# Final Examination Solution
**May 2, 2017**

**Problem 1. (50%)**
Consider a continuous stirred-tank reactor operated under isothermal conditions in which the following three reactions take place:

$$A + B \xrightarrow{k_1} C \qquad A + C \xrightarrow{k_2} D \qquad B + C \xrightarrow{k_3} E \tag{1}$$

The rates of the three irreversible reactions are given by an elementary mechanism,

$$r_1 = k_1[A][B] \qquad r_2 = k_2[A][C] \qquad r_3 = k_3[B][C] \tag{2}$$

The material balances for the reactor are given by (accumulation = in – out +/- generation/consumption):

$$\frac{d[A]}{dt} = F[A]_{in} - F[A] - r_1 - r_2 \tag{3.A}$$

$$\frac{d[B]}{dt} = F[B]_{in} - F[B] - r_1 - r_3 \tag{3.B}$$

$$\frac{d[C]}{dt} = F[C]_{in} - F[C] + r_1 - r_2 - r_3 \tag{3.C}$$

$$\frac{d[D]}{dt} = F[D]_{in} - F[D] + r_2 \tag{3.D}$$

$$\frac{d[E]}{dt} = F[E]_{in} - F[E] + r_3 \tag{3.E}$$

Consider a specific case where the volumetric flowrate per unit volume $F$ = 0.1 1/s and the inlet concentrations $[A]_{in} = [B]_{in} = 1.0$ mol/liter & $[C]_{in} = [D]_{in} = [E]_{in} = 0.0$ mol/liter. Assume the reactor is initially empty, $[A] = [B] = [C] = [D] = [E] = 0.0$ mol/liter at t = 0.

An experiment is performed in which the concentrations of the five species are measured as a function of time. This data is recorded in the file located at http://www.utkstair.org/clausius/docs/mse510/data/mse510_xm02_p01_s17.txt . The data file contains six columns corresponding to time, and the concentrations of A, B, C, D and E.

Using this information, determine the optimal values of the three reaction rate constants, $k_1$, $k_2$ and $k_3$. The magnitude of these rate constants should all be on the order of one. Also report the value of the objective function.

**Solution:**

I solved this problem using the Classical Fourth Order Runge-Kutta method to integrate the set of five nonlinear ODEs numerically. I used the amoeba method to perform the multivariate optimization. The objective is the sum of the squares of the error (SSE) between the data points and the model.

To solve this problem I used the following script (based on the script in the notes). The time range of the Runge-Kutta technique, from 0 to 60 seconds, was chosen to cover all of the experimental sampling times. The number of steps, 600, was chosen as a guess. The goodness of the size step can be verified after convergence by reproducing the result with a finer resolution.

script: xm02_p01_s17.m

```
clear all;
close all;
format long;
%
%  define global variables
%

% make initial guesses for unknown variables
global k1 k2 k3
k1 = 1;
k2 = 1;
k3 = 1;
xo_opt = [k1, k2, k3];

%  variables for Runge-Kutta solution of ODEs
global nstep_rk4 to_rk4 tf_rk4 xo_rk4
nstep_rk4 = 600;
to_rk4 = 0.0; % initial time (x)
tf_rk4 = 60.0; % final time (s)
xo_rk4 = [0.0; 0.0; 0.0; 0.0; 0.0]; % initial concentrations of A, B, C, D & E

% this data provides time (sec; column 1), concentrations of A, B, C D & E (columns 2
through 6)
global datamat
datamat = [
 0.000000000000000e+00   0.000000000000000e+00   0.000000000000000e+00   0.000000000000000e+00   0.000000000000000e+00   0.000000000000000e+00
 1.250000000000000e+00   1.118046481051662e-01   1.116623980954711e-01   4.700874673408187e-03   2.851082014856675e-04   4.273582111807591e-04
 2.500000000000000e+00   1.839860402475692e-01   1.816059658652099e-01   2.023393846448672e-02   4.866387931384379e-03   7.246462313743707e-03
...
 4.750000000000000e+01   2.546886949424418e-01   2.069463948721118e-01   4.284358868628287e-02   2.153579070297921e-01   2.631002071001223e-01
 4.875000000000000e+01   2.546894322129148e-01   2.069458149154847e-01   4.284360449373908e-02   2.156960839231346e-01   2.634397012205650e-01
 5.000000000000000e+01   2.546900041005546e-01   2.069453650575034e-01   4.284361675523387e-02   2.159945976983829e-01   2.637392367414344e-01     ];
%
%  call the optimization routine
%
global icount
icount = 0;
tolx = 1.0e-6;
tolf = 1.0e-6;
[f,x] = amoeba(xo_opt,tolx,tolf);
fprintf(1,'code performed %i function evaluations   \n', icount);
%
%  report solution
%
k1 = x(1);
k2 = x(2);
k3 = x(3);
fprintf(1,' k1 = %15.7e  \n', k1);
fprintf(1,' k2 = %15.7e  \n', k2);
fprintf(1,' k3 = %15.7e  \n', k3);
fprintf(1,' fobj = %15.7e \n', f);
```

```
%
%  evaluate model with converged parameters
%

[xplot,yplot]=rk4n(nstep_rk4,to_rk4,tf_rk4,xo_rk4);
%
%  plot model versus data
%
% plot concentrations
figure(10)
plot(datamat(:,1),datamat(:,2),'ko');
hold on;
plot(xplot(:),yplot(:,1),'k-');
hold on;
plot(datamat(:,1),datamat(:,3),'ro');
hold on;
plot(xplot(:),yplot(:,2),'r-');
hold on;
plot(datamat(:,1),datamat(:,4),'bo');
hold on;
plot(xplot(:),yplot(:,3),'b-');
hold on;
plot(datamat(:,1),datamat(:,5),'go');
hold on;
plot(xplot(:),yplot(:,4),'g-');
hold on;
plot(datamat(:,1),datamat(:,6),'mo');
hold on;
plot(xplot(:),yplot(:,5),'m-');
hold off;
xlabel('time (s)')
ylabel('concentration')
legend('A-data','A-model','B-data','B-model','C-data','C-model','D-data','D-model','E-data','E-model')
```

Second, I also modified the `amoeba.m` input function as follows:

```
function fobj = funkeval(x)
%  get parameters
global k1 k2 k3
k1 = x(1);
k2 = x(2);
k3 = x(3);
%  get data
global datamat
ndata = max(size(datamat));
timevec(1:ndata) = datamat(1:ndata,1);
% evaluate model
global nstep_rk4 to_rk4 tf_rk4 xo_rk4
[xplot,yplot]=rk4n(nstep_rk4,to_rk4,tf_rk4,xo_rk4);
% compute objective function
fobj = 0.0;
for i = 1:1:ndata
     yinterp(1:5) = interp1(xplot,yplot(:,1:5),timevec(i));
     for m = 1:1:5
        fobj = fobj + (datamat(i,m+1) - yinterp(m))^2;
     end
end
fprintf(1, '  %15.7e %15.7e %15.7e %15.7e \n', k1, k2, k3, fobj);
global icount
icount = icount + 1;
```

Third, I also modified the `rk4.m` input function as follows:

```
 function dydt = funkeval(t,y)
 global k1 k2 k3
 A = y(1);
 B = y(2);
 C = y(3);
 D = y(4);
 E = y(5);
%
%  inlet composition
%
Ain = 1.0;
Bin = 1.0;
Cin = 0.0;
Din = 0.0;
Ein = 0.0;
%
% flowrate over volume
%
F = 0.1; %
%
%  rxn 1:  A + B -->  C
%  rxn 2:  A + C -->  D
%  rxn 3:  B + C -->  E
rate1 = k1*A*B;
rate2 = k2*A*C;
rate3 = k3*B*C;
%
dAdt = Ain*F - A*F - rate1 - rate2;
dBdt = Bin*F - B*F - rate1 - rate3;
dCdt = Cin*F - C*F + rate1 - rate2 - rate3;
dDdt = Din*F - D*F + rate2;
dEdt = Ein*F - E*F + rate3;
%
dydt(1) = dAdt;
dydt(2) = dBdt;
dydt(3) = dCdt;
dydt(4) = dDdt;
dydt(5) = dEdt;
```

At the command line prompt, I called the script:

```
>> xm02_p01_s17
```

which generated the following output and plot.

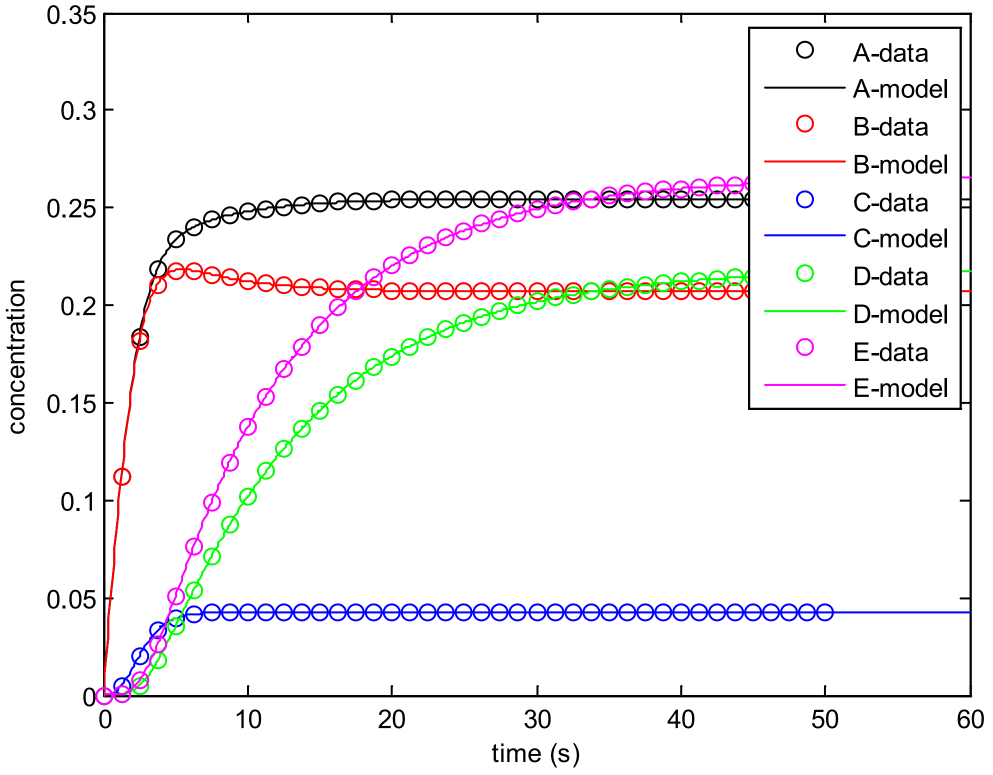
```
    1.0000000e+00   1.0000000e+00   1.0000000e+00   2.0100666e-01
    1.0100000e+00   1.0000000e+00   1.0000000e+00   2.0207576e-01
    1.0000000e+00   1.0100000e+00   1.0000000e+00   2.0255486e-01
    1.0000000e+00   1.0000000e+00   1.0100000e+00   1.9569036e-01
i =    1   1.0000000e+00   1.0000000e+00   1.0000000e+00 f =   2.0100666e-01
i =    2   1.0100000e+00   1.0000000e+00   1.0000000e+00 f =   2.0207576e-01
i =    3   1.0000000e+00   1.0100000e+00   1.0000000e+00 f =   2.0255486e-01
i =    4   1.0000000e+00   1.0000000e+00   1.0100000e+00 f =   1.9569036e-01
```

```
    1.0066667e+00   9.9000000e-01   1.0066667e+00   1.9672664e-01
    9.9444444e-01   9.9333333e-01   1.0111111e+00   1.9364643e-01
...
    9.9999721e-01   1.9999937e+00   2.9999909e+00   3.1821196e-09
    9.9999728e-01   1.9999936e+00   2.9999905e+00   3.1821208e-09
    9.9999725e-01   1.9999937e+00   2.9999907e+00   3.1821189e-09
code performed 224 function evaluations
 k1 =   9.9999721e-01
 k2 =   1.9999937e+00
 k3 =   2.9999909e+00
 fobj =   3.1821196e-09

Based on this analyis, the optimal values of the rate constants are
```

$k_1 = 1$ $k_2 = 2$ $k_3 = 3$ and the objective function (SSE) is 3.2x10$^{-9}$.

The units of these second-order reaction rate constants are liter/(mole-sec).

The script also generated the following plot, showing the agreement between the model evaluated at the converged parameters and the data.



**Problem 2. (50%)**

The one-dimensional heat equation can describe heat transfer in a rutile $TiO_2$ with both heat conduction and radiative heat loss.

$$\frac{\partial T}{\partial t} = \frac{k}{\rho C_p}\frac{d^2T}{dz^2} - \frac{\varepsilon\sigma S}{\rho C_p}\left(T^4 - T_s^{\ 4}\right)$$

where the following variables [with units] are given as

temperature in the material $T$ [K]
surrounding temperature $T_s = 800$ [K]
axial position along material $z$ [m]
thermal conductivity $k = 9.0$ [J/K/m/s]
mass density $\rho =$ 4250 [kg/m$^3$]
heat capacity $C_p = 711.3$ [J/kg/K]
Stefan–Boltzmann constant $\sigma = 5.670373x10^{-8}$ [J/s/m$^2$/K$^4$]
gray body permittivity $\varepsilon = 0.79$
surface area to volume ratio $S = 200$ [m$^{-1}$] (for a cylindrical rod of diameter 0.05 m)

A rutile $TiO_2$ cylindrical rod of diameter 0.05 m and length 0.4 m is initially at $T(z,t=0) = 1000$ K. One end of the rod is maintained at $T(z=0,t) = 1000$ K. The other end of the rod is maintained at $T(z=L,t) = 1500$ K.

(a) Sketch the transient behavior.
(b) Find the approximate steady-state temperature in the material at z=0.2 m.

**Solution:**

This is a single non-linear parabolic PDE with one spatial dimension and a Dirichlet boundary condition at *z*=0 and a Dirichlet boundary condition at *z*=0.4. To solve this problem, I will use the code `parapde_1_anyBC.m`.

I modified the input functions in `parapde_1_anyBC.m` as follows.

I assigned the appropriate type of boundary conditions.

```
BC(1)  =  'D';
BC(2)  =  'D';
```

I set the final time to 1000 seconds and chose dt to be 0.1 seconds, so I had 10,000 temporal intervals.

```
% discretize time
to = 0;
tf = 1.0e+3;
```

```
dt = 1.0e-1;
```

The rod spans from 0 to 0.4 meter. I set dx to be 0.004 m, so I had 100 spatial intervals.

```
% discretize space
xo = 0;
xf = 0.4;
dx = 4.0e-3;
```

I defined the PDE in the following function.

```
%
%  function defining PDE
%
function k = pdefunk(x,t,y,dydx,d2ydx2);
%
Temp = y;
% rho = density [kg/m^3]
rho = 4250.0;
% Cp = heat capacity [J/kg/K]
Cp = 711.3;
% k = thermal conductivity [W/m/K]
k = 9.0;
%  alpha = thermal diffusivity
alpha = k/rho/Cp;
% length of rod [m]
L = 0.4;
% diameter in [m]
radius = 0.025;
diameter = 2.0*radius;
% surface Area in [m^2]
Area = pi*diameter*L;
% Volume in [m^3]
Volume = pi/4*diameter^2*L;
% surface area to volume ratio
S = Area/Volume;
%  Temperature of the surroundings [K]
Tsurround = 800.0;
% Stefan-Boltzmann constant [J/s/m^2/K^4]
sigma = 5.670373e-8;
% gray body permittivity [dimensionless]
eps = 0.79;
fac = eps*sigma*S/(rho*Cp);
k = alpha*d2ydx2 - fac*(Temp^4 - Tsurround^4);
```

I defined the IC and BCs in the functions below.

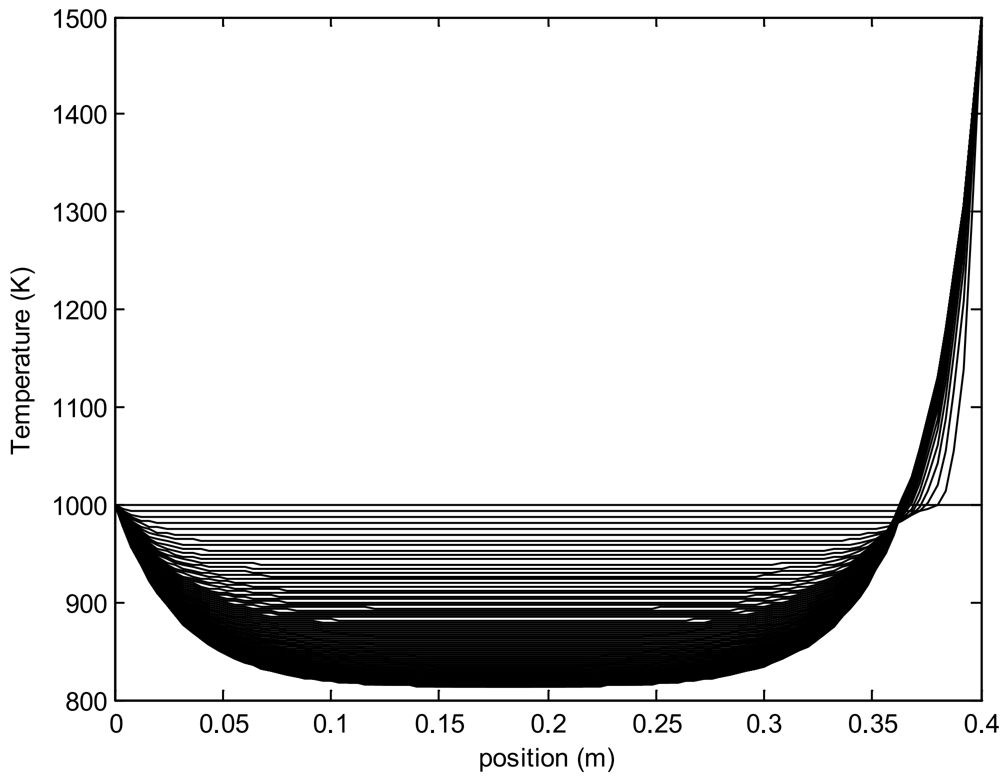
```
%
%  function defining initial condition
%

function ic = icfunk(x);
ic = 1000;
```

```
%
%  functions defining LHS boundary condition
%

function f = aBCo(t);
f = 1;

function f = bBCo(t);
f = 0;

function f = cBCo(t);
f = -1000;

%
%  functions defining RHS boundary condition
%

function f = aBCf(t);
f = 1;

function f = bBCf(t);
f = 0;

function f = cBCf(t);
f = -1500;
```

At the command line prompt, I typed

```
[xvec,tvec,Tmat] = parapde_1_anyBC;
```

This command generated the following plot.



To determine if this system is at steady state, I examine the last values of the midpoint temperature.

I first remind myself of the size of the solution matrix, Tmat. This helps me identify the spatial and temporal dimensions.

```
>> whos Tmat
  Name        Size                 Bytes  Class     Attributes
  Tmat      103x10001            8240824  double
```

To find the last values at x = 0.2 m, I confirmed that I knew the correct spatial index.

```
>> xvec(52)

ans =   0.200000000000000
```

I then printed out the temperature at x = 0.2 m at the last step (t = 1000 seconds).

```
>> Tmat(52,10001)

ans =   8.134692477890361e+02
```

The temperature at the midpoint is 813.5 K at the end of 1000 s.

If I rerun the code out to 2000 s, keeping everything else the same,

```
>> Tmat(52,20001)

ans =     8.029265563603503e+02
```

The temperature at the midpoint is 802.9 K at the end of 2000 s.

If I rerun the code out to 4000 s, keeping everything else the same,

```
>> Tmat(52,40001)

ans =     8.021668441351183e+02
```

The temperature at the midpoint is 802.2 K at the end of 4000 s.

If I rerun the code out to 8000 s, keeping everything else the same,

```
>> Tmat(52,80001)

ans =     8.021634809897909e+02
```

The temperature at the midpoint is 802.2 K at the end of 8000 s.

The two answers agree to four digits, so we are pretty close to the steady state solution, where the midpoint temperature is 802.2 K.