HOMEWORK 5 - FRANCISCO CASTILLO

Contents

Defined functions

```
function d=GaussTriSol(a,b,c,d)
    N=length(a);
    for i=2:N
        b(i)=b(i)-c(i-1)*a(i)/b(i-1);
        d(i)=d(i)-d(i-1)*a(i)/b(i-1);
    end
    d(N)=d(N)/b(N);
    for i=N-1:-1:1
        d(i)=(d(i)-c(i)*d(i+1))/b(i);
    end
end
```

Problem 1

In this problem we implement the code for the node based mesh fromulated previously. I start by initializing the variables and ploting the initial condition, an horizontal profile $T = 2$.

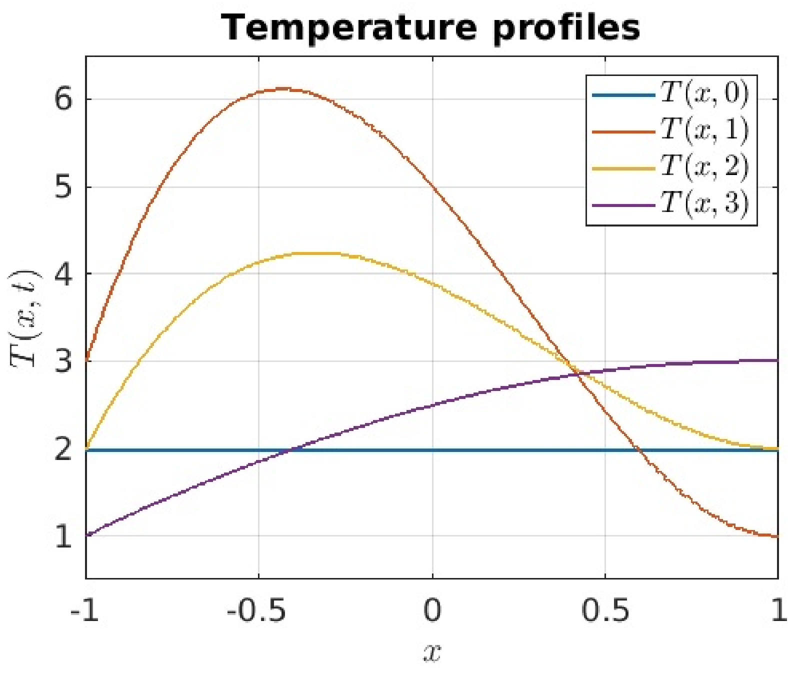
```
format long
clear all; close all; clc
labelfontsize=14;
linewidth=1.5;
axisfontsize=15;

M=256;
L=2;
alpha = 0.1;
h=L/M;
dt=4*h^2/(2*alpha);
x=linspace(-1,1,M+1)';
g = @(t) 2-sin(3*pi*t/2);
q = @(x,t) 3*pi*sin(pi*sqrt(t)/2).*cos(pi*sqrt(t)/2).*(x.^3-x.^2-x+1)./(2*sqrt(t))...
    +3*pi*cos(3*pi*t/2).*sin(pi*x/2)/2+...
    alpha*(sin(pi*sqrt(t)/2).^2.*(6-18*x)+pi^2*sin(3*pi*t/2).*sin(pi*x/2)/4);

T=2*ones(M+1,1);
qnew=zeros(M+1,1);
time=0;
outputTime=[1 2 3];
endtime=outputTime(end);
j=1;
```

At the beginning of the loop I check that the next time step is not going to be after the output time desired. I adjust the time step if that was the case and for the next iteration I reset the time step to the specified by the problem

$$\Delta t = 2\frac{h^2}{\alpha}$$

Since the value of dt might be different, so it might the value of B and therefore the values of the tridiagonal vectors are recalculated before the tridiagonal solver function. In the case of the right hand side vector, it needs to be recalculated since its value depends on the Temperature values and the values of q.

```
figure(1)
plot(x,T,'linewidth',linewidth)
hold on
axis([-1 1 0.5 6.5])
grid on
xlabel('$x$','fontsize',labelfontsize,'interpreter','latex')
ylabel('$T(x,t)$','fontsize',labelfontsize,'interpreter','latex')
set(gca,'fontsize',axisfontsize)
title('Temperature profiles')

while time < endtime
    if (time < outputTime(j) && time+dt >= outputTime(j))
        dt=outputTime(j)-time;
    else
        dt=4*h^2/(2*alpha);
    end
    qold=qnew;
    qnew=q(x,time+dt);

    B=alpha*dt/(2*h^2);
    a=-B*ones(M-1,1);
    b=(1+2*B)*ones(M-1,1);
    b(end)=1+B;
    c=-B*ones(M-1,1);
    d=zeros(M-1,1);

    for i=2:M-1 %interior of the interior for d
        d(i)=T(i+1)+B*(T(i+2)-2*T(i+1)+T(i))+...
           dt*0.5*(qold(i+1)+qnew(i+1));
    end
    d(1)=T(2)+B*(T(3)-2*T(2)+g(time)+g(time+dt))+...
        +dt*0.5*(qold(2)+qnew(2));
    d(end)=T(M)+B*(-T(M)+T(M-1))+...
        +dt*0.5*(qold(M)+qnew(M));
    %
    T(2:end-1)=GaussTriSol(a,b,c,d);
    T(1)=g(time+dt);
    T(end)=T(end-1);
    if time+dt == outputTime(j)
        j=j+1;
        figure(1)
        plot(x,T,'linewidth',linewidth)
        axis([-1 1 0.5 6.5])
    end
    time=time+dt;
end
lh = legend({'$T(x,0)$','$T(x,1)$','$T(x,2)$','$T(x,3)$'},...
    'Interpreter','latex','Fontsize',14);
```



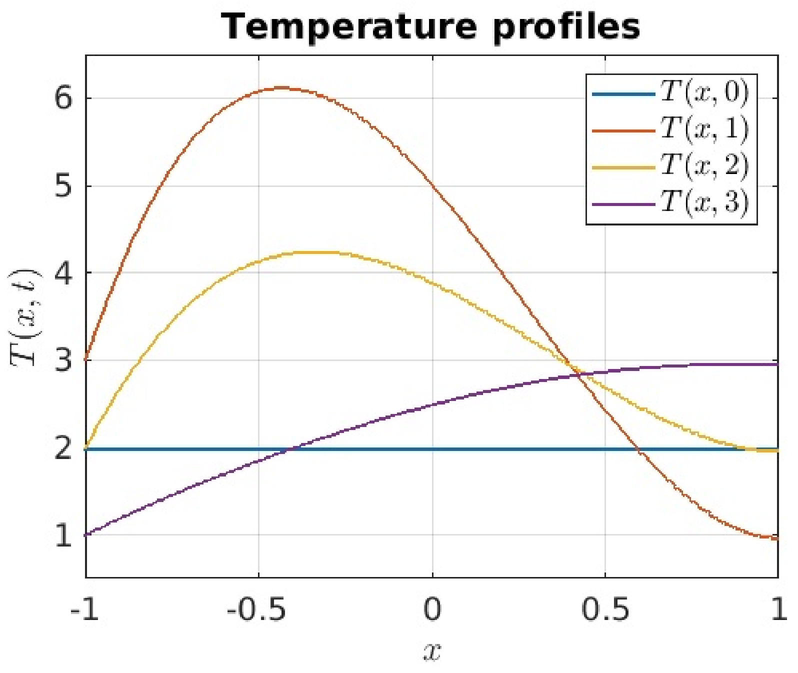
In the previous image we can see the temperature profiles with time. The temperature profiles have obviously oscillatory behaviour. This is easier seen by plotting all the timesteps into a plot and making a movie. I simulated up to 40 seconds and never got to a stabilization.

Problem 2

In this problem we repeat the same proccess but using cell based mesh. The code follows the same logic, only the equations for a, b, c, d and the boundary conditions change.

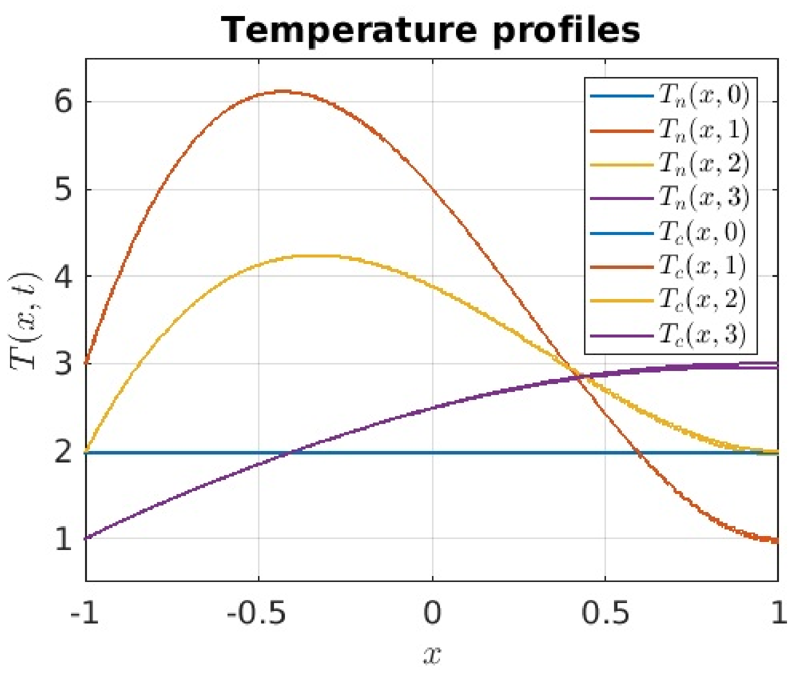
```
M=256;
L=2;
alpha = 0.1;
h=L/M;
dt=4*h^2/(2*alpha);
x=[-1-h/2:h:1+h/2]';
g = @(t) 2-sin(3*pi*t/2);
q = @(x,t) 3*pi*sin(pi*sqrt(t)/2).*cos(pi*sqrt(t)/2).*(x.^3-x.^2-x+1)./(2*sqrt(t))...
    +3*pi*cos(3*pi*t/2).*sin(pi*x/2)/2+...
    alpha*(sin(pi*sqrt(t)/2).^2.*(6-18*x)+pi^2*sin(3*pi*t/2).*sin(pi*x/2)/4);

T=2*ones(M+2,1);
qnew=zeros(M+2,1);
time=0;
outputTime=[1 2 3];
endtime=outputTime(end);
```

```
j=1;

figure(2)
plot(x,T,'linewidth',linewidth)
hold on
axis([-1 1 0.5 6.5])
grid on
xlabel('$x$','fontsize',labelfontsize,'interpreter','latex')
ylabel('$T(x,t)$','fontsize',labelfontsize,'interpreter','latex')
set(gca,'fontsize',axisfontsize)
title('Temperature profiles')

while time < endtime
    if (time < outputTime(j) && time+dt >= outputTime(j))
        dt=outputTime(j)-time;
    else
        dt=4*h^2/(2*alpha);
    end
    qold=qnew;
    qnew=q(x,time+dt);

    B=alpha*dt/(2*h^2);
    a=-B*ones(M,1);
    b=(1+2*B)*ones(M,1);
    b(1)=1+3*B;
    b(end)=1+B;
    c=-B*ones(M,1);
    c(end)=0;
    d=zeros(M,1);

    for i=2:M-1 %interior of the interior for d
        d(i)=T(i+1)+B*(T(i+2)-2*T(i+1)+T(i))+...
           dt*0.5*(qold(i+1)+qnew(i+1));
    end
    d(1)=T(2)+B*(T(3)-3*T(2)+2*g(time)+2*g(time+dt))+...
        +dt*0.5*(qold(2)+qnew(2));
    d(end)=T(M+1)+B*(-T(M+1)+T(M+1))+...
        +dt*0.5*(qold(M+1)+qnew(M+1));

    Tint(:)=GaussTriSol(a,b,c,d);
    T(2:end-1)=Tint(:);
    T(1)=2*g(time+dt)-T(2);
    T(end)=T(end-1);
    if time+dt == outputTime(j)
        j=j+1;
        figure(2)
        plot(x,T,'linewidth',linewidth)
        axis([-1 1 0.5 6.5])
    end
    time=time+dt;
end
lh = legend({'$T(x,0)$','$T(x,1)$','$T(x,2)$','$T(x,3)$'},...
    'Interpreter','latex','Fontsize',14);
```



In the previous image we can see the temperature profiles with time. The temperature profiles are clearly equal to the ones obtained by the node based method. However, I will superpose them to make sure.

```
L = findobj(1,'type','line');
copyobj(L,findobj(2,'type','axes'));
lh = legend({'$T_n(x,0)$','$T_n(x,1)$','$T_n(x,2)$','$T_n(x,3)$','$T_c(x,0)$','$T_c(x,1)$','$T_c(x,2)$','$T_c(x,3)$'},...
    'Interpreter','latex','Fontsize',13);
```



In the previous image the n subindex stands for node based and the c stands for cell based mesh. We can observe how the matching of the two types of meshing is good, as expected. It gets a bit worse close to the right boundary condition, the Neumann boundary condition, and for higher values of time.

Published with MATLAB® R2017a