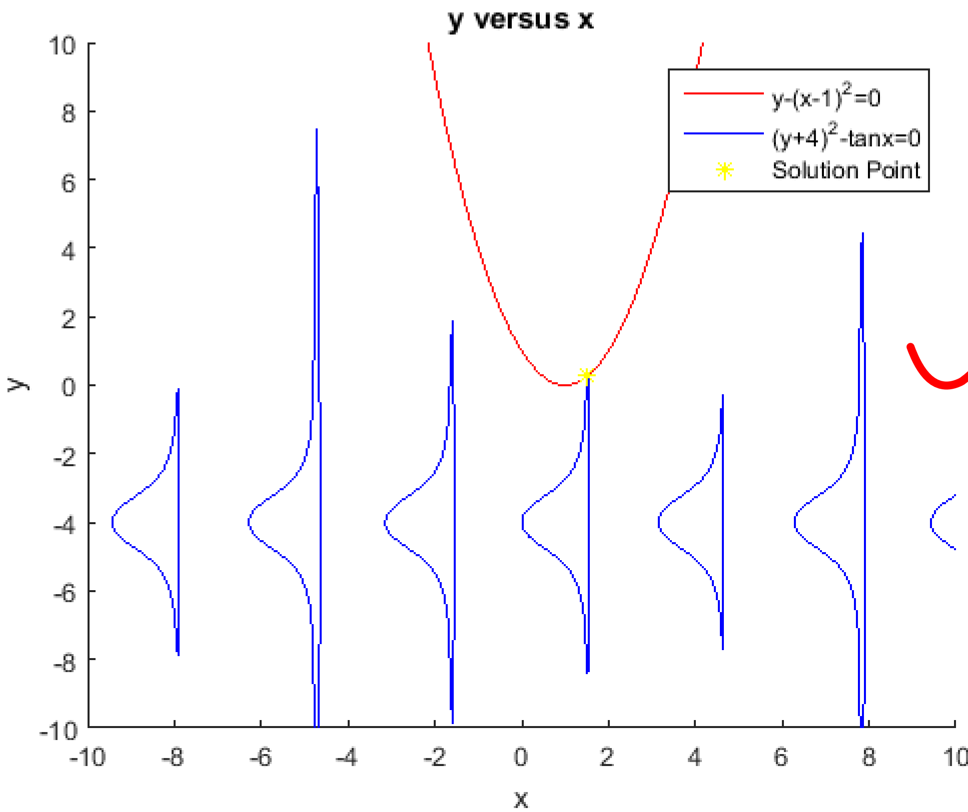
```
ans =

When h = 1.48958, velocity requirements is satisfied.
```

Published with MATLAB® R2015a

```
function Assignment_2_6()
clear all
x=[0;0];
i=0;
N = 10000;
eps = 10^-6;
J = Jac(x);
F = Fun(x);
while norm(F,Inf)>eps && i<N
J = Jac(x);
F = Fun(x);
detla = J\(-1*F);
x = x + detla;
i = i+1;
end

if i >= N
    disp('Fail: reaching maximum iteration times')
else
    vpa(x,8)
end
figure ;
hold on;
f1=ezplot('y-(x-1).^2',[-10 10]);
set(f1,'color','r');
f2=ezplot('(y+4).^2 - tan(x)',[-10 10]);
set(f2,'color','b');
plot(x(1),x(2),'*y');
legend('y-(x-1)^2=0','(y+4)^2-tanx=0','Solution Point')
title('y versus x');
hold off;
end

function J = Jac(x)
J = [-2*x(1)+2, 1;
    -sec(x(1)).^2, 2*x(2)+8];
end
function F = Fun(x)
F = [x(2)-(x(1)-1).^2;
    (x(2)+4).^2 - tan(x(1))];
end
```

```
ans =

  1.5159068
 0.26615985
```

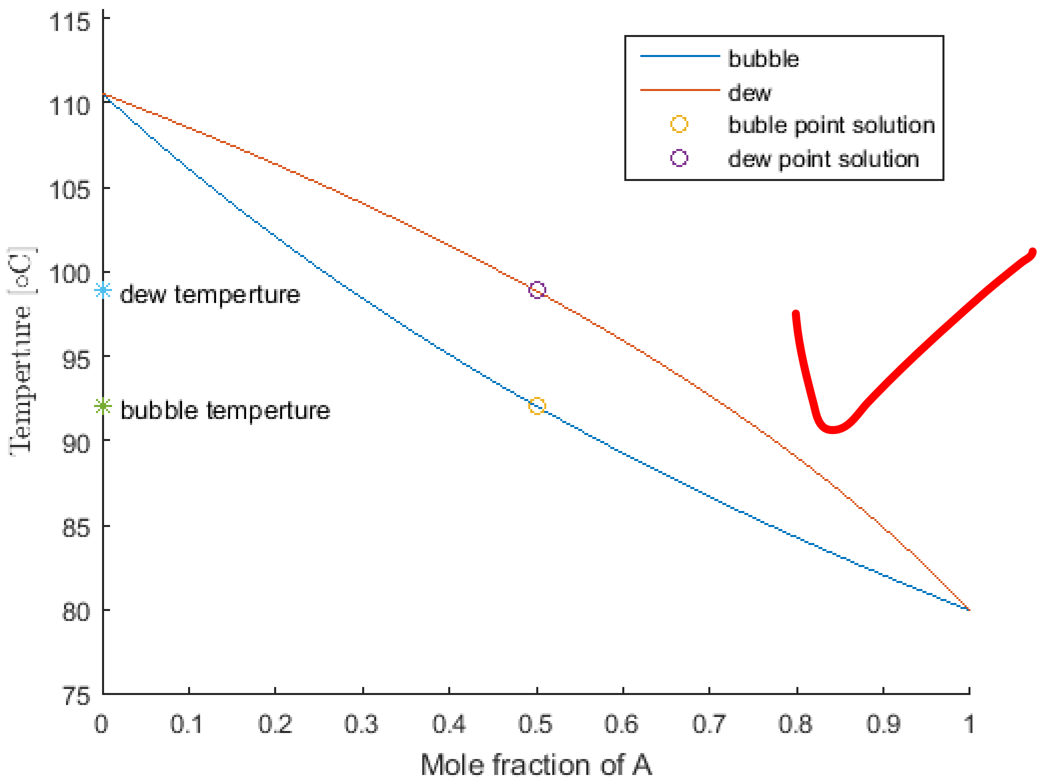
10



Published with MATLAB® R2015a

10

```
function Assignment_2_7()
%clear all
P = 760; %mmHg
T1 = fzero(@(x) exp(-3848.09/(x+273.15)+17.5318)-P,10); % boiling
 point of alpha
T2 = fzero(@(x) exp(-4328.12/(x+273.15)+17.913)-P,10);  % boiling
 point of beta
T = linspace(T1,T2);
x = zeros(100,1); % assign vector x;
y = zeros(100,1); % assign vector y
for i = 1:1:100   % iteration times equalls 100, decided by
 'linspace'.
    P1 = P1(T(i)); % function P1 to calculate pressure of alpha at
 temperature T(i)
    P2 = P2(T(i)); % function P2 to calculate pressure of beta at
 temperature T(i)
    p = fzero(@(x)P-P1*x-P2*(1-x),0.5); % solve for liquid composition
    q = fzero(@(y)1/P-y/P1-(1-y)/P2,0.5); % solve for gas composition
    x(i)=p;
    y(i)=q;
end
figure;
hold on;
plot(x,T,y,T);
axis([0,1,T1-5,T2+5]);
xlabel('Mole fraction of A');
ylabel('Temperture [$\circ$C]','Interpreter','LaTex');
Tb = fzero(@(x) 0.5*exp(-3848.09/(x+273.15)+17.5318)+0.5*exp(-4328.12/
(x+273.15)+17.913)-P,90);
% solve for bubble point for equimolar mixture.
Td = fzero(@(x) 0.5/exp(-3848.09/(x+273.15)+17.5318)+0.5/exp(-4328.12/
(x+273.15)+17.913)-1/P,100);
% solve for dew point for equimolar mixture.
plot(0.5,Tb,'o')
plot(0.5,Td,'o')
legend('bubble','dew','buble point solution','dew point solution')
plot(0,Tb,'*')
plot(0,Td,'*')
text(0.02,Tb,'bubble temperture')
text(0.02,Td,'dew temperture')
end

function P=P1(T)
A = -3848.09;
B = 17.5318;
P = exp(A/(T+273.15)+B);
end
function P=P2(T)
A = -4328.12;
B = 17.913;
P = exp(A/(T+273.15)+B);
end
```



Published with MATLAB® R2015a

```
function Assignment2_8
x0=[0.05,0.02]';
x =fsolve(@extenta,x0);
disp('Equilibrium Extent on Condition A is')
vpa(x,4)
x =fsolve(@extentb,x0);
disp('Equilibrium Extent on Condition B is')
vpa(x,4)
end
```

problem a:

7

```
function E=extenta(x) % vector x is the reaction extents of two
 reactions.
K =[0.1071;0.01493];  %Equilibrium constant
Ni=[1/3, 0, 1/3, 1/3, 0]';  % Feed mole vector
N(1) = Ni(1)-2*x(1);        % Equilibrium mole vector
N(2) = Ni(2)+2*x(1);
N(3) = Ni(3)+x(1)-x(2);
N(4) = Ni(4)-x(2);
N(5) = Ni(5)+2*x(2);
Nt = N(1)+N(2)+N(3)+N(4)+N(5);   % Totle mole
y(1) = N(1)/Nt;                  % mole fraction
y(2) = N(2)/Nt;
y(3) = N(3)/Nt;
y(4) = N(4)/Nt;
y(5) = N(5)/Nt;
E = [y(2)^2*y(3)/y(1)^2-K(1);
    y(5)^2/y(3)/y(4)-K(2)];
end
```

```
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.



Equilibrium Extent on Condition A is

ans =

 0.05931
 0.02083
```

Problem b:

```
function E=extentb(x) % vector x is the reaction extents of two
 reactions.
```

```
K =[0.1071;0.01493];  %Equilibrium constant
Ni=[2, 0, 1/3, 1/3, 0]';  % Feed mole vector
N(1) = Ni(1)-2*x(1);        % Equilibrium mole vector
N(2) = Ni(2)+2*x(1);
N(3) = Ni(3)+x(1)-x(2);
N(4) = Ni(4)-x(2);
N(5) = Ni(5)+2*x(2);
Nt = N(1)+N(2)+N(3)+N(4)+N(5);   % Totle mole
y(1) = N(1)/Nt;                  % mole fraction
y(2) = N(2)/Nt;
y(3) = N(3)/Nt;
y(4) = N(4)/Nt;
y(5) = N(5)/Nt;
E = [y(2)^2*y(3)/y(1)^2-K(1);
    y(5)^2/y(3)/y(4)-K(2)];
end
```

```
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.



Equilibrium Extent on Condition B is

ans =

   0.405
 0.02842
```

Published with MATLAB® R2015a

10

```
function Assignment_2_9
clear all
x0=[0.1 0.1 0.1 0.1 0.1 0.1 0.1]'; %initial guess
x =fsolve(@myfun,x0);
vpa(x,4)
end

function F=myfun(x)
l=[100,100,200,75,100,75,50];
F=[x(1)-x(2)-x(6); % q1 = q2+q6
x(1)-x(7);         % q1 = q7
x(2)-x(3)-x(4);    % q2 = q3 + q4
x(2)-x(5);         % q5 = q3 + q4 = q2
l(3)*x(3)^2-l(4)*x(4)^2;
l(2)*x(2)^2+l(4)*x(4)^2+l(5)*x(5)^2-l(6)*x(6)^2;
l(1)*x(1)^2+l(6)*x(6)^2+l(7)*x(7)^2-5.2*10^5*pi^2*0.2^5/8/0.02/998];
end
```

```
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.



ans =

  0.2388
 0.08694
 0.03302
 0.05392
 0.08694
  0.1519
  0.2388
```

✓

Presentation

9

Published with MATLAB® R2015a