MATH5004 LAB 2

Part I: Solution of BVP using MATLAB built-in functions

The MATLAB PDE solver, **pdepe**(), solves initial-boundary value problems for systems of parabolic and elliptic PDEs in one space variable and time. There must be at least one parabolic equation in the system.

$$c(x,t,u,u'(x))\frac{\partial u}{\partial t} = x^{-m}\frac{\partial}{\partial x}\left(x^m f(x,t,u,u'(x))\right) + s(x,t,u,u'(x)) \qquad (*)$$

with initial and boundary conditions in the following forms:

$$\begin{aligned} &u(x,t_0) = u_0(x) \\ &p(x,t,u) + q(x,t)f(x,t,u,u'(x)) = 0 \end{aligned} \qquad (**)$$

The PDEs (*) hold for $t_0 \le t \le t_f$ and $a \le x \le b$. The interval [a,b] must be finite. m can be 0, 1, or 2, corresponding to slab, cylindrical, or spherical symmetry, respectively. If m>0, then $a \ge 0$ must also hold. The basic syntax of the solver is

```
sol = pdepe(m,@pdefun,@icfun,@bcfun,xmesh,tspan)
```

where

m : specifies the symmetry of the problem; m can be 0 = slab, 1 = cylindrical, or 2 = spherical.

pdefun : Function that defines the components of the PDEs. It computes the terms c, f and s in Equation (*), and has the form

```
[c,f,s] = pdefun(x,t,u,dudx);
```

where x and t are scalars; and u and dudx are vectors that approximate the solution u and its partial derivative with respect to x; c, f, and s are column vectors; c stores the diagonal elements of the matrix c.

icfun : function that evaluates the initial conditions. It has the form

```
u = icfun(x);
```

When called with an argument x, **icfun**() evaluates and returns the initial values of the solution components at x in the column vector u.

bcfun : Function that evaluates the terms and the boundary conditions. It has the form

```
[pl,ql,pr,qr] = bcfun(xl,ul,xr,ur,t);
```

where ul is the approximate solution at the left boundary xl=a and ur is the approximate solution at the right boundary xr=b; pl and ql are column vectors corresponding to p and the diagonal of q evaluated at xl. Similarly, pr and qr correspond to xr.

Example 1. Solve

$$\frac{\partial u}{\partial t} = \frac{1}{10}\frac{\partial^2 u}{\partial x^2}, \quad (x,t) \in (0,1) \times (0,\tau)$$

$$u(x,0) = 1,$$

$$u(0,t) = 0, \qquad \frac{\partial u}{\partial t}(1,t) = 0.$$

Firstly, rewrite the PDE in the form of (*) and the boundary conditions in the form of (**).

$$\frac{\partial u}{\partial t} = x^0 \frac{\partial}{\partial x}\left(x^0 \frac{1}{10}\frac{\partial u}{\partial x} \right) + 0$$

with parameter $m = 0$ and the terms

$$c(x,t,u,u'(x)) = 1$$

$$f(x,t,u,u'(x)) = \frac{1}{10}\frac{\partial u}{\partial x}$$

$$s(x,t,u,u'(x)) = 0$$

and

$$u(0,t) + 0 \cdot \left(\frac{1}{10}\frac{\partial u}{\partial x}(0,t) \right) = 0 \quad \text{at} \quad x = 0$$

$$0 \quad + 10 \cdot \left(\frac{1}{10}\frac{\partial u}{\partial x}(1,t) \right) = 0 \quad \text{at} \quad x = 1$$

with the terms

$$pl = 1, \;\; ql = 0, \;\; pr = 0, \;\; qr = 10.$$

Then, we create three M-files for the PDE function “pdex1pde()”, the initial condition function “pdex1ic()”and the boundary condition function “pdex1bc()” as follows.

```
function [c, f, s] = pdex1pde(x,t,u,DuDx)
c = 1;
f = (1/10)*DuDx;
s = 0;
```

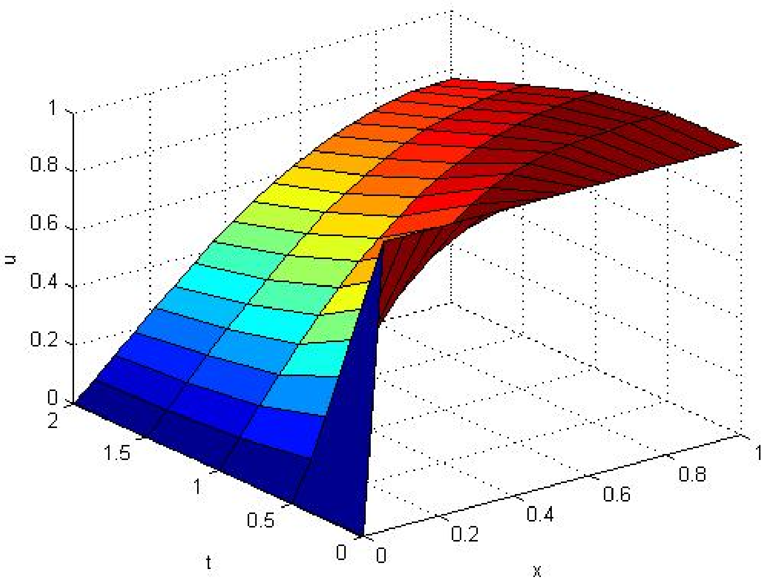
```
function  u0 = pdex1ic(x)
u0 = 1;
```

```
function [pl,ql,pr,qr]=pdex1bc(xl,ul,xr,ur,t)
pl =ul;
ql = 0;
pr = 0;
qr = 10;
```

Next we use the function “**pdepe**()” to find the solution on the mesh produced by 20 equally spaced points from the spatial interval [0,1] and five values t from the time interval [0, 2] and plot the solution.

```
x=linspace(0,1,20);
t=linspace(0,2,5);
m=0;
sol=pdepe(m,@pdex1pde,@pdex1ic,@pdex1bc,x,t);
u = sol( :, :, 1);
surf(x,t,u);
```

yields



Notes: The output argument sol is a three-dimensional array, such that:

sol(:,:,k) approximates component k of the solution .

sol(i,:,k) approximates component k of the solution at time tspan(i) and mesh points xmesh(:).

sol(i,j,k) approximates component k of the solution at time tspan(i) and the mesh point xmesh(j).

Part II: FDM for 1D BVP

Example 2. Consider the problem of determining the subsurface temperature fluctuations of rock as the result of daily or seasonal temperature variations. As the surface is heated or cooled, the heat will diffuse through the soil and rock. Mathematically, diffusion may be represented by the equation

$$\frac{\partial T(t,z)}{\partial t} = K\frac{\partial^2 T(t,z)}{\partial z^2}$$

where $T(t,z)$ is the temperature at time t and depth z, and K is a constant which measures the "diffusivity" of the rock, i.e. how quickly heat moves through the rock. A typical value for K is 10^-6 m^2/s.

In order to solve the system numerically, we must first choose an appropriate "space" to solve the problem in. We will have two axes, the vertical axes representing depth, from 0m to 100 m below the surface, and the horizontal axis will represent one year's worth of time.

Boundary Conditions

The heat equation is an example of what is known as a "partial differential equation." A differential equation is any equation in which a function (Temperature in time and space in this instance) is not represented directly, but via it's derivative. Partial indicates that there are at least two variables (time and space) in the derivatives. An ordinary DE only has one derivative.

In order to solve a PDE numerically, we need to specify boundary conditions. The four boundaries of our "space" are the surface (0m), the bottom depth (100m), an arbitrary beginning time, and one year later.

- Top Boundary:

The top boundary is the simplest, we will simply specify a temperature that varies seasonally, i.e.

$$T(t,0) = 15 - 10\sin(\frac{2\pi t}{12})$$

This will create an average annual temperature of 15, an annual low of 5 and an annual high of 25 (time is measured in months, and we are not starting on Jan 1).

- Bottom Boundary:

*Bottom boundary

The bottom boundary condition is easy as well... we will assume that no heat reaches the bottom from the top, i.e.

$$\frac{\partial T(t,z)}{\partial z} = 0.$$

This specifies that there is 0 heat flow at the bottom. We do not yet know what this bottom temperature will be.

- Left and right boundaries

The left and right boundary conditions are a bit more problematic, as they require that we already know the temperature at every depth, which is exactly what we are trying to find out! But we do know that the temperature now and exactly one year from now should be the same.

Finite difference approximations

The final step is to determine how to use finite differences to evaluate the derivatives. For the first derivative of temperature with respect to time we will use the "forward derivative"

$$\frac{\partial T(t_i, z_j)}{\partial t} \approx \frac{T(t_{i+1}, z_j) - T(t_i, z_j)}{\Delta t}.$$

For the first derivative of temperature with respect to depth, we will use the "central derivative"

$$\frac{\partial T(t_i, z_j)}{\partial z} \approx \frac{T(t_i, z_{j+1}) - T(t_i, z_{j-1})}{2\Delta z}.$$

For the second derivative of temperature with respect to depth, we will use

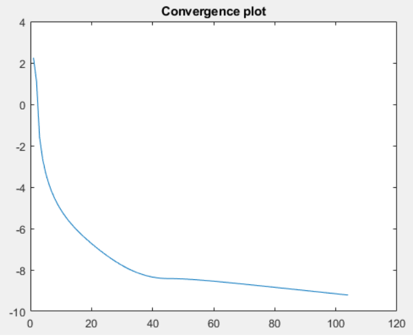
$$\frac{\partial^2 T(t_i, z_j)}{\partial z^2} \approx \frac{\frac{T(t_i, z_{j+1}) - T(t_i, z_j)}{\Delta z} - \frac{T(t_i, z_j) - T(t_i, z_{j-1})}{\Delta z}}{\Delta z}.$$

The following MATLAB program implements this procedure

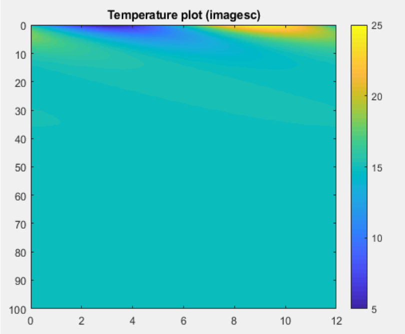
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Script to solve the heat equation in soil with seasonal temperature forcing
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Initialize variables
dz = .25; %each depth step is 1/4 meter
Nz = 400; % Choose the number of depth steps (should go to at least 100 m)
Nt = 5000; % Choose the number of time steps
dt = (365*24*60*60)/Nt; %Length of each time step in seconds (~ 6.3*10^3
seconds, or ~105 minutes)
K =  2*10^-6; % "canonical" K is 10e-6 m^2/s
T = 15*ones(Nz+1,Nt+1);  %Create a matrix with Nz+1 rows, and Nt+1 columns
                         % Initial guess is that T is 15 everywhere.
time = [0:12/Nt:12];
T(1,:)  = 15-10*sin(2*pi*time/12);  %Set surface temperature
maxiter = 500
for iter = 1:maxiter
    Tlast = T; %Save the last guess
    T(:,1) = Tlast(:,end); %Initialize the temp at t=0 to the last temp
    for i=2:Nt+1,
        depth_2D = (T(1:end-2,i-1)-2*T(2:end-1,i-1)+T(3:end,i-1))/dz^2;
        time_1D = K*depth_2D;
        T(2:end-1,i) = time_1D*dt + T(2:end-1,i-1);
        T(end,i) = T(end-1,i); % Enforce bottom BC
    end
    err(iter) = max(abs(T(:)-Tlast(:))); %Find difference btw last 2 sols
    if err(iter)<1E-4
        break; % Stop if solutions very similar, we have convergence
    end
end
if iter==maxiter;
    warning('Convergence not reached')
end
figure(1)
plot(log(err)), title('Convergence plot')
figure(2)
imagesc([0 12],[0 100],T); title('Temperature plot (imagesc)')
colorbar
figure(3)
depth = [0:dz:Nz*dz];
contourf(time,-depth,T); title('Temperature plot (contourf)')
colorbar
figure(4)
plot(time,T(1,:),time,T(21,:),time,T(41,:),time,T(61,:),time,T(81,:))
xlabel('Time (months)'); ylabel('Temperature (C)');
legend('0m','5m','10m','15m', '20m')
```

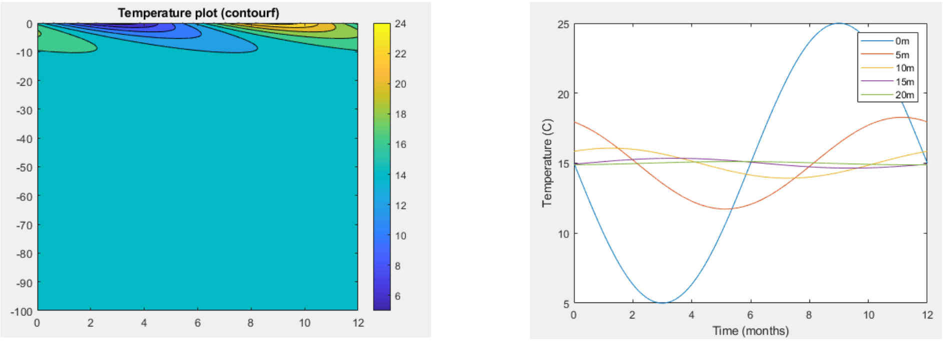
Results:



(a)



(b)



(c) (d)

Figure 1.

Example 3. Consider a Two-point BVP:

$$\boldsymbol{u_{xx} = f(x), \qquad x \in (0, 1)}$$

$$\boldsymbol{u(0) = 1, \qquad u(1) = -1,}$$

where $f(x) = -\pi^2 \cos(\pi x)$.

MATLAB code based on FDM

Main.m

```
function main()
clear;
close all
%input
a=0; b=1; n=40;
ua=1; ub=-1;
f= @(x) -pi^2*cos(pi*x);
%call twoPointBVP function
[x,U] = twoPointBVP(a,b,ua,ub,f,n);
%Plot and error analysis
plot(x, U,'o');
hold on

u=zeros(n-1,1);
for i=1:n-1
     u(i)=cos(pi*x(i));
end
plot(x,u)

%Plot error

figure(2)
plot(x,U-u)
norm(U-u,inf)    %print out the maximum error
```

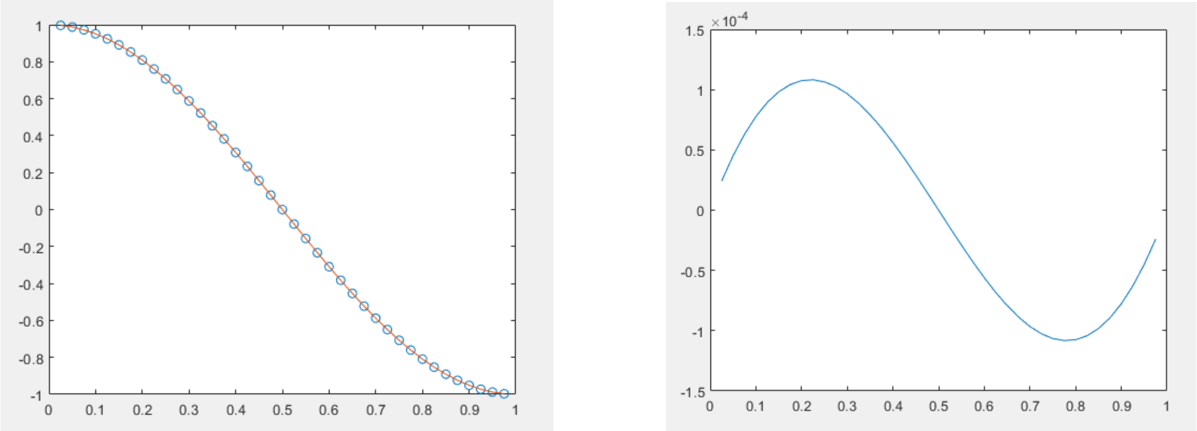
twoPointBVP.m

```
function [x, U] = twoPointBVP(a,b, ua, ub, f, n)
% Sove BVP: u"(x)=f(x) using centreal difference scheme
%input:    a, b are two end points
%          ua, ub : dirichlet bc at a and b
%          f : extenal force
%          n : number of grid points
%output    x: x(1),...,x(n-1) are drid points
%          U: U(1),...,U(n-1) are approx. sol
%
h= (b-a)/n;
h1=h*h;
A = sparse(n-1,n-1);
F = zeros(n-1,1);

for i=1:n-2
    A(i,i) = -2/h1;
    A(i+1,i) = 1/h1;
    A(i,i+1)=1/h1;
end
A(n-1,n-1) = -2/h1;
for i=1:n-1
    x(i) = a+i*h;
    F(i) = f(x(i));
end
F(1) = F(1)-ua/h1;
F(n-1)=F(n-1)-ub/h1;

U = A\F;
```

Result



(a) (b)

Figure 2. (a) The plot of the computed solution (a line with little ‘o’s), and the exact solution (solid line); (b) The plot of the errors in the infinity norm

Assignment I

Question 1. (LAB-WK2)

Implement the above MATLAB code in Example 3 to solve the following steady state heat conduction problem

$$k(x)\frac{\partial^2 u}{\partial x^2} = f(x), \quad 0 < x < 1,$$

$$u(0) = 1, \qquad \frac{\partial}{\partial x}u(1) = 0,$$

where $f(x) = -\cos(\pi x),\ k = \pi x^2$.

Note: Assignments I & II (50%): Assignment Questions will be given weekly.

In this week, Questions 1 (LAB-WK2) is a part of Assignment I, please submit a document file with MATLAB code via Blackboard by the due date of Assignment I on Friday 11 September 2020