MATLAB SOLUTIONS TO THE CHEMICAL ENGINEERING PROBLEM SET[1]

Joseph Brule, John Widmann, Tae Han, Bruce Finlayson[2]
Department of Chemical Engineering, Box 351750
University of Washington
Seattle, Washington 98195-1750

INTRODUCTION

These solutions are for a set of numerical problems in chemical engineering. The problems were developed by Professor Michael B. Cutlip of the University of Connecticut and Professor Mordechai Shacham of Ben-Gurion University of the Negev for the ASEE Chemical Engineering Summer School held in Snowbird, Utah in August, 1997. The problem statements are provided in another document.[3] Professors Cutlip and Shacham provided a document which shows how to solve the problems using POLYMATH, Professor Eric Nuttall of the University of New Mexico provided solutions using Mathematica and Professor J. J. Hwalek provided solutions using Mathcad. After the conference, Professor Ross Taylor provided solutions in Maple, and Edward Rosen provided solutions in EXCEL. This paper gives the solution in MATLAB. All documents and solutions are available from http://www.che.utexas/cache.

These solutions are obtained using the version 5.0 of MATLAB Pro. Minor changes are needed to the files when using version 4.0 of MATLAB, mainly in the command giving the limits of integration when solving ordinary differential equations. The appropriate commands (changes from MATLAB 5.0) are given in the files as comments. The program MATLAB runs by executing commands, which can call files called m-files. Given below are the commands and m-files. The m-files are also available on a diskette. For ease in interpreting the text below, text is printed in Times font, whereas the MATLAB files are printed in Geneva font. Each problem is solved by setting the path for MATLAB (most easily done by opening the appropriate m-file, and issuing the command Prob_X. The m-file Prob_X.m may call other m-files, which are described below and are on the diskette. In the description below, any line beginning with a % is a comment.

The authors thank Professor Larry Ricker for helpful comments on the first draft of this paper.

[2] Joseph Brule just obtained his B.S. degree. Tae Han is a current undergraduate. Dr. John Widmann a recent Ph.D. graduate, and Bruce Finlayson is the Rehnberg Professor and Chair.

[3] "The Use of Mathematical Software packages in Chemical Engineering", Michael B. Cutlip, John J. Hwalek, Eric H. Nuttal, Mordechai Shacham, Workshop Material from Session 12, Chemical Engineering Summer School, Snowbird, Utah, Aug., 1997.

MATLAB Problem 1 Solution

A function of volume, f(V), is defined by rearranging the equation and setting it to zero.

$$pV^3 - b\,V^2 - R\,T\,V^2 + a\,V - a\,b = 0$$

This problem can be solved either by using the fzero command to find when the function is zero, or by using the roots command to find all the roots of the cubic equation, and both methods are illustrated here.

MATLAB has equation solvers such as fzero (in all versions) and fsolve (in the optimization Toolbox). To use the solvers one must define f(V) as a MATLAB function. An example of a function is the following script file named waalsvol.m. All statements following % are ignored by MATLAB. The semi-colons prevent the values from being printed while the program is being executed.

```
% filename waalsvol.m
function x=waalsvol(vol)
global press a b R T
x=press*vol^3-press*b*vol^2-R*T*vol^2+a*vol-a*b;
```

This script file can now be called by other MATLAB script files. In this problem, the molar volume and the compressibility factors are the variables of interest and the fsolve function finds the value of vol that makes x zero. The three parts of the problem, a, b, and c are done together in the m-file Prob_1.m.

```
%filename Prob_1.m
clear all
format short e
global press a b R T % make these parameters available to waalsvol.m

%set the constants
Pcrit=111.3;        % in atm
Tcrit=405.5;        % in Kelvin
R=0.08206;          % in atm.liter/g-mol.K
T=450;              % K

% the different values of pressure are stored in a single vector
Preduced=[0.503144 1 2 4 10 20];
a=27/64*R^2*Tcrit^2/Pcrit;
b=R*Tcrit/(8*Pcrit);

% each pass of the loop varies the pressure and the volume is calculated
for j=1:6
   press=Pcrit*Preduced(j);
   volguess=R*T/press;
```

```
    % Use fzero ( or fsolve) to calculate volume
    vol= fzero('waalsvol',volguess);
    z=press*vol/(R*T);
    result(j,1)=Preduced(j);
    result(j,2)=vol;
    result(j,3)= press*vol/(R*T);
end
% end of calculation

disp('   Preduced     Molar Vol      Zfactor ')
disp(result)
plot(result(:,1),result(:,3),'r')
title('Compressibility factor vs Reduced pressure')
xlabel('Reduced pressure')
ylabel('Compressibiliity factor')
```

The output is presented below in tabular form and in Figure 1.

```
  P-reduced      Molar Vol       Zfactor
  5.0314e-001   5.7489e-001   8.7183e-001
  1.0000e+000   2.3351e-001   7.0381e-001
  2.0000e+000   7.7268e-002   4.6578e-001
  4.0000e+000   6.0654e-002   7.3126e-001
  1.0000e+001   5.0875e-002   1.5334e+000
  2.0000e+001   4.6175e-002   2.7835e+000
```

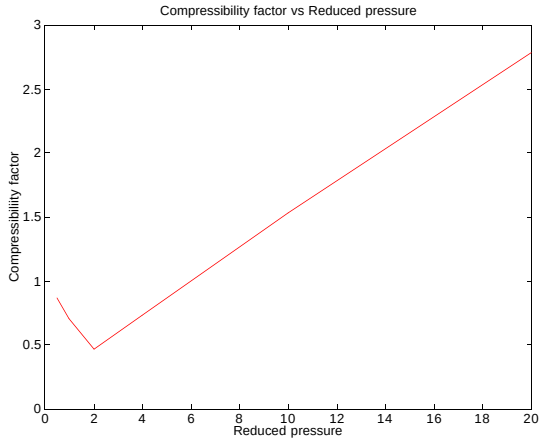


Figure 1. Compressibility Factor versus Reduced Pressure

An alternative suggested by Professor Ricker is to find all three roots to the cubic equation, and then use the largest one as the volume appropriate to a gas. This option is achieved by replacing the vol= fzero (...) command with the following.

```
vols=roots([press, -(press*b+R*T), a, -a*b]);  % Finds all roots
vol=max(vols(find(imag(vols) == 0)));              % finds largest real root
```

MATLAB Problem 2 Solution

To solve the first part of this problem, Equation (6) is written as a matrix problem

$$A\,X = f$$

and solved with one command.

$$X = A \backslash f'$$

```
%filename Prob_2.m
A=[0.07 0.18 0.15 0.24
   0.04 0.24 0.10 0.65
   0.54 0.42 0.54 0.10
   0.35 0.16 0.21 0.01];
f = [0.15*70 0.25*70 0.40*70 0.2*70];
disp('Solution for D1 B1 D2 B2 is:')
X = A\f'
```

The solution is $D_1 = 26.25$, $B_1 = 17.50$, $D_2 = 8.75$, $B_1 = 17.50$.

The mole fractions for column 2 are solved for directly by evaluating Equation (7).

```
D1 = X(1);
B1 = X(2);
disp ('Solve for Column 2')
D=D1+B1                          %43.75 mol/min
X_Dx=(0.07*D1+0.18*B1)/D         %0.114 mole fraction
X_Ds=(0.04*D1+0.24*B1)/D         %0.120 mole fraction
X_Dt=(0.54*D1+0.42*B1)/D         %0.492 mole fraction
X_Db=(0.35*D1+0.16*B1)/D         %0.274 mole fraction
```

The mole fractions for column 3 are solved for directly by evaluating Equation (8).

```
D2 = X(3);
B2 = X(4);
disp('Solve for Column 3')
B=D2+B2                          %26.25 mol/min
X_Bx=(0.15*D2+0.24*B2)/B         %0.2100 mole fraction
X_Bs=(0.10*D2+0.65*B2)/B         %0.4667 mole fraction
X_Bt=(0.54*D2+0.10*B2)/B         %0.2467 mole fraction
X_Bb=(0.21*D2+0.01*B2)/B         %0.0767 mole fraction
```

MATLAB Problem 3 Solution

Problem 3a involves fitting a polynomial to a set of data, which is done with the command MATLAB polyfit. Problem 3b can be put into a form that creates a polynomial, too, and it is solved with polyfit. Problem 3c, however, involves nonlinear regression, and an optimization routine, fmins, is used to find the parameters for it. The same approach could be used for Problem 3b as well, in fact for any nonlinear regression problem.

(a) Data regression with a polynomial

```
%To solve part a, insert the data:
vp = [    1     5    10    20    40    60   100   200   400   760]
T = [-36.7  -19.6  -11.5   -2.6   7.6   15.4   26.1 42.2   60.6   80.1]

%set the degree of polynomial: p(1) = a(n),...p(n+1) = a(0)
m = 4              % 'm' here is one less than 'n' in the problem statement

%fit the polynomial
p=polyfit(T,vp,m)
%p =   3.9631e-06   4.1312e-04   3.6044e-02   1.6062e+00   2.4679e+01

%evaluate the polynomial for every T (if desired)
z=polyval(p,T)
%z =   1.0477e+00   4.5184e+00   1.0415e+01   2.0739e+01   3.9162e+01
%   5.9694e+01   1.0034e+02   2.0026e+02   3.9977e+02   7.6005e+02

%calculate tne norm of the error
norm(vp-polyval(p,T))

%plot results
plot(T,z,'or',T,vp,'b')
Title('Vapor Pressure with m = 4')
xlabel('T (C)')
ylabel('vp (mm Hg)')
```

The norm is the square root of the sum of squares of differences between the data and the curvefit, and its value here is 1.4105. A plot of the correlation and data is shown in Figure 2. If one runs the same file with different values of n, the results for the least squares value, a, are:

m	a (Vandermonde)	a(powers of x)
1	344.	344.
2	92.2	92.2
3	14.3	14.3
4	1.41	1.41
5	1.39	1.41
6	1.10	1.65
7	0.630	
8	0.564	
9	1.31×10^{-11}	

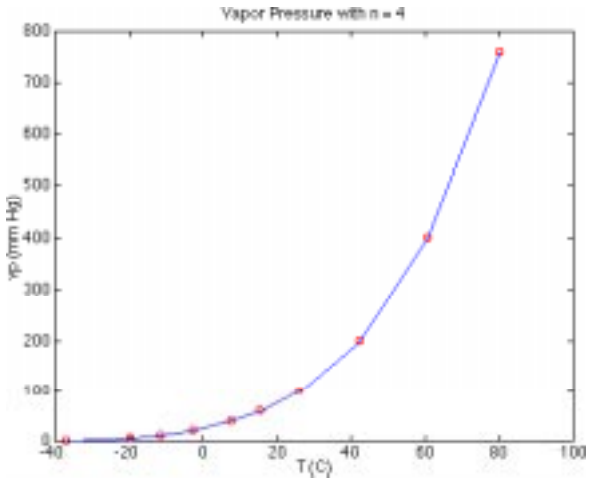


Figure 2. Comparison of Polynomial Correlation with Original Data

Note that when a high enough degree of polynomial is used the curve fit is exact at the data points. This result happens because MATLAB utilizes the Vandermonde matrix to solve the equations. Less complicated methods of solution have more numerical roundoff error, and that is the reason the error eventually starts increasing as more terms are added to the polynomial.

(b) Data regression with Clausius-Clapeyron Equation

The file Prob_3b is run to minimize the sum of the squares of the difference between the predicted value and the data when expressed as a logarithm to the base 10.

```
% file Prob_3b.m
%To solve part b, insert the data:
vp = [    1     5    10    20    40    60   100   200   400   760]
T = [-36.7  -19.6  -11.5   -2.6   7.6   15.4   26.1 42.2   60.6   80.1]

% create the new variables
y = log10(vp);
x = 1./(T+273.15);

% fit the polynomial
p = polyfit(x,y,1)
% p = -2035.33  8.75201

%To compute the norm based on the logarithm of the vapor pressure
norm(y - polyval(p,x))
% norm = 0.2464

%To compute the norm based on the vapor pressure
norm(vp-10.^(polyval(p,x)))
% norm = 224.3
```

Note that the sum of squares is actually greater with this form of the curve, but this form of the solution can be extended beyond the limits of the data with more confidence since it has a theoretical justification.

(c) Data regression with the Antoine Equation

This curvefit cannot be rearranged into a polynomial function. Thus, we create a function as the sum of squares of the difference between the data and the curvefit, and minimize this function with respect to the parameters in the Antoine equation. We first construct the function that calculates function to be minimized; note that it is the logarithm of the vapor pressure that is being fit, rather than the vapor pressure itself. (A good homework problem is to change the function 'f' to be the vapor pressure rather than its logarithm and compare the results.)

```
function y3=fit_c(p)
global vp T
% function to fit vapor pressure to the data
a = p(1);
b = p(2);
c = p(3);
f = log10(vp) - a + b./(T+c);
%f = vp - 10.^(a - b./(T+c));
y3=sum(f.*f);
```

The script Prob_3c is run to minimize the sum of the squares of the difference between the predicted value and the data.

```
%filename Prob_3c.m
%To solve part c, insert the data:
vp = [    1     5    10    20    40    60   100   200   400   760]
T = [-36.7  -19.6  -11.5   -2.6   7.6   15.4   26.1 42.2   60.6   80.1]

% Make vp and T available in fit2
global vp T

% set initial guesses of parameters
p0(1) = 10;
p0(2) = 2000;
p0(3) = 273;

% call least squares minimization
ls = fmins('fit_c',p0)
```

The result is ls = 5.7673 677.09   153.89.

```
% To compute the sum of squares of errors:
vpfit1=(vp - 10.^(ls(1) - ls(2)./(T+ls(3))));
norm(vpfit1)
```

```
%To compute the norm based on the logarithm of the vapor pressure
vpfit2 = log10(vp) - (ls(1) - ls(2)./(T+ls(3)));
norm(vpfit2)
```

The value of the norm for the vapor pressure is 16.3. The norm for the logarithm of the vapor pressure is 0.0472. (The sum of squares of the difference is then 0.00223). Note that the norm of the logarithm of the vapor pressure went down when going from Problem 3b to 3c, as it should since an additional parameter has been included.

MATLAB Problem 4 Solution

The functions f(1) through f(7) are defined by setting the given linear and nonlinear equations to zero:

$$f(1) = C_C\, C_D - K_{C1}\, C_A\, C_B$$

$$f(2) = C_X\, C_Y - K_{C2}\, C_B\, C_C$$

$$f(3) = C_Z - K_{C3}\, C_A\, C_X$$

$$f(4) = C_{A0} - C_A - C_D - C_Z$$

$$f(5) = C_{B0} - C_B - C_D - C_Y$$

$$f(6) = C_D - C_Y - C_C$$

$$f(7) = C_Y - C_X - C_Z$$

The equilibrium equations are rearranged so that division by the unknowns is avoided. Root finding techniques may have iterates that approach zero which can cause divergence.

This set of equations is solved in two different ways: the first method uses the command fsolve and the second method uses the Newton-Raphson method. While fsolve is sufficient for this problem, it might not be work in all cases. Then the Newton-Raphson method must be programmed by the user.

Method 1 using fsolve.

```
%filename prob4.m
function f = prob4(cvector)
global Cao Cbo Kci Kcii Kciii

% cvector are the concentrations of the seven species. cvector(1) is the concentration of species
a,
% cvector(2) is the concentration of b etc.

f(1)= cvector(3)*cvector(4)-Kci*cvector(1)*cvector(2);
f(2)= cvector(6)*cvector(5)-Kcii*cvector(2)*cvector(3);
f(3)= cvector(7)- Kciii*cvector(1)*cvector(5);
f(4)= Cao - cvector(1) - cvector(4) - cvector(7);
f(5)= Cbo - cvector(2) - cvector(4) - cvector(6);
f(6)= cvector(4) - cvector(6) - cvector(3);
f(7)= cvector(6) - cvector(5)- cvector(7);
```

Next one calls fsolve in the main program.

```
%filename Prob_4.m
global Cao Cbo Kci Kcii Kciii cvector

% define constants
Cao = 1.5; Cbo=1.5; Kci= 1.06; Kcii= 2.63; Kciii= 5;

%set initial conditions

% Initial guess and set tolerance
% remove the % in front of the desired initial guess
%cvector=[1.5 1.5 0 0 0 0 0]; %initial guess, part a
%cvector=[-.5 -1.5 -1 1 1 2 1];  %initial guess, part b
%cvector=[-18.5 -28.5 -10 10 10 20 10]; %initial guess, part c
guess=cvector;

%call fsolve
y = fsolve('prob4',guess)
```

The program gives the following solution.

```
guess = [1.5 1.5 0 0 0 0 0];
y = 0.4207    0.2429    0.1536    0.7053    0.1778    0.5518    0.3740
```

To test the solution, the function was evaluated at the value of y.

```
gg = feval('prob4', y)
gg =1.0e-06 *
  -0.0434   -0.1188    0.0759   -0.0021   -0.0021   -0.0010   -0.0012
```

Other initial conditions gave the same result, along with an initial message that the problem was nearly singular.

```
guess = [ -0.5000   -1.5000   -1.0000    1.0000    1.0000    2.0000    1.0000]
y = 0.4207    0.2429    0.1536    0.7053    0.1778    0.5518    0.3740

guess = [-18.5 -28.5 -10 10 10 20 10]
y = 0.4207    0.2429    0.1536    0.7053    0.1778    0.5518    0.3740
```

Method 1 using the Newton-Raphson method.

The Newton Raphson method for a system of equations is:

$$\text{cvector}_i^{k+1} = \text{cvector}_i^k - J_{ij}^k \, f(\text{cvector}^k)$$

where $\mathbf{J}^k$ is the Jacobian matrix as defined by:

$$J_{ij}^k = \left. \frac{\partial f_i}{\partial \text{cvector}_j} \right|_{\text{cvector}^k}$$

There must be a file, prob4.m, which computes the function (the same one used above), and a file, jac.m, which computes the jacobian.

```
%filename jac4.m
function J = jac4(cvector)
global Cao Cbo Kci Kcii Kciii
%  row 1
J(1,1)= -cvector(2)*Kci; J(1,2)= -cvector(1)*Kci; J(1,3)= cvector(4);
J(1,4)= cvector(3); J(1,5)=0; J(1,6)=0; J(1,7)=0;
%  row 2
J(2,1)= 0; J(2,2)= -Kcii*cvector(3); J(2,3)= -Kcii*cvector(2); J(2,4)=0;
J(2,5)= cvector(6); J(2,6)= cvector(5); J(2,7)=0;
% row 3
J(3,1)= -Kciii*cvector(5); J(3,2)=0; J(3,3)=0; J(3,3)=0; J(3,4)=0;
J(3,5)= -Kciii*cvector(1); J(3,6)= 0; J(3,7)=1;
% row 4
J(4,1)= -1; J(4,2)= 0; J(4,3)= 0; J(4,4)=-1; J(4,5)=0; J(4,6)=0; J(4,7)=-1;
% row 5
J(5,1)=0; J(5,2)=-1; J(5,3)=0; J(5,4)=-1; J(5,5)=0; J(5,6)=-1; J(5,7)=0;
% row 6
J(6,1)=0; J(6,2)=0; J(6,3)=-1; J(6,4)=1; J(6,5)=0; J(6,6)=-1; J(6,7)=0;
% row 7
J(7,1)=0; J(7,2)=0; J(7,3)=0; J(7,4)=0; J(7,5)=-1; J(7,6)=1; J(7,7)=-1;
```

The program Prob_4NR.m calls the functions prob4.m and jac4.m to use the Newton Raphson method to solve the system of equations.

```
%filename Prob_4NR.m
clear all
clc
global Cao Cbo Kci Kcii Kciii cvector

% define constants
Cao = 1.5; Cbo=1.5; Kci= 1.06; Kcii= 2.63; Kciii= 5;

% Initial guess and set tolerance
err=1;
iter=0;

% remove the % in front of the desired initial guess
cvector=[1.5 1.5 0 0 0 0 0]; %initial guess, part a
%cvector=[-.5 -1.5 -1 1 1 2 1];  %initial guess, part b
%cvector=[-18.5 -28.5 -10 10 10 20 10]; %initial guess, part c
guess=cvector;

while err > 1e-4 & iter < 200
   x= prob4(cvector);
   J= jac4(cvector);
   errr= -J\x';
   cvector=cvector+errr';
   errr=abs(errr);
   err= sqrt(sum(errr));
   iter=iter+1;
end
```

```
disp('guess')
disp(guess)
disp('error')
disp(err)
disp('   A        B        C        D        X        Y       Z');
disp(cvector);
disp('iter')
disp(iter)
```

The cases a, b, and c give the following results.

Part a:

```
EDU»react2
guess   1.5     1.5          0          0          0          0       0
error   1.4093e-06
       A              B              C              D              X
4.2069e-01   2.4290e-01   1.5357e-01   7.0533e-01   1.7779e-01
       Y              Z
  5.5177e-01     3.7398e-01
iter   7
```

Part b:

```
guess  -5.0e-01  -1.5e+00  -1.0e+00   1.0e+00   1.0e+00   2.0e+00   1.0e+00
error   8.9787e-08
       A              B              C              D              X
3.6237e-01  -2.3485e-01  -1.6237e+00   5.5556e-02   5.9722e-01
        Y              Z
  1.6793e+00     1.0821e+00
iter   8
```

Part c:

```
guess  -1.85e+01  -2.85e+01  -1.0e+01   1.0e+01   1.0e+01   2.0e+01   1.0e+01
error   1.4690e-05
       A              B              C              D              X
-7.0064e-01  -3.7792e-01   2.6229e-01   1.0701e+00  -3.2272e-01
        Y              Z
 8.0782e-01      1.1305e+00
iter   12
```

MATLAB Problem 5 Solution

The solution to problem 5 is obtained by issuing the command Prob_5. This problem is solved iteratively using

$$v_t^{k+1} = \sqrt{\frac{4\, g\, (\rho_p - \rho)\, D_p}{3\, C_D(v_t^k)\, \rho}}$$

until $v_t^{k+1} = v_t^k$ to machine accuracy.

```
%filename Prob_5.m
%(a) Calculate the terminal velocity for particles of coal

%Input the known values
rho_p=1800;                 %kg/m^3
D_p=0.208*10^(-3);          %m
T=298.15;          %K
rho=994.6;                  %kg/m^3
mu=8.931*10^(-4);           %kg/m/s
g=9.80665;                  %m/s^2

%Input an value of v_t and different value of v_t_g
v_t=10;                     %m/s
v_t_g=20;          %m/s

%Rearrange equation 13 to solve for zero
%
%           f(v_t)=v_t^2*(3C_D*rho)-4g(rho_p-rho)D_p=0

%Begin while loop
tae=0;
while tae == 0
    %Calculate Re from the guessed v_t_g
    Re=D_p*v_t_g*rho/mu;

    %Determine C_D
    if Re<0.1
            C_D=24/Re;
      elseif Re<1000
            C_D=24*(1+0.14*Re^0.7)/Re;
      elseif Re<350000
            C_D=0.44;
      else
            C_D=0.19-80000/Re;
    end

    %Calculate v_t
    v_t=sqrt((4*g*(rho_p-rho)*D_p)/(3*C_D*rho));
    if (v_t == v_t_g)
            tae = 1;
       else
            v_t_g=v_t;
```

```
    end
end
v_t             %0.0158 m/s
Re              %3.6556
C_D             %8.8427
```

(b) Estimate the terminal velocity of the coal particles in water within a centrifugal separator where the acceleration is 30.0g. To solve this problem, substitute 30.0*9.80655 for g in the above code. The results are: v_t = 0.2060 m/s. Re = 47.7226, C_D = 1.5566.

MATLAB Problem 6 Solution

Because the energy balance equations are identical for each of the tanks in series, the problem is easily solved allowing the number of tanks to be a variable defined by the user. Below, the variable "num_tanks" is defined immediately after the global variables are declared. Run the problem with the command Prob_6. The set of ordinary differential equations is defined in the file tanks.m.

```
function dT_dt = tanks(t,T)
global W UA M Cp Tsteam num_tanks To

for j = 1:num_tanks
        if j==1
                dT_dt(j) = (W*Cp*(To-T(j)) + UA*(Tsteam-T(j)))/(M*Cp);
        else
                dT_dt(j) = (W*Cp*(T(j-1)-T(j)) + UA*(Tsteam-T(j)))/(M*Cp);
        end
end

%       filename Prob_6.m
%       Heat Exchange in a Series of Tanks
clear
global W UA M Cp Tsteam num_tanks To

num_tanks = 3
W = 100;  % kg/min
UA = 10;        % kJ/min.C
M = 1000; % kg
Cp = 2.0;       % kJ/kg
Tsteam = 250;   % C
To = 20;        % C

T_initial = ones(1,num_tanks)*To;

t_start = 0;            % min
t_final = 90;           % min
tspan = [t_start t_final];
[t,T] = ode45('tanks',tspan,T_initial);
% For Version 4, use
% [t,T] = ode45('tanks',t_start,t_final,T_initial);
plot(t,T)
title('Temperature in Stirred Tanks')
xlabel('time (min)')
ylabel('T (C)')
output = [t T];
save temps.dat output -ascii
```

The solution is shown in Figure 3. The time to reach 99% of steady-state can be obtained b y interpolation from the values listed in the text file temps.dat which is written by MATLAB.

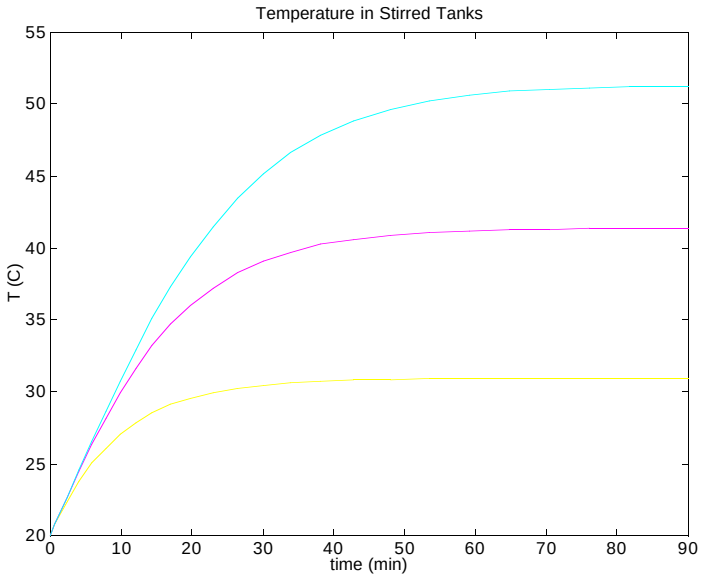


Figure 3. Temperature in three Stirred Tanks

MATLAB Problem 7 Solution

The second order differential equation can be written as a set of first order differential equations. First one defines the functions y_1 and y_2.

$$\frac{dy_1}{dz} = y_2, \quad \frac{dy_2}{dz} = \frac{k\, y_1}{D_{AB}}$$
$$y_1(0) = C_{A0},\ y_2(0) = \alpha \text{ (unknown)} \qquad (7.1)$$

This is the same as the original equation.

$$\frac{d}{dz}\left(\frac{dy_1}{dz}\right) = \frac{k\, y_1}{D_{AB}}$$

The derivatives of the y functions are defined by:

$$y_3 = \frac{\partial y_1}{\partial \alpha}, \quad y_4 = \frac{\partial y_3}{\partial \alpha}$$

The original equations and boundary conditions (7.1) are differentiated with respect to α to obtain:

$$\frac{dy_3}{dz} = y_4, \quad \frac{dy_4}{dz} = \frac{k\, y_3}{D_{AB}}$$
$$y_3(0) = 0,\ y_4(0) = 1$$

MATLAB has several ode solvers such as ode15, ode23, ode45 etc. In this problem, the initial condition of y_2 is not known. The shooting method guesses the initial condition of y_2, integrates the set of ode's and compares the integrated final condition with the boundary at z= 10^{-3}. The choice of α is adjusted and the whole process is repeated.

A number of root finding techniques can be used to find the initial value of y_2. The Newton Raphson method converges quickly and will be used in this problem. The function of interest is the value of y_2 at L.

$$f(\alpha) = y_2\,(L)$$

The derivative of the function is found by differentiating y_1 and y_2 with respect to alpha and z to obtain the derivative of 'f' with respect to α.

$$\frac{df}{d\alpha} = \left.\frac{\partial y_2}{\partial \alpha}\right|_{z=L} = y_4(L)$$

Thus the iteration procedure is

$$\alpha^{k+1} = \alpha^k - \frac{y_2^k(L)}{y_4^k(L)}$$

The m-file rhs7.m defines the derivatives with respect to z.

```
% filename rhs7.m
function ydot=rhs7(z,y)
global k Dab

row(1)=y(2);
row(2)=k*y(1)/Dab;
row(3)=y(4);
row(4)= k*y(3)/Dab;
ydot = row';
```

The m-file anyl7.m calculates the analytical solution to the differential equation for comparision.

```
% filename anyl7.m
function cc=anyl7(pos)
global L k Dab Cao
term1=L*(k/Dab)^0.5;
cc=Cao*(cosh(term1*(1-pos/L))/cosh(term1));
```

The m-file Prob_7.m chooses a value for α, integrates the equations, adjusts the value for α until f = $y_2(L) = 0$. It then compares the analytical solution with the integrated solution.

```
% filename Prob_7.m
clear all
clc
format short e
global L k Dab Cao alpha

L= 1e-3; Cao=0.2; k=1e-3; Dab=1.2e-9;
alpha=0;            % initial guess on shooting parameter
errr=1;
count=0;

%  use shooting method to solve ode.  Use Newton Raph to iterate on alpha
while errr>1e-12 &  count <100
            yo(1)=Cao;
            yo(2)= alpha;
```

```
            yo(3)=0;
            yo(4)=1;
            tspan = [0 L];
            [z,y]=ode45('rhs7',tspan,yo);          % 4 order RK method from 0 to L
%           For version 4 use
%           [z,y]=ode45('rhs7',0,L,yo);
            nn=size(y);
            n=max(nn);                             % finds length of vector y
            if y(n,4)==0
              'Newtons failed, derivative is zero'
               break
            end
            errr= y(n,2)/y(n,4);                        % finds the values at endpoint of
integration then
            alpha=alpha-y(n,2)/y(n,4);             %  adjust the shooting parameter alpha
            errr=abs(errr)
            count=count+1;
end

%  use analytical solution to compare to 4 order RK solution
for kk=1:n
            pos=z(kk);
            result(kk,1)=z(kk);
            result(kk,2)=anyl7(pos);
            result(kk,3)= y(kk,1);
            result(kk,4)= anyl7(pos)-y(kk,1);
end

' position   Analytical                                  integrated   difference'
disp(result);
count
y
plot(result(:,1),result(:,2),'r',result(:,1),result(:,3),'ob')
title('Diffusion/Reaction Inside Catalyst')
xlabel('x')
ylabel('c')
```

The toolbox PDE (an add-on to MATLAB) can also be used to solve this problem as well as the more complicated two-dimensional problem. Nonlinear reactions can easily be included. Shown in Figure 5a is a three-dimensional view of a finite element solution to the problem. While the finite element method in two dimensions is overkill for this problem, the same method can be applied to reaction and diffusion in a cylindrical catalyst pellet. The problem is then

$$\frac{d^2y_1}{dz^2} + \frac{1}{r}\frac{d}{dr}\left(r\frac{dy_1}{dr}\right) = \frac{k\,L^2}{D_{AB}}\,y_1, \quad \text{for } 0 \le r \le R \equiv 0.5\ L$$

$$y_1(r,0) = C_{A0},\ y_1(R,z) = C_{A0},\ \left.\frac{\partial y_1}{\partial r}\right|_{r=0} = 0,\ \left.\frac{\partial y_1}{\partial z}\right|_{z=1} = 0$$

and the solution is shown in Figure 5b. This additional capability of MATLAB is one of its most important benefits for students and faculty that solve transport problems numerically.

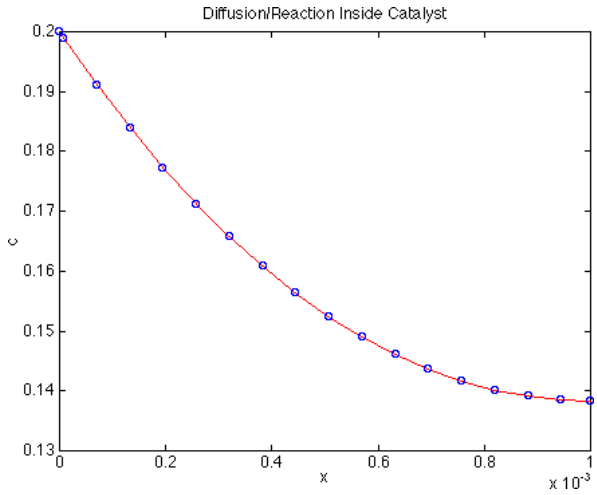
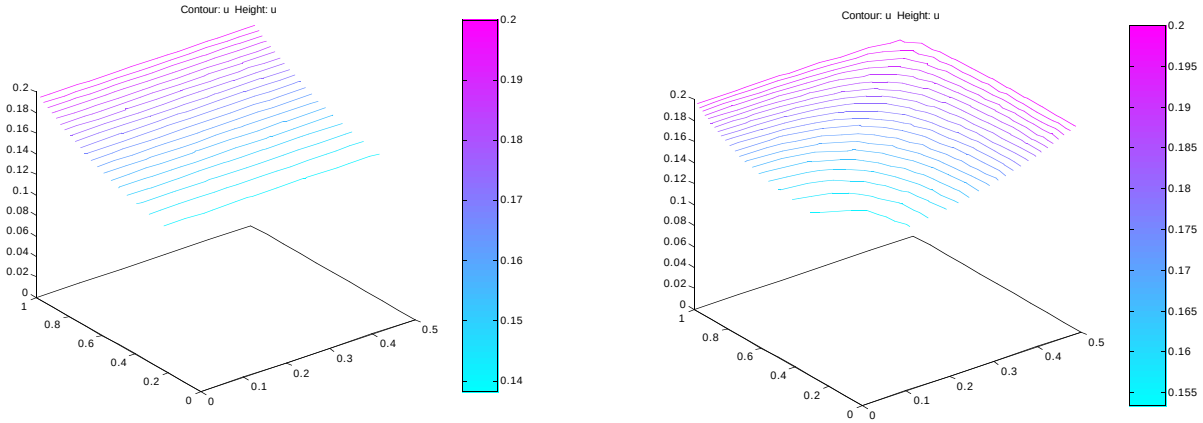


Figure 4. Concentration Profile inside Catalyst
(numerical solution indistinguishable from analytical solution)



(a) 1D Catalyst **(b) 2D Catalyst**

Figure 5. Concentration Profile inside Catalyst

MATLAB Problem 8 Solution

This problem requires the simultaneous solution of an ordinary differential equation and a non-linear algebraic equation. MATLAB does not have a function specifically designed for this task, but it does have functions that perform each of the individual tasks. In the following program, the non-linear algebraic equation solver, FZERO, is called from within the ordinary differential equation solver, ODE45. Run the problem by issuing the command Prob_8. It calls distill.m, which defines the distillation equation, and vap_press.m, which calculates the vapor pressure.

```
%filename Prob_8.m
%       Binary Batch Distillation
clear
global A B C P T_guess

A = [6.90565 6.95464];
B = [1211.033 1344.8];
C = [220.79 219.482];
P = 1.2*760;            % mmHg
Lo = 100;               % moles
x_start = 0.40;         % moles of toluene
x_final = 0.80;% moles of toluene
T_guess = (80.1+110.6)/2;  % C
xspan = [x_start x_final];
[x L] = ode45('distill',xspan,Lo);
% For Version 4, use
%[x L] = ode45('distill',x_start,x_final,Lo);
plot(x,L,'r')
title('Batch Distillation')
xlabel('Mole Fraction of Toluene')
ylabel('Moles of Liquid')
output = [x L];
save batch.dat output -ascii
```

———————————————————————————

```
%filename distill.m
function dL_dx = distill(x,L)
global A B C P T_guess x2

x2 = x;
T = fzero('vap_press',T_guess);
P_i = 10.^(A-B./(T+C));
k = P_i./P;
dL_dx = L/x2/(k(2)-1);
```

———————————————————————————

```
%filename vap_press.m
function f = vap_press(T)

global A B C P x2

x1 = 1-x2;
P_i = 10.^(A-B./(T+C));
k = P_i./P;
f = 1 - k(1)*x1 - k(2)*x2;
```

The results are:

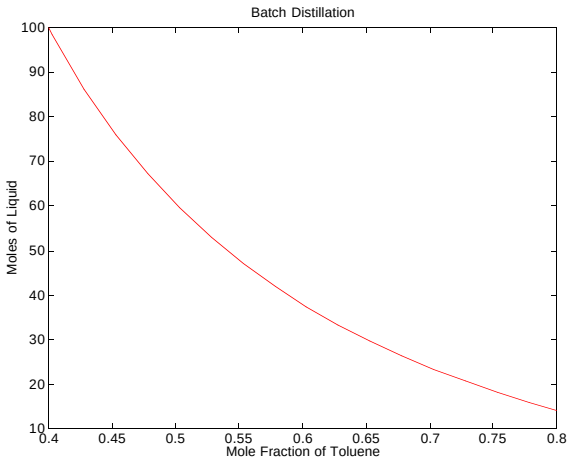


Figure 6. Batch Distillation

MATLAB Problem 9 Solution

To solve this problem, construct a file react.m which defines the differential equation.

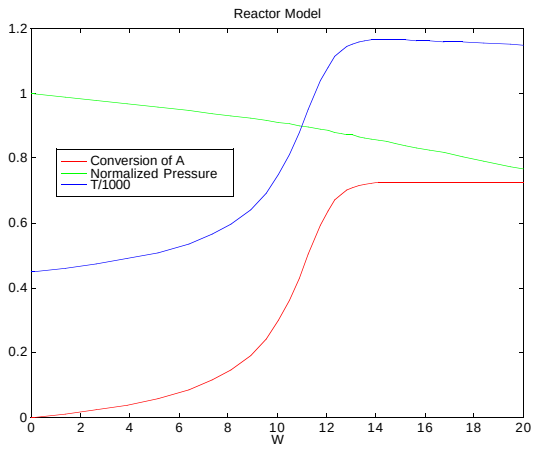
```
%filename react.m
function der=react(W,var)
global Ta delH CPA FA0
x = var(1);
T = var(2);
y = var(3);
k = 0.5*exp(5032*(1/450 - 1/T));
temp = 0.271*(450/T)*y/(1-0.5*x);
CA = temp * (1-x);
CC = temp * 0.5 * x;
Kc = 25000*exp(delH/8.314*(1/450-1/T));
rA = -k*(CA*CA - CC/Kc);
der(1) = -rA/FA0;
der(2) = (0.8*(Ta-T)+rA*delH)/(CPA*FA0);
der(3) = -0.015*(1-0.5*x)*(T/450)/(2*y);
```

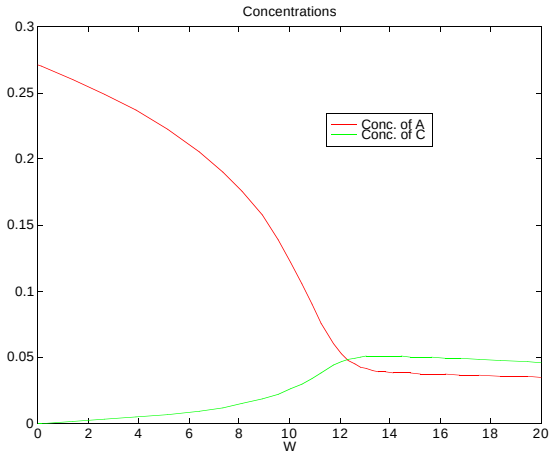
Prob_9.m integrates the equations.

```
%filename Prob_9.m
global Ta delH CPA FA0
%set parameters
Ta = 500;
delH = -40000;
CPA = 40;
FA0=5;

%set initial conditions
var0(1) = 0.;
var0(2) = 450;
var0(3) = 1.0;
Wspan = [0 20];

%integrate equations
[W var]=ode45('react',Wspan,var0)
% For version 4.0 use
%[W var]=ode45('react',0,20,var0)

%plot results
tt = var(:,2)/1000;
plot(W,var(:,1),'r',W,var(:,3),'g',W,tt,'b')
title ('Reactor Model')
xlabel ('W')
legend('Conversion of A','Normalized Pressure','T/1000')
```

To plot the concentrations, use the file conc.m.

```
%filename conc.m
x=var(:,1);
y=var(:,3);
T=var(:,2);
temp = 0.271*(450./T).*y./(1-0.5.*x);
CA = temp .* (1.-x);
CC = temp .* 0.5 .* x;
plot(W,CA,'r',W,CC,'g')
title ('Concentrations')
xlabel('W')
legend ('Conc. of A','Conc. of C')
```





Figures 7 and 8. Model of Chemical Reactor

MATLAB Problem 10 Solution

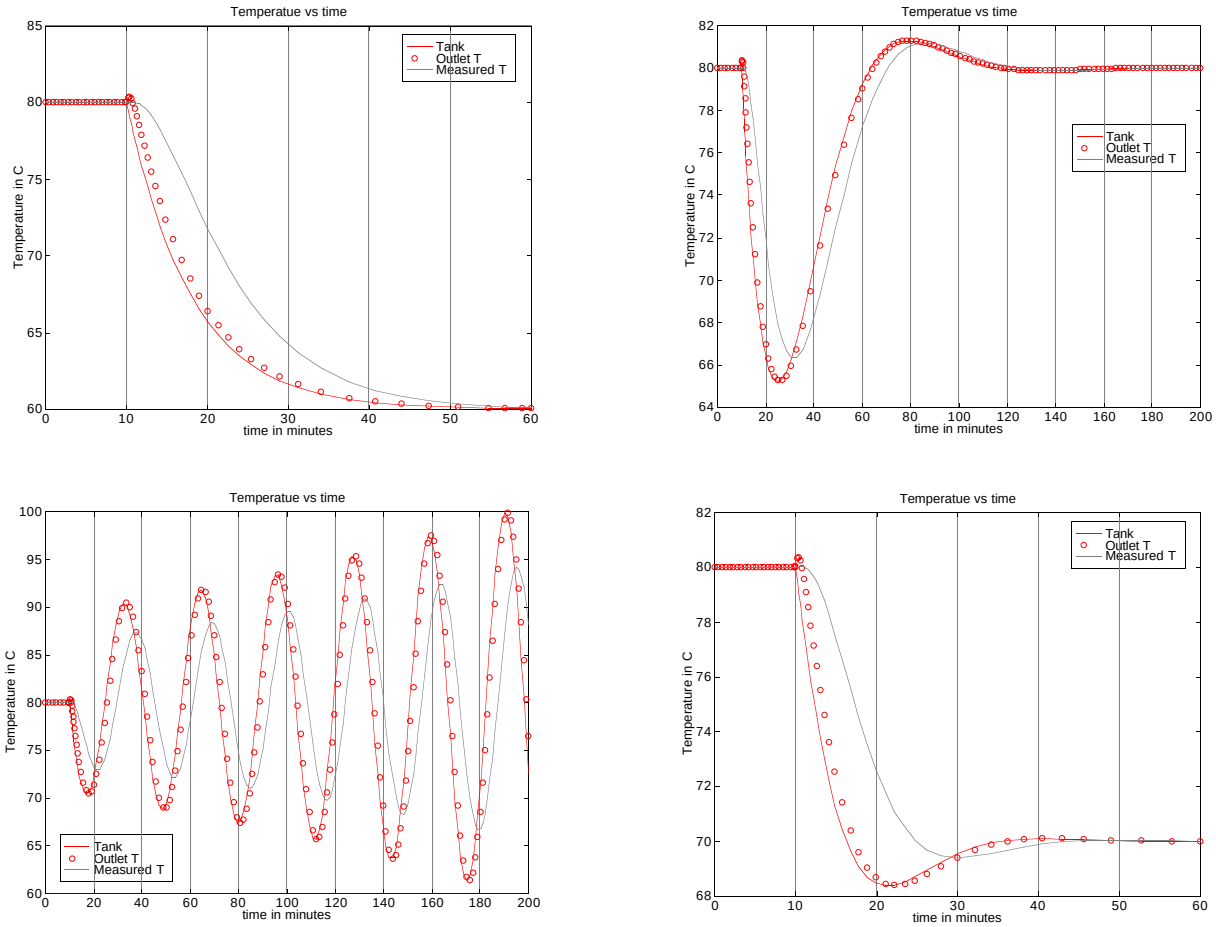
A file, tempdyn.m is constructed to define the equations.

```
%filename tempdyn.m
function Tdot=tempdyn(t,T)
global qsetpt taud taui Kc Tsetpt onoff
%  Use logical block to model the step change at 10 min.
if t<10
    Tinlet = 60;
else
   Tinlet = 40;
end
qin= qsetpt+Kc*(Tsetpt-T(3))+onoff*Kc/taui*T(4);% total heat sent in
% use the following statement for part (e)
%qin=max(0,min(2.6*qsetpt,qin));
row(1)= (500*(Tinlet-T(1))+qin)/(4000);     % energy balance
row(2) = (T(1)-T(2)- 0.5*taud*row(1))*2/taud;
                    % Pade approximation for delay
row(3) = (T(2)-T(3))/5;                     % Thermocouple dynamics
row(4) = Tsetpt - T(3);                     % the error message
% row(4) not needed for part (e), but is calculated anyway
Tdot = row';
```

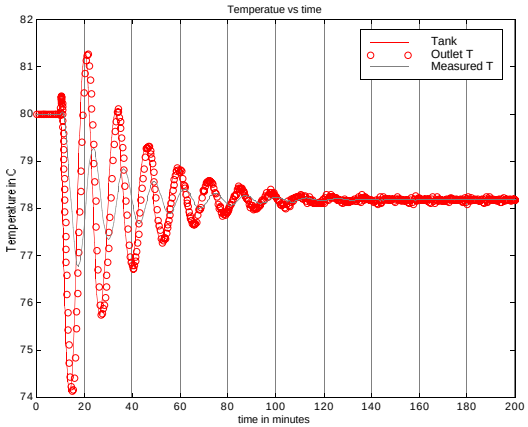
A file Prob_10.m is constructed to run the problem.

```
%filename Prob_10.m
clear all
clc
global qsetpt taud taui Kc Tsetpt onoff

qsetpt= 1e4; taud=1; taui=2; Tsetpt=80;
Kc= input('enter the gain')
onoff=input('enter 0 for no integrator, enter 1 if integrator on')

% initialization
to=0;  tfin=200;   % limits of integration
tspan = [to tfin];
To=[80 80 80 0];%  initial condition of system. Tank Temp, Outlet
                                        %  Temp Thermocouple Temp and error signal
[t,T] = ode45('tempdyn',tspan,To);
% For version 4 use
% [t,T] = ode45('tempdyn',to,tfin,To);

T
plot(t,T(:,1),'r', t,T(:,2),'ro',t,T(:,3),'r:')
grid
title('Temperatue vs time')
xlabel('time in minutes')
ylabel('Temperature in C')
legend('Tank','Outlet T','Measured T')
```

The problems are solved by using the command Prob_10. Input values as follows: first entry is the gain, second one is 1 unless tauc is infinite. Thus
(a) 0,1; (b) 50,1; (c) 500,1; (d) 500,0; (e) 5000,0. The SIMULINK option can also be used.



Figures 9, 10, 11 and 12. Control Problem



Figures 13. Control Problem with a limited response