Final Examination Solutions
May 6, 2019

1. A System of Non-Linear Parabolic PDEs

Consider a plug flow reactor. (This is a pipe with a reaction taking place in the fluid flowing inside it.) Consider the irreversible dimerization reaction 2A → B taking place in a non-reactive solvent. The molar balance for each component, A and B, is given by

$$\frac{\partial C_i}{\partial t} = -v_z \frac{dC_i}{dz} + D_i \frac{d^2 C_i}{dz^2} + \nu_i r$$

where *z* is the spatial dimension in the axial direction, *t* is time, *C_i* is the molar concentration of species i, v_z is the axial velocity, D_i is the diffusion coefficient of species i, ν_i is the stochiometric for species i, (namely -2 for A +1 for B) and *r* is the reaction rate. The reaction rate is given by

$$r = kC_A^2$$

where k is the rate constant. Assume the reactor is operated isothermally so we have no need for an energy balance. The circular pipe is 10 m long with a diameter of 0.1 m. The axial velocity is 0.15 m/s. The diffusivities are all 2.0x10^{-9} m^2/s. The rate constant is $k = 1.0x10^{-5}\frac{m^3}{mol \cdot s}$. Initially, the pipe contains nothing but solvent. At the inlet, the reactants, A is fed in at a concentration of 1200.0 mol/m^3 respectively. No B is present in the feed stream. At the outlet, assume the concentrations no longer change (i.e. a no flux boundary condition).

(a) Solve the problem. Estimate how long it takes this reactor to get to steady state.
(b) Show the steady state profile.
(c) What is the fractional consumption of A at steady state? Reminder: $Y_A = \frac{C_{A,in} - C_{A,out}}{C_{A,in}}$.
(d) What can be done to the axial velocity to increase the fractional consumption? How does this impact the amount of product, B, made per hour, i.e. the through-put? Reminder: $Q_B = v_z A_x C_{B,out}$ where A_x is the cross-sectional area of the pipe.

Solution:

This is a system of two coupled non-linear parabolic PDEs with one spatial dimension and Dirichlet boundary conditions at z=0 and a Neumann boundary conditions at z=10 m. Moreover, this is a system in which convection is dominant. Therefore, to solve this problem, I will use the code `parapde_n_anyBC_flow.m`.

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

I defined the number of PDEs.

```
%  define number of PDEs
neq = 2;
```

I assigned the appropriate type of boundary conditions.

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

I set the final time to 300 seconds and chose dt to be 0.1 seconds, so I had 3000 temporal intervals.

```
% discretize time
to = 0;
tf = 3.0e+2;
dt = 1.0e-1;
```

The rod spans from 0 to 10.0 meter. I set dx to be 1.0 m, so I had 10 spatial intervals.

```
% discretize space
xo = 0;
xf = 10.0;
dx = 1.0e-0;
```

I defined the PDE in the following function.

```
function dydt_out = pdefunk(x,t,y,dydx,d2ydx2,keq);
% molar concentrations [mol/m^3]
CA = y(1);
CB = y(2);
% velocity [m/s]
v = 0.15;
% diffusivity [m^2/s]
D = 2.0e-9;
DA = 1.0*D;
DB = 1.0*D;
%  rate constant [m^6/mol^2/s]
k = 1.0e-5;
% stoichiometric coefficients
```

```
nuA = -2.0;
nuB = 1.0;
% reaction rate [mol/m^3/s]
rate = k*CA*CA;
dydt(1) = -v*dydx(1) + DA*d2ydx2(1) + nuA*rate;
dydt(2) = -v*dydx(2) + DB*d2ydx2(2) + nuB*rate;
dydt_out = dydt(keq);
```

I defined the IC and BCs in the functions below.

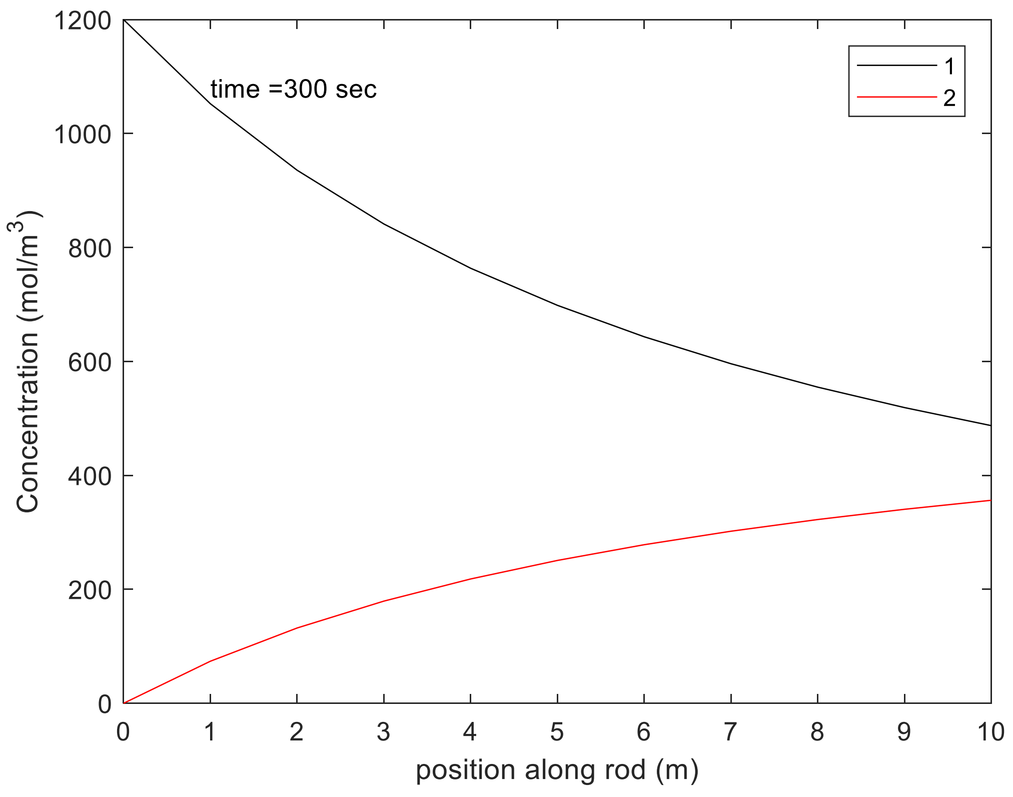
```
%
%  function defining initial condition
%

function ic_out = icfunk(x,keq);
ic(1) = 0.0;
ic(2) = 0.0;
ic_out = ic(keq);

%
%  functions defining LHS boundary condition
%

function fout = aBCo(t,k);
f(1) = 1.0;
f(2) = 1.0;
fout = f(k);

function fout = bBCo(t,k);
f(1) = 0.0;
f(2) = 0.0;
fout = f(k);

function fout = cBCo(t,k);
f(1) = -1200.0;
f(2) = 0.0;
fout = f(k);

%
%  functions defining RHS boundary condition
%

function fout = aBCf(t,k);
f(1) = 0.0;
f(2) = 0.0;
fout = f(k);

function fout = bBCf(t,k);
f(1) = 1.0;
f(2) = 1.0;
fout = f(k);

function fout = cBCf(t,k);
f(1) = 0.0;
```

```
f(2) = 0.0;
fout = f(k);
```

At the command line prompt, I typed

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

This command generated the following final frame of a movie.



(a) Solve the problem. Estimate how long it takes this reactor to get to steady state.

If a pipe (plug flow reactor) is L = 10 m long and the velocity is 0.15 m/s, then the "residence time of the reactor" is given by $\tau = \frac{L}{v_z}$, (66.67 s in this example) which sets a lower bound for reaching steady state. Maybe it takes two residence times to reach steady state. We can examine the concentration of A at the outlet as a function of time.

First, make sure you are plotting the correct variables.

The twelfth spatial index is the end of the pipe.

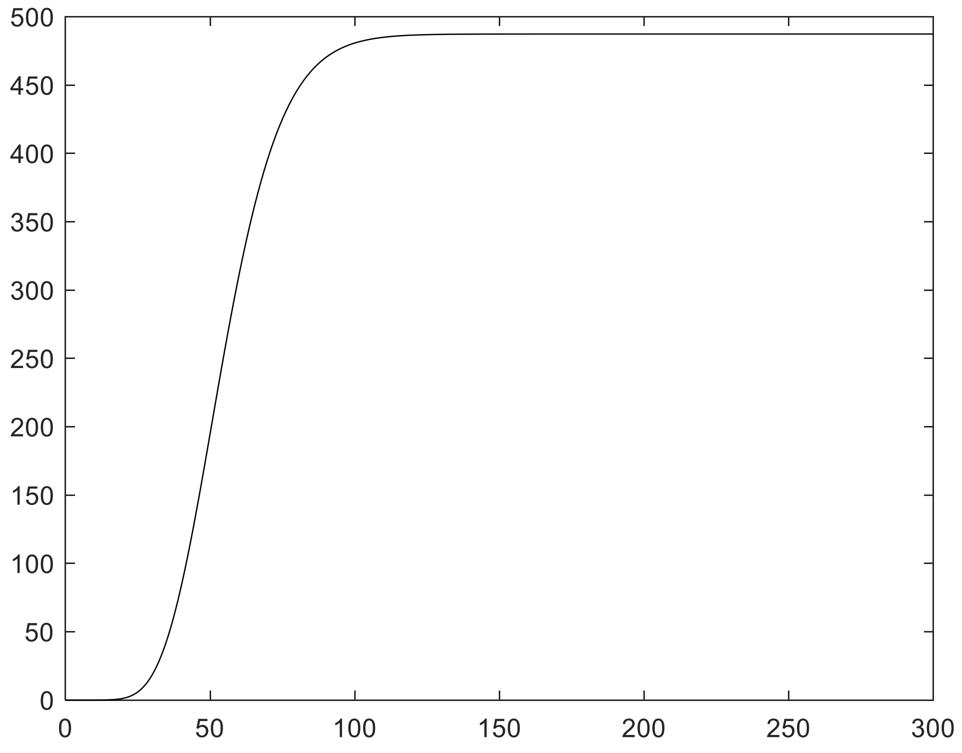
```
>> xvec(12)
ans =    10
```

The solution matrix, Tmat, has three indices, space, time, and equation respectively.

```
>> whos Tmat
  Name        Size                    Bytes  Class     Attributes

  Tmat      13x3001x2                624208  double
```

This command plots the first concentration at the twelfth spatial node for all times.

```
>> plot(tvec,Tmat(12,:,1),'k-')
```



The system has pretty clearly reach steady state by 150 sec.

(b) Show the steady state profile.

The steady state profile is shown above in the final frame of the movie, since the PDEs were solved out to steady state.

(c) What is the fractional consumption of A at steady state? Reminder: $Y_A = \frac{C_{A,in} - C_{A,out}}{C_{A,in}}$.

The reactant A is input at 1200 mol/m^3 and, at the end of the simulation, emerges at

```
>> Tmat(12,3001,1)
ans =  487.3180
```

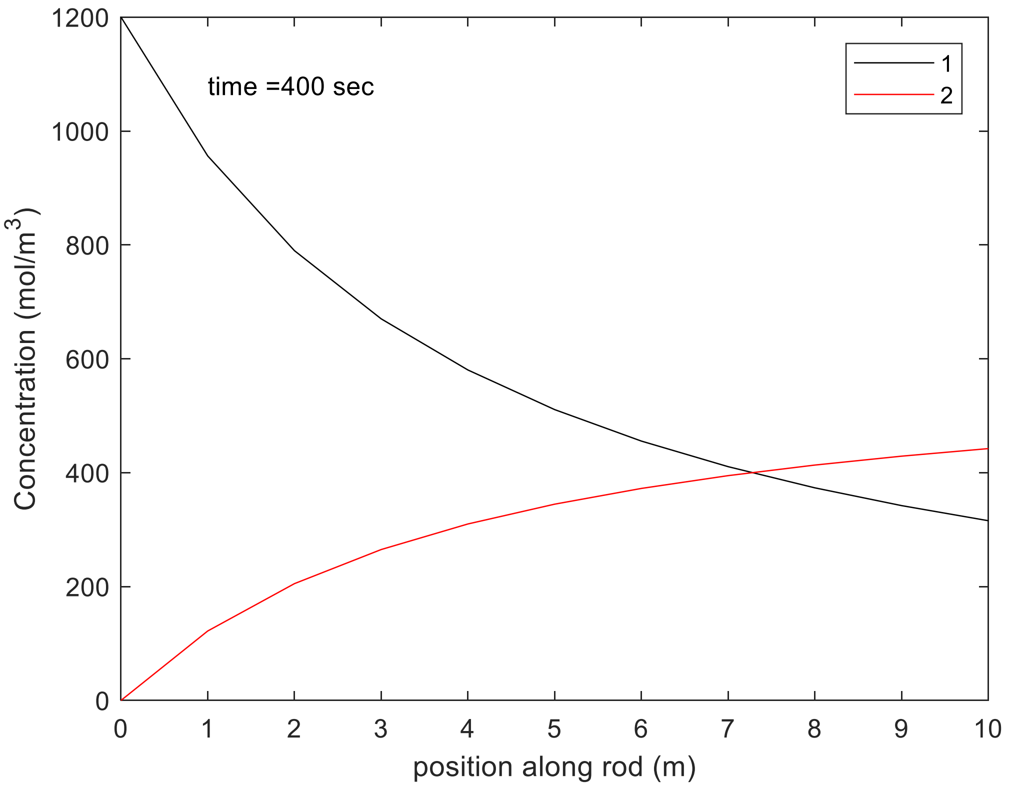
So the fractional consumption is

$$Y_A = \frac{C_{A,in} - C_{A,out}}{C_{A,in}} = \frac{1200 - 487.3}{1200} = 59.39\%$$

(d) What can be done to the axial velocity to increase the fractional consumption? How does this impact the amount of product, B, made per hour, i.e. the through-put? Reminder: $Q_B = v_z A_x C_{B,out}$ where A_x is the cross-sectional area of the pipe.

Slowing down the axial velocity will increase the residence time and improve the fraction yield.

If we run the code again cutting the velocity in half and increasing the final simulated time to 400 s, then we have the following final frame of the movie.



The concentration of A at the pipe outlet at steady state is

```
>> Tmat(12,4001,1)
ans =  315.5597
```

So the fractional yield is

$$Y_A = \frac{C_{A,in} - C_{A,out}}{C_{A,in}} = \frac{1200 - 315.6}{1200} = 73.70\%$$

Reducing the axial velocity increases the fractional consumption of the reactants.

The through-put is the amount of B produced per unit time.

$$Q_B = v_z A_x C_{B,out} = v(\pi r^2) C_{B,out}$$

For the first case with the higher velocity, using the final concentration of B, we have

```
>> Tmat(12,3001,2)
ans =  356.3410
```

$$Q_A = v A_X C_{A,out} = v(\pi r^2) C_{A,out} = 0.15(\pi r^2)356.34 = (\pi r^2)53.45$$

For the second case with the lower velocity, using the final concentration of B, we have

```
>> Tmat(12,4001,2)
ans =  315.5597
```

$$Q_A = v A_X C_{A,out} = v(\pi r^2) C_{A,out} = 0.075(\pi r^2)442.22 = (\pi r^2)33.17$$

The ratio of the high-velocity to low-velocity through-puts is 1.61. So the higher velocity has a lower fractional yield but a higher through-put.

2. Multivariate Nonlinear Optimization

Download the data located at
http://utkstair.org/clausius/docs/mse510/data/mse510_xm02_p02.txt
The first column corresponds to wavelength, x. The second column corresponds to signal intensity, y.

Perform a multivariate nonlinear optimization in order to fit this data to **two** weighted Gaussian curves. The equation for a normalized Gaussian is

$$f_G(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

where μ is the mean and σ is the standard deviation. Your model should take the form

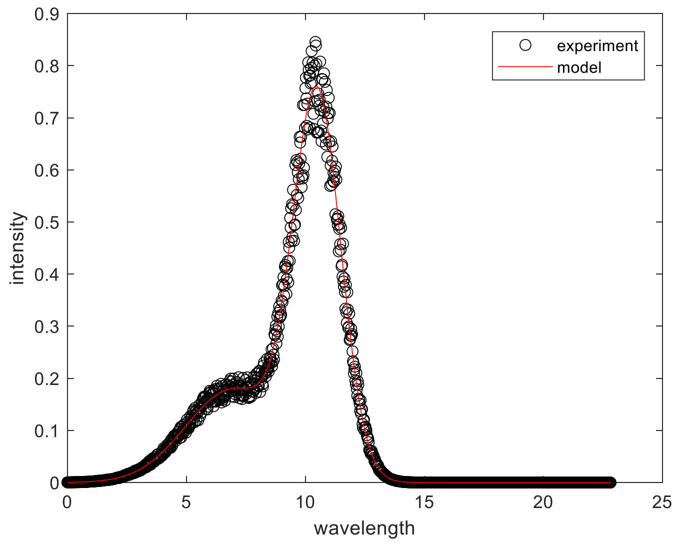
$$y_{model} = w_1 f_G(x;\mu_1,\sigma_1) + w_2 f_G(x;\mu_2,\sigma_2)$$

where w is the weighting constant for each Gaussian.

Determine the optimal values of the weighting constant, the mean and the standard deviation for each Gaussian.

Solution:

I first plotted the data in order to get a good idea of an initial guess for the mean and variance.



From this plot, I chose initial guesses of 1.0, 5.0 and 3.0 for the weight, mean and standard deviation of one of the Gaussians and for the three means and 1.0, 12.0 and 2.0 for the weight, mean and standard deviation of the other Gaussian. (These are not particularly good initial guesses.)

I used the following script, driver_2019_xm02p02.m, to perform the optimization with the amoeba method and plot the result.

```
clear all;
close all;
format long;

global datamat
datamat = [0.00    0.000408181
0.02    0.000415796
0.04    0.000419199
... 1000+ lines of data omitted here ...
22.78   4.83169E-15
22.8    4.71657E-15
22.82   4.56922E-15];

%
%  initial guesses
%
xo(1) = 1.0; % first weight
xo(2) = 5.0; % first mean
xo(3) = 3.0; % first standard deviation
xo(4) = 1.0; % first weight
xo(5) = 12.0; % first mean
xo(6) = 2.0; % first standard deviation

%
% call amoeba for optimization
%
[f,x,iter] = amoeba(xo,1.0e-6,1.0e-6)

%
% evaluate model with optimized parameters
%
ngauss = 2;
w(1) = x(1);
mu(1) = x(2);
sig(1) = x(3);
w(2) = x(4);
mu(2) = x(5);
sig(2) = x(6);
%
ndata = max(size(datamat));
xvec(1:ndata) = datamat(1:ndata,1);
yexp(1:ndata) = datamat(1:ndata,2);
fac = sqrt(2.0*pi);
for i = 1:1:ndata
    ymod(i) = 0.0;
    for j = 1:1:ngauss
       ymod(i) = ymod(i) + w(j)/(sig(j)*fac)*exp(-((xvec(i)-mu(j))^2)/(2.0*sig(j)^2));
    end
end

%
%  plot data and solution
```

```
%
figure(1);
plot(xvec(:),yexp(:),'ko');
hold on;
plot(xvec(:),ymod(:),'r-');
hold off;
legend('experiment','model')
ylabel('intensity');
xlabel('wavelength');
```

The objective function used by amoeba.m contains the same content as this script, namely

```
function fobj = funkeval_prob(x)
%
%convert variables
ngauss = 2;
w(1) = x(1);
mu(1) = x(2);
sig(1) = x(3);
w(2) = x(4);
mu(2) = x(5);
sig(2) = x(6);
%
global datamat

ndata = max(size(datamat));
xvec(1:ndata) = datamat(1:ndata,1);
yexp(1:ndata) = datamat(1:ndata,2);
fac = sqrt(2.0*pi);
for i = 1:1:ndata
    ymod(i) = 0.0;
    for j = 1:1:ngauss
        ymod(i) = ymod(i) + w(j)/(sig(j)*fac)*exp(-((xvec(i)-mu(j))^2)/(2.0*sig(j)^2));
    end
end
fobj = 0.0;
for i = 1:1:ndata
    fobj = fobj + (yexp(i) - ymod(i))^2;
end
```

At the command line prompt, I typed:

```
>> driver_2019_xm02p02
```

This generated the following output.

```
i =    1   1.0000000e+00   5.0000000e+00   3.0000000e+00   1.0000000e+00   1.2000000e+01   2.0000000e+00 f =    3.2034128e+01
i =    2   1.5000000e+00   5.0000000e+00   3.0000000e+00   1.0000000e+00   1.2000000e+01   2.0000000e+00 f =    3.1907926e+01
i =    3   1.0000000e+00   7.5000000e+00   3.0000000e+00   1.0000000e+00   1.2000000e+01   2.0000000e+00 f =    2.4637052e+01
i =    4   1.0000000e+00   5.0000000e+00   4.5000000e+00   1.0000000e+00   1.2000000e+01   2.0000000e+00 f =    3.0777992e+01
i =    5   1.0000000e+00   5.0000000e+00   3.0000000e+00   1.5000000e+00   1.2000000e+01   2.0000000e+00 f =    2.7497234e+01
i =    6   1.0000000e+00   5.0000000e+00   3.0000000e+00   1.0000000e+00   1.8000000e+01   2.0000000e+00 f =    5.8630095e+01
i =    7   1.0000000e+00   5.0000000e+00   3.0000000e+00   1.0000000e+00   1.2000000e+01   3.0000000e+00 f =    3.4626772e+01

f =   0.298179364165348

x =
  Columns 1 through 3
```

```
   0.893981456484229   6.974082099302574   1.987878576798020
  Columns 4 through 6
   1.812625724433016  10.493582054819676   1.000474985186837

iter =   335
```

The method converged in 335 iterations. The values of the parameters are given by

x = [0.8940 6.9741 1.9879 1.8126 10.4936 1.0005]

Thus $w_1 = 0.894$, $\mu_1 = 6.974, \sigma_1 = 1.988$, $w_2 = 1.813, \mu_2 = 10.494, \sigma_2 = 1.000$

The script can be modified to call the conjugate gradient method instead of the amoeba method. The call to the amoeba function should be replaced with the following line.

```
[f,x] = polak(xo,1.0e-6,1,'min')
```

The method converged in 64 iterations. The values of the parameters are the same as those generated by the amoeba method.

x = [0.8940 6.9741 1.9879 1.8126 10.4936 1.0005]