Computer Based Engineering Mathematics

Project 1

Solving heat equation

Group Members:

Mostafa Mohamed	(3101359)
Ahmed Mousa	(3101343)
Ahmed Mohamed	(3101361)
Osama Elsafty	(3101070)
Abdelrahman Salam	(3100657)

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Prof. Dr.-Ing. habil. Jörg Schröder
Universität Duisburg-Essen
Fakultät für
Ingenieurwissenschaften Abteilung
Computer Engineering
(Software)

SoSe 2021
Lecturers : Dr. Claudia Weis

Problem description:

In simple terms, we have function $u(x,t)$, we need to calculate its approximate values at discrete points in grid with unit step in the x direction is h and the unit step in the t direction is k.

We have the following constraints:

$$0 \le x \le 1, 0 \le t \le T$$

$$u(x,0) = f(x), x \in (0;1)$$

$$u(0,t) = 0, t \in [0;T]$$

$$\frac{\partial u}{\partial t}(x,t) - c\frac{\partial^2 u}{\partial x^2}(x,t) = 0$$

We are asked to write a MATLAB code that calculates the requirements.

u(x; t) is the temperature at the point x in the rod at the time t, the real number c > 0 is a material constant, L is the length of the rod and f is the initial temperature (distribution); note that all constants and variables are written without dimension.
The goal of this project is to implement a general algorithm for numerically solving (1) and to visualize the solution for a specific

Approach:

To start, we need to discretize the continuous differential equation, to do that we need an approximation of its terms.

Finding an approximation for $\frac{\partial^2 u}{\partial x^2}$:

From Taylor series:

$$u(x+h,t) = u(x,t) + h\frac{\partial u}{\partial x}(x,t) + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}(x,t) + \frac{h^3}{6}\frac{\partial^3 u}{\partial x^3}(x,t) + \frac{h^4}{24}\frac{\partial^4 u}{\partial x^4}(x,t) + \cdots$$

Similarly:

$$u(x-h,t) = u(x,t) - h\frac{\partial u}{\partial x}(x,t) + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}(x,t) - \frac{h^3}{6}\frac{\partial^3 u}{\partial x^3}(x,t) + \frac{h^4}{24}\frac{\partial^4 u}{\partial x^4}(x,t) + \cdots$$

By addition we get:

$$u(x+h,t) + u(x-h,t) = 2u(x,t) + h^2\frac{\partial^2 u}{\partial x^2}(x,t) + O(h^4)$$

$$\frac{\partial^2 u}{\partial x^2}(x,t) = \frac{u(x+h,t) + u(x-h,t) - 2u(x,t)}{h^2} + O(h^2)$$

Finding an approximation for $\frac{\partial u}{\partial t}$:

From the Taylor series:

$$u(x,t+k) = u(x,t) + k\frac{\partial u}{\partial t}(x,t) + \frac{k^2}{2}\frac{\partial^2 u}{\partial t^2}(x,t) + \frac{k^3}{6}\frac{\partial^3 u}{\partial t^3}(x,t) + \cdots$$

We get:

$$k\frac{\partial u}{\partial t}(x,t) = u(x,t+k) - u(x,t) + O(k^2)$$

$$\frac{\partial u}{\partial t}(x,t) = \frac{u(x,t+k) - u(x,t)}{k} + O(k)$$

Note 1: We could theoretically get a better approximation by subtracting the Taylor Expansion of $u(x,t-k)$ from $u(x,t+k)$. This would give us $O(k^2)$ error, however, because we are not bounded on the $t-$ axis from the top, it's not easily applicable.

Note 2: $O(k)$ is not a good approximation for our purposes, we will show how to handle this later.

Substitution:

Let's ignore discretization errors for now, we get the following equation by substituting the derivatives in our equation:

$$\frac{u(x,t+k) - u(x,t)}{k} - \frac{u(x+h,t) + u(x-h,t) - 2u(x,t)}{h^2} = 0$$

Multiply by $k\,h^2$

$$h^2u(x,t+k) + (2k - h^2)u(x,t) - k\,u(x+h,t) - k\,u(x-h,t) = 0$$

For simplicity, and to match with our code, we will use the following notation:

Given $n = {}^{1}/_{h}, m = {}^{T}/_{k}$ and setting $x = i \cdot h, t = j \cdot k$ we conclude the following:

$i \in [0,n], j \in [0,m]$

$u(x,t) = u_{i,j}$

$u(x+h,t) = u_{i+1,j}, u(x,t+k) = u_{i,j+1}, u(x-h,t) = u_{i-1,j}, u(x,t-k) = u_{i,j-1},$

This gives us the formula in the new notation as:

$$h^2u_{i,j+1} + (2k - h^2)u_{i,j} - k\,u_{i+1,j} - k\,u_{i-1,j} = 0$$

We can rewrite our boundaries (constraints) as follows:

- $u_{0,j} = u_{n,j} = 0$ (left and right walls)
- $u_{i,0} = f(i \cdot h)$ (bottom)

We can choose whichever approach we seem fit, but it seems the simplest approach is the bottom-up approach. This is simpler than the linear algebra approach to solve all equations at once.

Since we are building bottom-up, we need to find the recursive formula for that, which is:

$$u_{i,j+1} = \frac{1}{h^2}\left(k\left(u_{i+1,j} + u_{i-1,j}\right) - (2k - h^2)u_{i,j}\right)$$

The right hand-side depends entirely on previous value of j, so we need to calculate all values of i for the current j before advancing, an outer loop over j and an inner loop over i will do that for us.

Thus, we need to build our base case of $i = 0$, by using $u_{i,0} = f(i \cdot h)$ and build the rest iteratively.

This is all what we need, except, running the code at this stage might or might not produce correct results… the reason is that our discretization error is relatively large $O(k) + O(h^2)$.

For general purposes $O(h^2)$ is not an issue, however $O(k)$ might produce oscillation issues. To reduce it, we will perform a simple trick: we will internally use a value of $k/1000$ and slice the results at once per 1000.

Task 1: Computing the grid and the approximate solution

```
%Number of unknowns, M points in (x) and N points in (y)
function [u] = heatsolution(fPassed,MPassed,NPassed,cPassed,L,time)
M = MPassed;
N = NPassed;
size = (N-1)*(M-2);
c = cPassed;
dt = time/(N-1);
dx = L/(M-1);
A = 2*c*dt;
B = -4*c*dt;
C = dx^2;
D = -dx^2;

%Matrix = zeros(size);
Matrix = sparse(size,size);

%Matrix of Coefficients
for i = 1:size

    %Coefficients of (i,j)
    Matrix(i,i) = B;

        if i <= N-1
            %Coefficients of (i+1,j)
            Matrix(i,i+(N-1)) = A;
        end
        %if i-(N-1) >= 1
        if i > N-1
            %Coefficients of (i-1,j)
            n = N-1;
            Matrix(i,i-n) = A;
        end
        if i > 1
            %Coefficients of (i,j-1)
            if mod(i-1,(N-1)) == 0
                Matrix(i,i-1) = 0;
            else


 Matrix(i,i-1) = C;
            end
        end
```

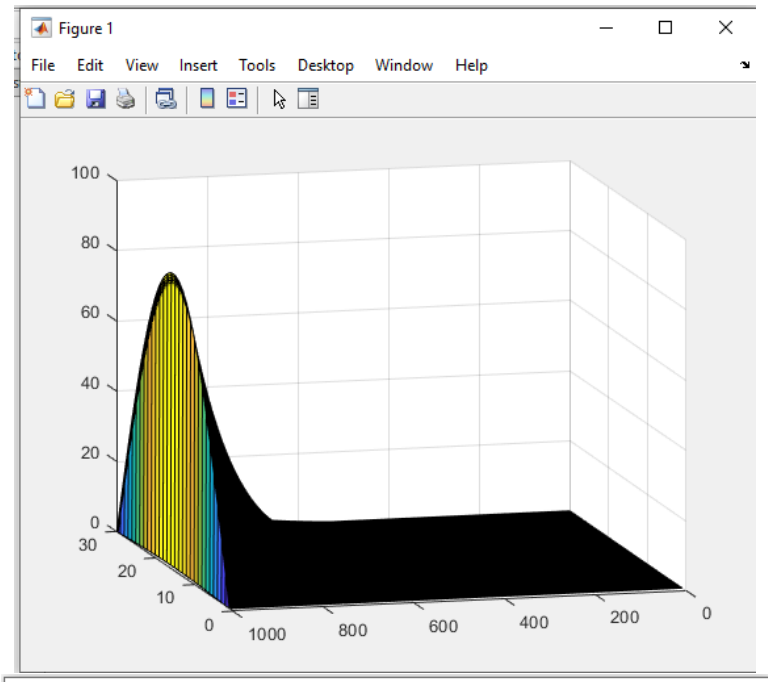
```
        if i<= size -1
            %Coefficients of (i,j+1)

     if mod(i,(N-1)) == 0
                Matrix(i,i+1) = 0;
            else
                Matrix(i,i+1) = D;
            end

        end
end
% RHS and Calculating the solution
U = zeros (1,M-2);
for idx = 1 : M-2
    f = fPassed(dx *idx);
    U(1,idx) = f;
end
%fprintf('%g ', U);
len = (M-2) * (N-1);
b = zeros(len,1);
for idxM = 1:M-2

    if (idxM==1)
    b(idxM,1) = U(1,idxM);
    else
        b(((idxM-1)*(N-1)+1),1) = U(1,idxM);
    end
end

b = -b;
b2 = sparse(b);
S = sparse(Matrix);
sol = S\b2;
%sol = Matrix\b;
%clear b Matrix c dt f dx A B C D c size len idxM idx i
clear b Matrix c f  A B C D c size len b2 S

% Concatonation Region to return the Matrix of the general solution
matrix = zeros(N,M);
submatrix = zeros(N-1,M-2);
for i=2:M-1
    matrix(N,i)= U(1,i-1);
end
sol = flip(sol);
cursor =1;
for i=1:M-2
   subcolumn = sol(cursor:(cursor-1)+N-1);
   cursor = cursor+N-1;
   submatrix(:,M-i)= subcolumn;
end
submatrix(:,1)=[];
matrix(1:N-1, 2:M-1)= submatrix;
u = matrix;
clear  N submatrix cursor sol U subcolumn i matrix
%plotiing
%%subplot(M,N,time)
figure(1)
surf(u); %drawing the  3d Eqn
%TODO: Drawing the 2d Eqn
```

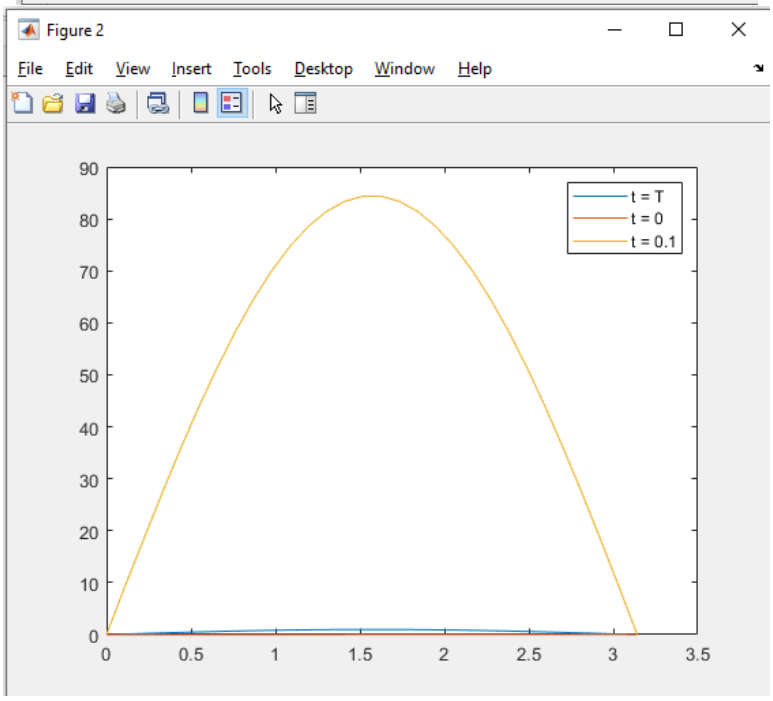
```
%Plotting the Heat equation

figure(2)
vec0 = u(end,:);
vecT = u(1,:);
row = round(0.1/dt);
vec01 = u(row,:);
h = linspace(0,L,M);
a1 = plot(h,vec0);  M1 = "t = T";
hold on
a2= plot(h,vecT);   M2 = "t = 0";
a3= plot(h,vec01);  M3 = "t = 0.1";
legend([a1,a2,a3], [M1, M2,M3]);
hold off
end
```

Task 2: Visualization

M= 30, N= 1000, c=1 , L= Pi, T=0.1

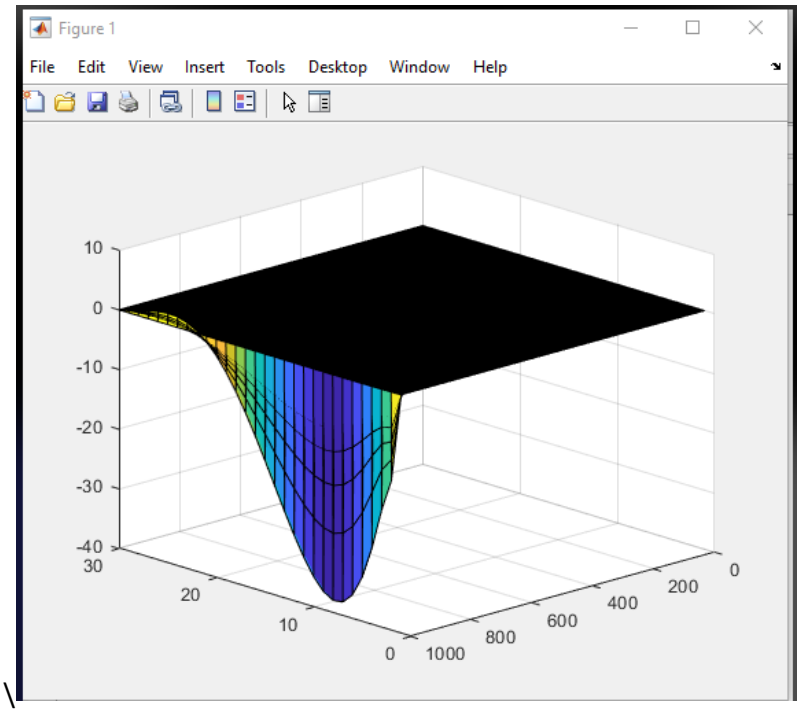
the initial temperature f with f(x) = sin(x)

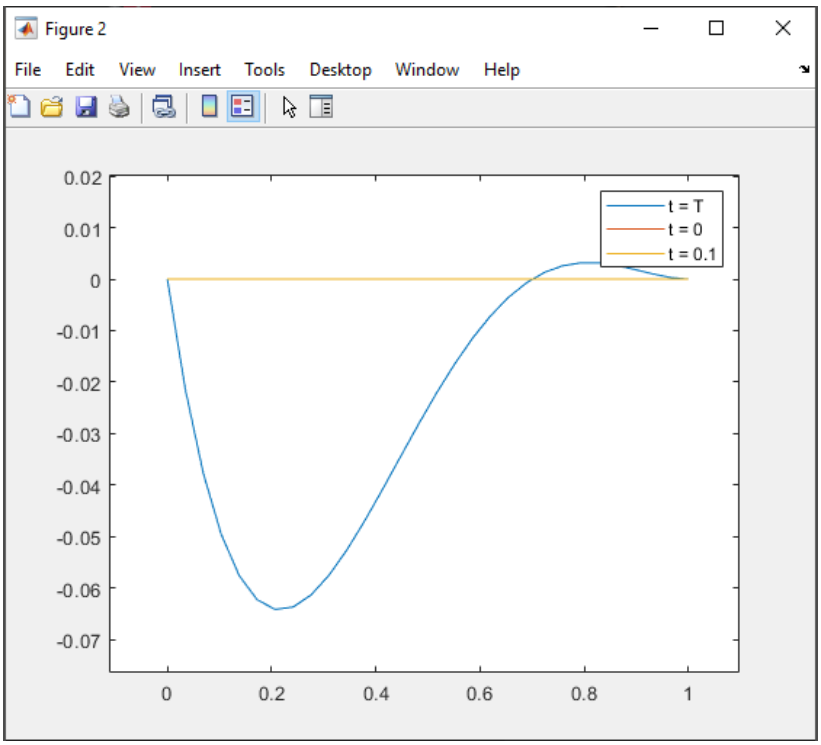




M= 30, N= 10000, c=1 , L= 1 , T=0.42

$$f(x) = x.(x - 0.7).(x - 1)^2$$





M= 20, N= 10000, c=1 , L= 1 , T=7

$$f(x) = x.(x - 0.7).(x - 1)^2$$