Homework Assignment Number Five Solutions

Problem (1) Single Non-Linear Parabolic PDE

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

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

where the following variables [with units] are given as

temperature in the material T [K]
surrounding temperature $T_s = 300$ [K]
axial position along material z [m]
thermal conductivity $k = 401$ [J/K/m/s] (for Cu)
mass density $\rho =$ 8960 [kg/m^3] (for Cu)
heat capacity $C_p = 384.6$ [J/kg/K] (for Cu)
Stefan–Boltzmann constant $\sigma = 5.670373x10^{-8}$ [J/s/m^2/K^4]
gray body permittivity $\varepsilon = 0.15$ (for dull Cu)
surface area to volume ratio $S = 200$ [m^{-1}] (for a cylindrical rod of diameter 0.01 m)

A cylindrical Cu rod of diameter 0.01 m and length 0.1 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 insulated, $\left.\frac{dT}{dz}\right|_{z=0.1} = 0$ K/m.

(a) Plot the transient behavior.
(b) Find the approximate steady-state temperature in the material at z=0.1 m.

Solution:

This is a single non-linear parabolic PDE with one spatial dimension and a Dirichlet boundary condition at *z*=0 and a Neumann boundary condition at *z*=0.1. 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) = 'N';
```

I set the final time to 100 seconds and chose dt to be 0.1 seconds, so I had 1000 temporal intervals.

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

The rod spans from 0 to 0.1 meter. I set dx to be 0.005 m, so I had 20 spatial intervals.

```
% discretize space
xo = 0;
xf = 0.1;
dx = 5.0e-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 = 8960.0;
% Cp = heat capacity [J/kg/K]
Cp = 384.6;
% k = thermal conductivity [W/m/K]
k = 401.0;
%  alpha = thermal diffusivity
alpha = k/rho/Cp;
% length of rod [m]
L = 0.1;
% diameter in [m]
radius = 0.1;
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 = 300.0;
% Stefan-Boltzmann constant [J/s/m^2/K^4]
sigma = 5.670373e-8;
% gray body permittivity [dimensionless]
eps = 0.15;
fac = eps*sigma*S/(rho*Cp);
k = alpha*d2ydx2 - fac*(Temp^4 - Tsurround^4);
```

I defined the IC and BCs in the functions below.

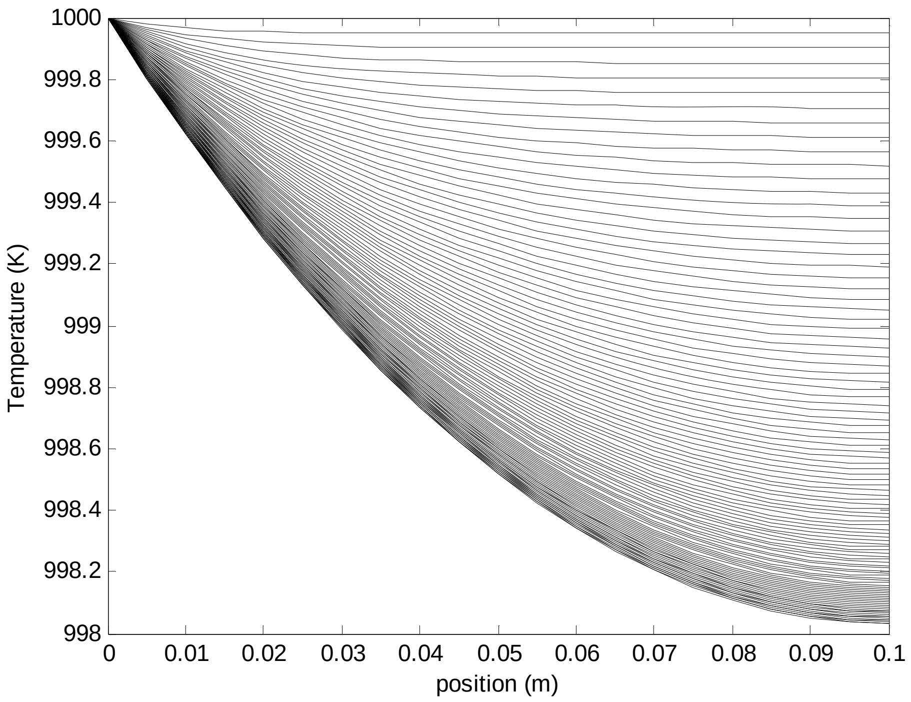
```
%
%  function defining initial condition
%

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 = 0;

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

function f = cBCf(t);
f = 0;
```

At the command line prompt, I typed

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

This command generated the following plot.



To find the last value at x = 0.1 m, I confirmed that I knew the correct spatial and temporal indices.

```
>> xvec(22)
ans =    0.1000

>> tvec(1001)
ans =   100

>> Tmat(22,1001)
ans =  998.0308
```

Therefore the temperature at the end at 100 seconds is 998.03 K.

I don't know that this is steady state. I can run the simulation longer. If I change nothing but the final time to 1000 seconds, then I generate the data point

```
>> Tmat(22,10001)
ans =  997.9108
```

Therefore the temperature at the end at 1000 seconds is 997.91 K.
The two answers agree to three digits, so we are pretty close to the steady state solution, but we can run for a longer time. If I change nothing but the final time to 5000 seconds, then I generate the data point

```
>> Tmat(22,50001)
ans =  997.9108
```

Therefore the temperature at the end at 5000 seconds is 997.91 K.

Problem (2) 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 reaction

A + 2B --> C + 2D

taking place in a non-reactive solvent.

The molar balance for each component is given by

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

where *z* is the spatial dimension in the axial direction, *t* is time, C_i is the molar concentration of species *i*, *v* is the axial velocity, D_i is the diffusion coefficient of species *i*, ν_i is the stochiometric for species *i*, (namely -1, -2, +1, +2 and 0 for A, B, C, D, and S respectively) and r is the reaction rate. The reaction rate is given by

$$r = kC_A C_B^2$$

where k is the rate constant. Assume the reactor is operated isothermally so we have no need for an energy balance.

The pipe is 10 m long with a diameter of 0.1 m. The velocity is 0.1 m/s. The diffusivities are all $1.0\text{x}10^{-9}$ m^2/s. The rate constant is $k = 1x10^{-7}\frac{m^6}{mol^2 \cdot s}$. Initially, the pipe contains nothing but solvent. At the inlet, the reactants, A and B, are fed in at 1000.0 and 2000.0 mol/m^3 respectively. No C or D 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 fraction of the reactants are used, i.e. what is the fractional yield?
(d) What can be done to the velocity to increase the fractional yield? How does this impact the amount of product made per hour, i.e. the through-put?

Solution:

This is a system of four 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 assigned the appropriate type of boundary conditions.

```
BC(1,1) = 'D';
BC(2,1) = 'N';
BC(1,2) = 'D';
BC(2,2) = 'N';
BC(1,3) = 'D';
BC(2,3) = 'N';
BC(1,4) = 'D';
BC(2,4) = '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);
CC = y(3);
CD = y(4);
% velocity [m/s]
v = 0.1;
% diffusivity [m^2/s]
D = 1.0e-9;
DA = 1.0*D;
DB = 1.0*D;
DC = 1.0*D;
DD = 1.0*D;
%  rate constant [m^6/mol^2/s]
k = 1.0e-7;
% stoichiometric coefficients
nuA = -1.0;
nuB = -2.0;
nuC = 1.0;
nuD = 2.0;
% reaction rate [mol/m^3/s]
rate = k*CA*CB*CB;
dydt(1) = -v*dydx(1) + DA*d2ydx2(1) + nuA*rate;
dydt(2) = -v*dydx(2) + DB*d2ydx2(2) + nuB*rate;
dydt(3) = -v*dydx(3) + DC*d2ydx2(3) + nuC*rate;
dydt(4) = -v*dydx(4) + DD*d2ydx2(4) + nuD*rate;
dydt_out = dydt(keq);
```

I defined the IC and BCs in the functions below.

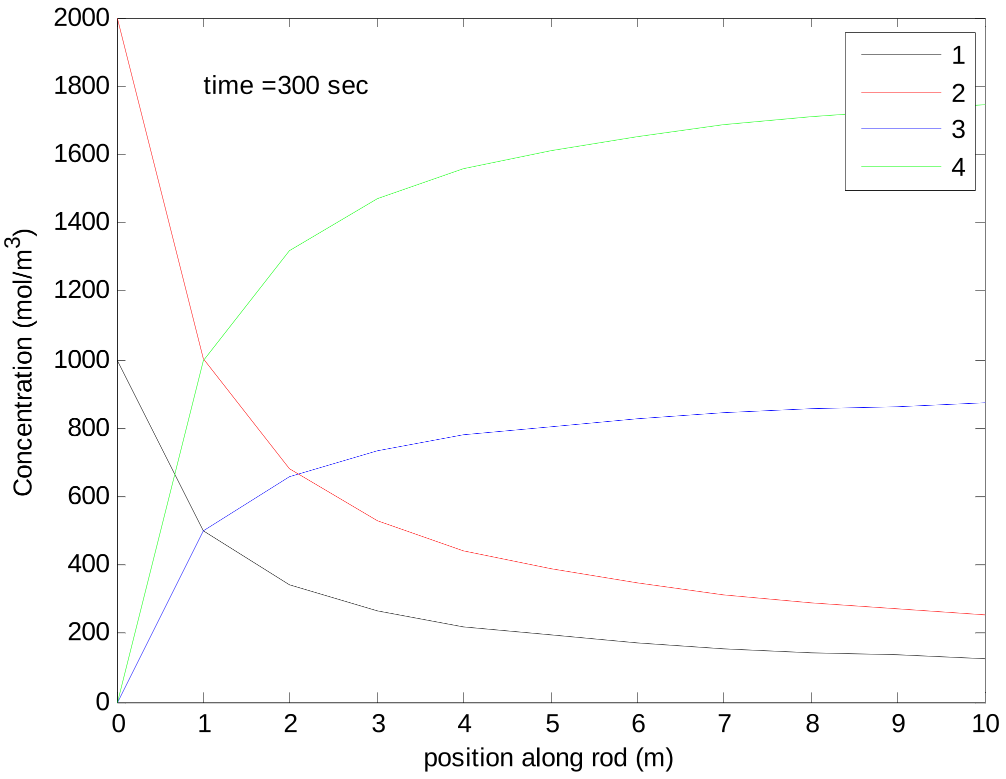
```
%
%  function defining initial condition
%

function ic_out = icfunk(x,keq);
ic(1) = 0.0;
ic(2) = 0.0;
ic(3) = 0.0;
ic(4) = 0.0;
ic_out = ic(keq);

%
%  functions defining LHS boundary condition
%

function fout = aBCo(t,k);
f(1) = 1;
f(2) = 1;
f(3) = 1;
f(4) = 1;
fout = f(k);

function fout = bBCo(t,k);
f(1) = 0;
f(2) = 0;
f(3) = 0;
f(4) = 0;
fout = f(k);

function fout = cBCo(t,k);
f(1) = -1000;
f(2) = -2000;
f(3) = 0;
f(4) = 0;
fout = f(k);

%
%  functions defining RHS boundary condition
%

function fout = aBCf(t,k);
f(1) = 0;
f(2) = 0;
f(3) = 0;
f(4) = 0;
fout = f(k);

function fout = bBCf(t,k);
f(1) = 1;
f(2) = 1;
f(3) = 1;
f(4) = 1;
fout = f(k);
```

```
function fout = cBCf(t,k);
f(1) = 0;
f(2) = 0;
f(3) = 0;
f(4) = 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 0.1 m/s, then the "residence time of the reactor" is given by $\tau = \frac{L}{v}$, (100 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.

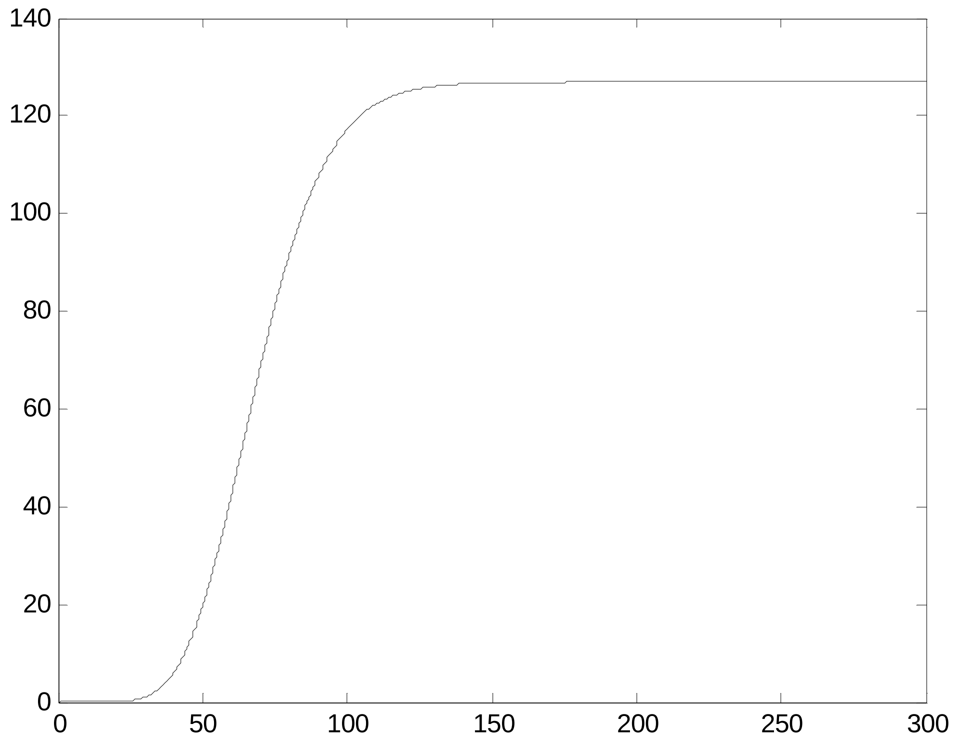
```
>> xvec(12)
ans =    10
```

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

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

  Tmat      13x3001x4            1248416  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 200 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 fraction of the reactants are used, i.e. what is the fractional yield?

The reactant A is input at 1000 mol/m^3 and emerges at

```
>> Tmat(12,3001,1)
ans =  126.9144
```

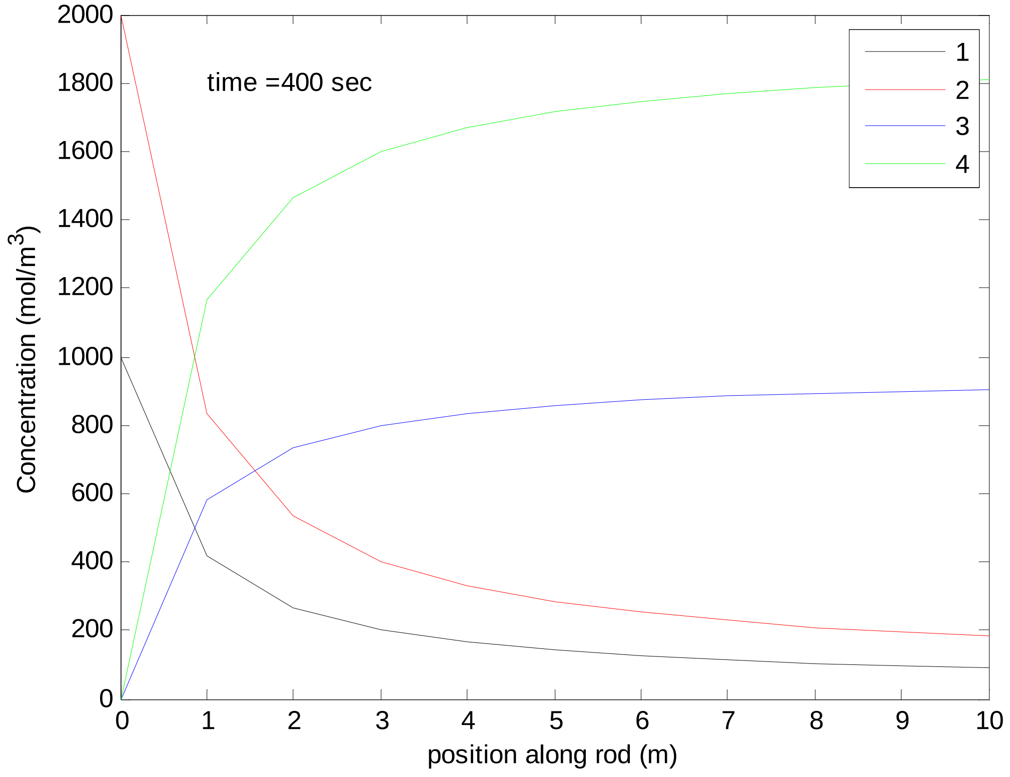
So the fractional yield is

$$Y_A = \frac{C_{A,in} - C_{A,out}}{C_{A,in}} = \frac{1000 - 126.9}{1000} = 87.3\%$$

(d) What can be done to the velocity to increase the fractional yield? How does this impact the amount of product made per hour, i.e. the through-put?

Slowing down the 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 =   91.4205
```

So the fractional yield is

$$Y_A = \frac{C_{A,in} - C_{A,out}}{C_{A,in}} = \frac{1000 - 91.4}{1000} = 90.9\%$$

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

$$Q_C = vA_X C_{C,out} = v\left(\pi r^2\right)C_{C,out}$$

For the case with the higher velocity, we have

```
>> Tmat(12,3001,3)
ans =  873.0784
```

$$Q_A = vA_X C_{A,out} = v\left(\pi r^2\right)C_{A,out} = 0.1\left(\pi r^2\right)873.1 = \left(\pi r^2\right)87.31$$

For the case with the lower velocity, we have

```
>> Tmat(12,4001,3)
ans =  903.5768
```

$$Q_A = vA_X C_{A,out} = v\left(\pi r^2\right)C_{A,out} = 0.05\left(\pi r^2\right)903.6 = \left(\pi r^2\right)45.18$$

The ratio is 1.93. So the higher velocity has a lower fractional yield but a higher through-put. This intuitively should make sense.

Problem (3) Hyperbolic PDE
Consider the wave equation

$$\frac{\partial^2 U}{\partial \mathrm{t}^2} = c^2 \frac{\partial^2 U}{\partial \mathrm{x}^2}$$

Consider a spatially one-dimensional problem where *x* ranges from 0 to 1. The hyperbolic problem generally requires two initial conditions (one for each order of the time derivative). Sample initial conditions are given below.

$$U(x, t=0) = \sin(2\pi x)$$
$$\frac{dU}{dt}(x, t=0) = 0.0$$

In the case of a string with each end fixed, the boundary conditions have the form:

$$U(x=0, t) = 0.0$$
$$U(x=1, t) = 0.0$$

(a) Show the profile of the wave at *t* = 5.0 for *c* = 1.5.
(b) What is the value of the wave at *x* = 0.75 and *t* = 5.0?

Solution
First, let's transform this hyperbolic PDE to a system of 2 parabolic PDEs.

Let $y^{(1)} = U$ and $y^{(2)} = \frac{\partial U}{\partial t}$. We resort to using subscripts in parentheses because subscripts will later denote position and superscripts time. So our two parabolic PDEs are

$$\frac{\partial y^{(1)}}{\partial \mathrm{t}} = y^{(2)} \qquad \frac{\partial y^{(2)}}{\partial \mathrm{t}} = c^2 \frac{\partial^2 y^{(1)}}{\partial \mathrm{x}^2}$$

with the initial conditions

$$y^{(1)}(x, t=0) = \sin(2\pi x) \qquad y^{(2)}(x, t=0) = 0.0$$

and the boundary conditions:

$$y^{(1)}(x=0, t) = 0.0 \qquad y^{(1)}(x=L, t) = 0.0$$

$$y^{(2)}(x=0, t) = 0.0 \qquad y^{(2)}(x=L, t) = 0.0$$

This is a single non-linear parabolic PDE with one spatial dimension and a Dirichlet boundary condition at *z*=0 and a Neumann boundary condition at *z*=0.1. 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 5.0 seconds and chose dt to be 0.001 seconds, so I had 5000 temporal intervals.

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

The string spans from 0 to 1.0 meter. I set dx to be 0.025 m, so I had 40 spatial intervals.

```
% discretize space
xo = 0;
xf = 1.0;
dx = 2.5e-2;
```

I defined the PDE in the following function.

```
function dydt_out = pdefunk(x,t,y,dydx,d2ydx2,keq);
c = 1.5;
dydt(1) = y(2);
dydt(2) = c^2*d2ydx2(1);
dydt_out = dydt(keq);
```

I defined the IC and BCs in the functions below.

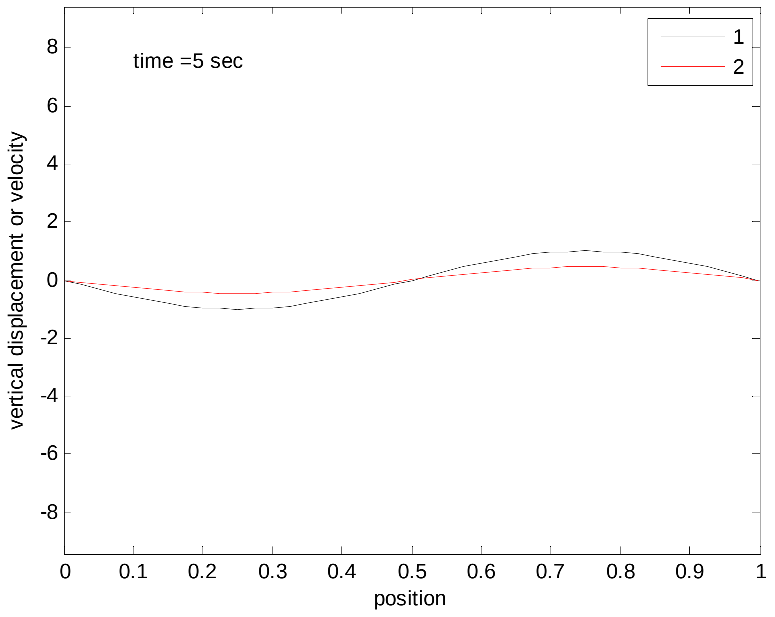
```
%
%  function defining initial condition
%
function ic_out = icfunk(x,keq);
ic(1) = sin(x*2.0*pi);
ic(2) = 0.0;
ic_out = ic(keq);

%
%  functions defining LHS boundary condition
%  for hyperbolic BC(2) must be time derivative of BC(1)
%
function fout = aBCo(t,k);
f(1) = 1;
f(2) = 1;
fout = f(k);
```

```
function fout = bBCo(t,k);
f(1) = 0;
f(2) = 0;
fout = f(k);

function fout = cBCo(t,k);
f(1) = 0;
f(2) = 0;
fout = f(k);

%
%  functions defining RHS boundary condition
%  for hyperbolic BC(2) must be time derivative of BC(1)
%

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

function fout = bBCf(t,k);
f(1) = 0;
f(2) = 0;
fout = f(k);

function fout = cBCf(t,k);
f(1) = 0;
f(2) = 0;
fout = f(k);
```

At the command line prompt, I typed

```
>> [xvec,tvec,Tmat] = hyperpde_n_anyBC;
```

This command generated a movie. The final frame of the movie is shown below.



(b) What is the value of the wave at *x* = 0.75 and *t* = 5.0?

I first made sure I understood the dimensions of the solution matrix and the location of 0.75 in the x vector.

```
>> whos Tmat
  Name        Size                     Bytes  Class     Attributes
  Tmat       43x5001x2               3440688  double
```

```
>> xvec(32)
ans =    0.7500
```

The value of the wave at *x* = 0.75 and *t* = 5.0 is stored here:

```
>> Tmat(32,5001,1)
ans =    0.9989
```

Therefore the displacement of the wave at *x* = 0.75 and *t* = 5.0 s is 0.9989.

Problem (4) Elliptic PDE

Consider the two-dimensional Laplace equation:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

on the unit square subject to the following boundary conditions,

$$\begin{aligned}
&T(x=0, y) = 75y \\
&T(x=1, y) = 50 + 50y \\
&T(x, y=0) = 50x \\
&T(x, y=1) = 75 + 25x + 50 * \sin(2\pi x)
\end{aligned}$$

(a) Show the steady state temperature distribution in the plate.
(b) What is the value of the wave at $x = 0.5$ and $y = 0.5$?

Solution:

I used Liebmann's method to solve this problem.

I set the maximum iterations to 1000 and I discretized the unit square with spatial intervals of 0.05 in both the x and y dimensions. I set $\lambda = 1.5$.

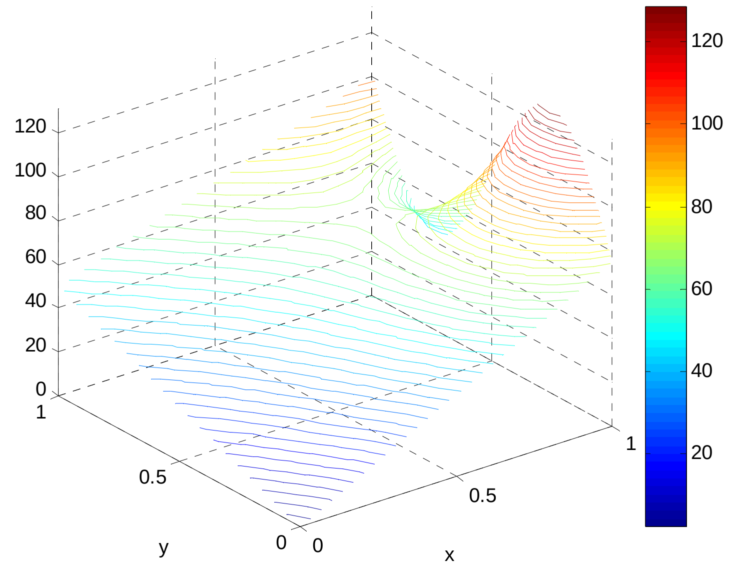
I input the following boundary conditions

```
%  Fill in Uold with four dirichlet BCs
for j = 1:1:ny
% BC for x = xo
    Uold(1,j) = 75.0*ygrid(j);
% BC for x = xf
    Uold(nx,j) = 50.0 + 50.0*ygrid(j);
% BC for y = yo
    Uold(j,1) = 50.0*xgrid(j);
% BC for y = yf
    Uold(j,ny) = 75.0 + 25*xgrid(j) + 50.0*sin(2.0*pi*xgrid(j));
end
```

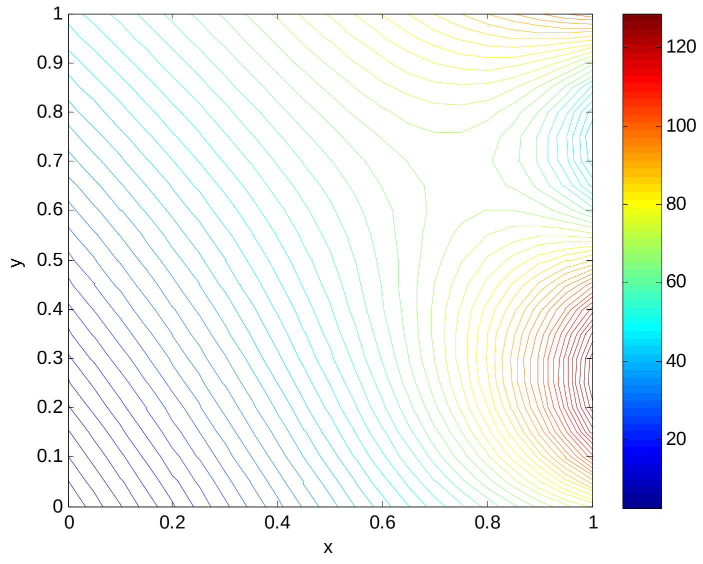
At the command prompt, I typed

```
>> U = ell_liebmann;
 lamdba = 1.500000 iteration = 198 , error = 0.000001
```

Therefore the code converged to a relative RMS error of less than $1.0\text{x}10^{-6}$ in 198 iterations. The plot is shown below.



or, changing the variable, `plot_dimensions` to 1 yields



(b) What is the temperature at $x = 0.5$ and $y = 0.5$?

```
>> U(11,11)
ans =   56.2500
```

So the temperature in the center of the plate is 56.25 K.

Problem (5) Numerical Integration
Consider the normal distribution

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

This function does not have an analytical integral.

For the standard normal distribution, where the mean is zero and the standard deviation is one, evaluate the integral from x = -2.0 to 1.0, i.e. $p(-2.0 \le x \le 1.0)$, using

(a) the trapezoidal method with 1 interval.
(b) the trapezoidal method with 10 intervals.
(c) the trapezoidal method with 100 intervals.
(d) the trapezoidal method with 1000 intervals.
(e) the Simpson's Second Order method with 100 intervals.
(f) the Simpson's Second Order method with 1000 intervals.
(g) the Simpson's Third Order method with 99 intervals.
(h) the Simpson's Fourth Order method with 100 intervals.
(i) Gaussian quadrature of sixth order.
(j) the cdf command in MatLab.
(k) Comment on the effect of number of intervals and order of the method.

Solution:

In each code, whether it be trapezoidal.m, simpson2.m, simpson3.m, simpson4.m or gaussquad.m, the integrand is input as

```
function f = funkeval(x)
f = 1/sqrt(2.0*pi)*exp(-0.5*x^2);
```

(a) the trapezoidal method with 1 interval.

```
>>   integral = trapezoidal(-2.0,1.0,1)
integral =   0.443942536548497
```

(b) the trapezoidal method with 10 intervals.

```
>> integral = trapezoidal(-2.0,1.0,10)
integral =   0.815965728062905
```

(c) the trapezoidal method with 100 intervals.

```
>> integral = trapezoidal(-2.0,1.0,100)
integral =   0.818568367248082
```

(d) the trapezoidal method with 1000 intervals.

```
>> integral = trapezoidal(-2.0,1.0,1000)
integral =   0.818594351655828
```

(e) the Simpson’s Second Order method with 100 intervals.

```
>> integral = simpson2(-2.0,1.0,100)
integral =   0.818594615812413
```

(f) the Simpson’s Second Order method with 1000 intervals.

```
>> integral = simpson2(-2.0,1.0,1000)
integral =   0.818594614120533
```

(g) the Simpson’s Third Order method with 99 intervals.

```
>> integral = simpson3(-2.0,1.0,99)
integral =   0.818594618084208
```

(h) the Simpson’s Fourth Order method with 100 intervals.

```
>> integral = simpson4(-2.0,1.0,100)
integral =   0.818594614119623
```

(i) Gaussian quadrature of sixth order.

```
>> integral = gaussquad(-2.0,1.0,6)
integral =   0.818594704229380
```

(j) the cdf command in MatLab.

```
>> plow = cdf('normal',-2.0,0,1)
plow =   0.022750131948179

>> phigh = cdf('normal',1.0,0,1)
phigh =   0.841344746068543

>> p = phigh - plow
p =   0.818594614120364
```

(k) Comment on the effect of number of intervals and order of the method.

As the number of intervals increases, the accuracy of the method increases. As the order of the method increases, the number of intervals needed to generate a given level of accuracy drastically decreases.

Problem (6) Integral Equations

Classify and numerically solve the following integral equation.

$$\phi(x) = \frac{x^2}{100} + \frac{5}{2}\int_5^x e^{-\frac{(x+y)}{10}} \left[\sin(\phi(y))\right]^2 dy$$

Solve for x = 5 to 10.
Classify as linear/nonlinear, Volterra/Fredholm, first/second kind.
Provide a plot of the solution.
Demonstrate (1) the effect of changing the increment size (use say 5 and 20 intervals).
Demonstrate (2) the convergence of the method by looking at the solution at each iteration.

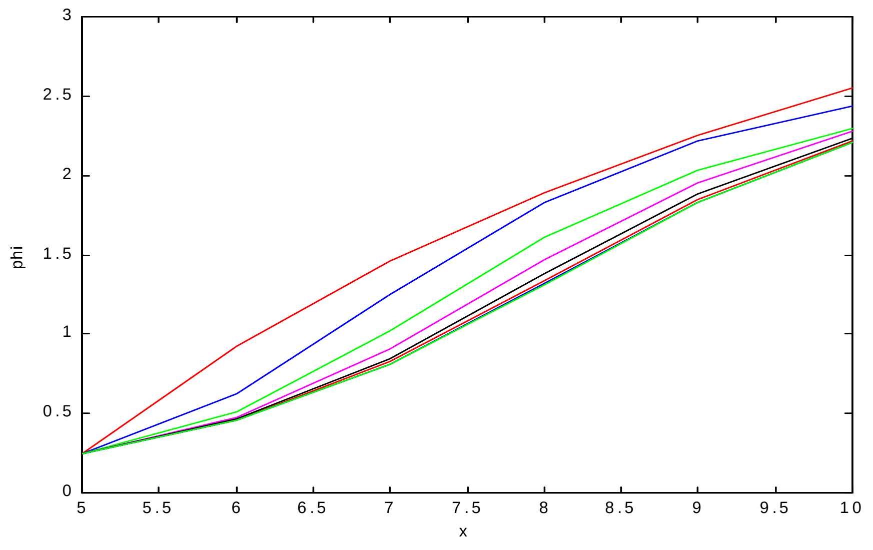
Solution:

This equation is a nonlinear Volterra equation of the second kind.
It is nonlinear in phi. It is Volterra because phi appears both inside and outside the integral.
It is of the second kind because the limits of integration are variable.

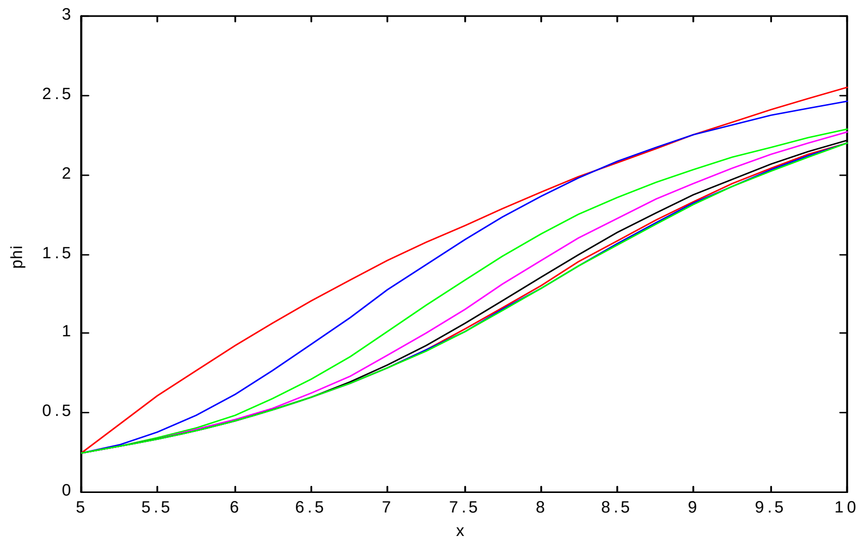
Color code for following two plots:

red solid	- solution after 1 iteration
blue solid	- solution after 2 iterations
green solid	- solution after 3 iterations
magenta solid	- solution after 4 iterations
black dotted	- solution after 5 iterations
red dotted	- solution after 6 iterations
blue dotted	- solution after 7 iterations
green dotted	- solution after 8 iterations

Plot of solution using 5 intervals.



Plot of solution using 20 intervals.



We can see that the code is converging with each iteration. We can also see that we obtain a smoother solution with more intervals.