Computer Base Engineering Mathematics Lab

Summer Semester 2018

Project 1: Vibration of a string

Project coordinators

Universität Duisburg-Essen

Faculty for Engineering Sciences

Prof. Dr. Johannes Gottschling

Dr. Robert Martin

Saad Alvi, M.Sc.

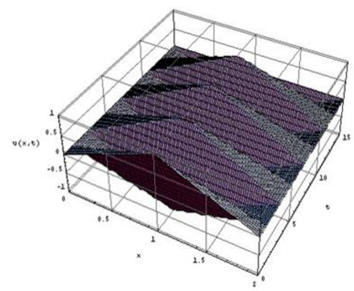
Presented by

Group Number	**11**
Md Abu Saym	3022015
Sadek Dewan	3056001
Mohammad khademul Amin	3022194
Azahar Hossain	3056371
Abdullah Al Mamun	3037571

<u>Project 1 : Vibration of a string</u>

The vibration of a string can be described by the "one-dimensional time dependent wave equation initial value problem"

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2}(x,t) - c^2\,\dfrac{\partial^2 u}{\partial x^2}(x,t) = 0\,, \\ u(0,t) = u(L,t) = 0 \quad \forall\, t \geq 0\,, \\ u(x,0) = f(x),\ u_t(x,0) = g(x) \quad \forall\, x \in [0;L] \end{cases}$$



where u (x; t) is the displacement of the string at the point x and the time t, the real number c > 0 is a material constant, L is the length of the vibrating string, f is the initial displacement and g is the initial velocity of the string; note that all constants and variables are written without dimension.

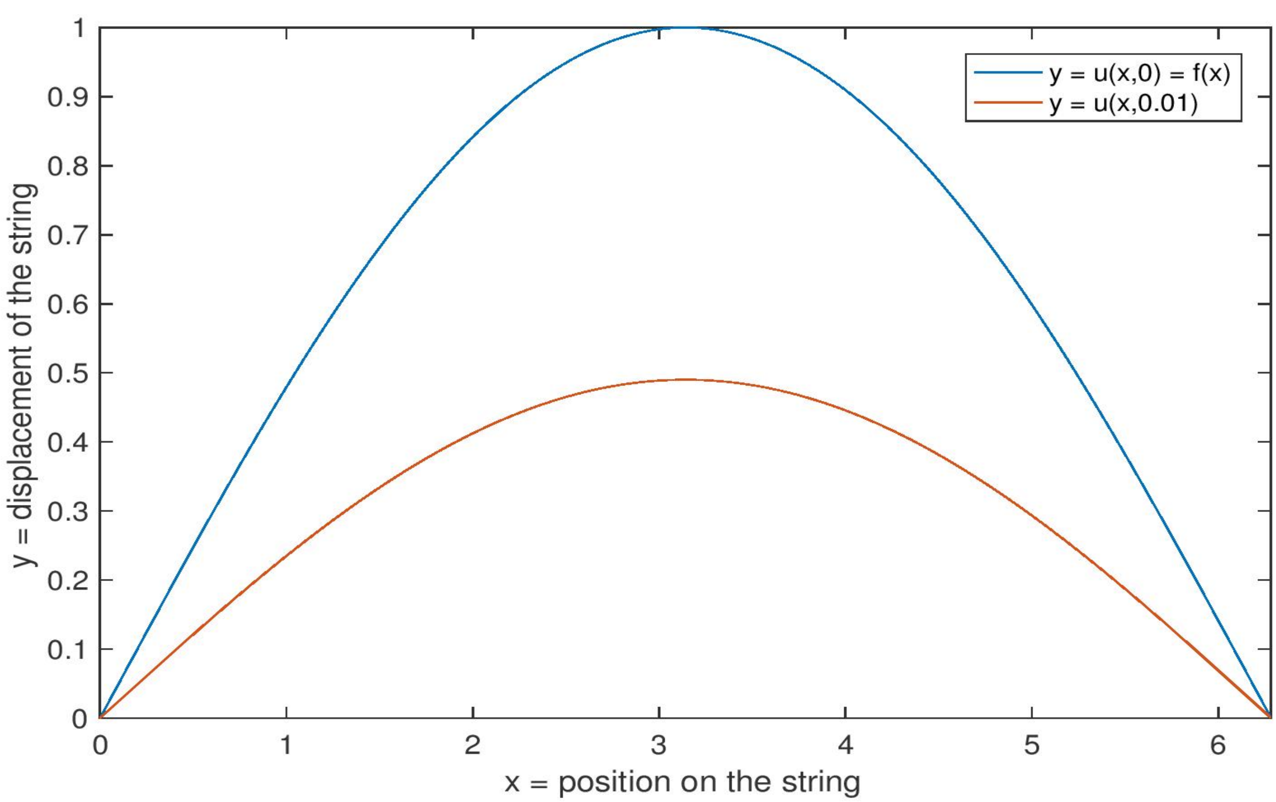


Figure 1: Displacement of a string with length L = 2π at the times t = 0 and t = 0:01.

Task 1:

Using the methods introduced in Section 1.1 of the script ("Motivation"), find (an approximation of) the displacement function u of a vibrating string for

- the grid given in Fig. 2,
- c = 1
- L = 1
- the initial velocity zero,
- the initial displacement f with

$$f(x) = x \cdot (x - 2) \cdot (x - 1)$$

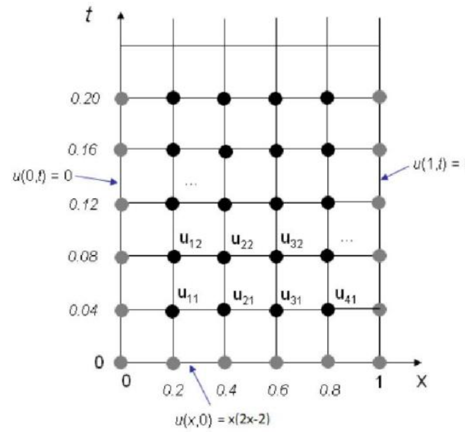


Figure 2: Grid for task 1.

Plot the (three-dimensional) graph of u(·,·) and the (two-dimensional) graphs of u(·,0), u(·,π) and u(·,2π).

Task 2:

Write a matlab function which takes the input arguments

- f (the initial displacement function),
- h (the grid size in x-direction),
- k (the grid size in t-direction),
- T (the endpoint of the time interval)

and returns the (approximate) solution to problem (1)

for c = 1, L = 1, zero initial velocity and the given initial displacement f on the grid given by h, k and T, see Fig. 3.

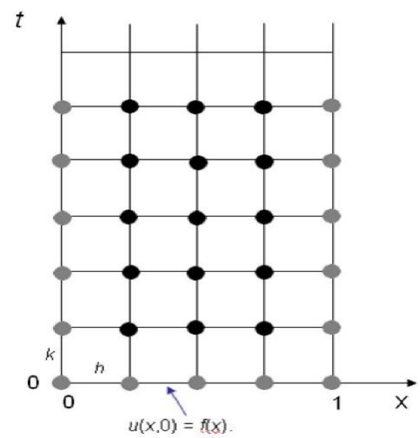


Figure 3: Grid for task 2.

Your project solution must be a file called waveSolution.m of the following form:

```
function [ u ] = waveSolution( f, h, k, T )
    % your code
    % ...
end
```

The return value u must be in the form of a matrix; more specifically: **u(m,n)** must be the value **u((m-1) . h; (n-1). k/T)** of the solution u at the (m; n)-grid point, see Fig. 3.

Theory and mathematical formulae

The main task of this project is to find the displacement of a sting which is fixed in both ends at various points in given grid. We do it using matrices extensively.

Au=B is the main formula used, where A is the coefficient matrix, B is the boundary matrix and u is the resulting temperature matrix.

$h = \Delta x$

$k = \Delta t$

The differential Vibration equation provided is

$$\frac{\partial^2 u}{\partial^2 t}(x,t) - c^2 \frac{\partial^2 u}{\partial^2 x}(x,t) = 0 \quad \text{-------} \ (1)$$

$u(0,t) = u(L,t) = 0 \ \ \forall t \geq 0,$

$u(x;0) = f(x),\ u_t(x,0) = g(x) \ \ \forall x \in [0;L]$

We know from Taylor series expansion we get,

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

And,

$$\frac{\partial^2 u}{\partial^2 x}(x,t) = \frac{u(x+h,y) - 2u(x,y) + u(x-h,y)}{\partial^2 x} \quad \text{-----} \ (3)$$

So, from, (1) we get,

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

$$\Rightarrow \frac{\partial^2 u}{\partial^2 t}(x,t) = c^2 \frac{\partial^2 u}{\partial^2 x}(x,t) \quad \text{-----} \ (4)$$

And from (2) & (3) we get,

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

$$\Rightarrow u(x,y+k) - 2u(x,y) + u(x,y-k) = \left(c^2 \frac{\partial^2 t}{\partial^2 x}\right) * \left(u(x+h,y) - 2u(x,y) + u(x-h,y)\right) \quad \text{-----} \quad (5)$$

Being, $s = (c^2 \frac{\partial^2 t}{\partial^2 x})$

and t+k(Δt) = j+1

x+h(Δx) = i+1

so, our equation now looks like this,

$$u_i^{j+1} = 2(1-s)u_{i,j} + s.u_{i+1,j} + s.u_{i-1,j} - 4_{x,j-1}$$

We can convert this equation in matrix form, Ax = b

Here,

- Main Diagonal = 2(1-s)

Boundary conditions:

- Along the diagonal below main diagonal = s
  - Except = 0 n^{th}, n+$(n-1)^{th}$, n+2$(n-1)^{th}$ = 0
- Along the diagonal above main diagonal = s
  - Except = 0 $(n-1)^{th}$, $2(n-1)^{th}$, $3(n-1)^{th}$ = s
- S is present above and below the main diagonal at (n-1)

Tast 1:

With the given at x and y direction width

Here, x direction = 4

y direction = 5

Being, $s = (c^2 \frac{\partial^2 t}{\partial^2 x})$

$$A = \begin{bmatrix} 2(1-s) & s & 0 & s \\ s & 2(1-s) & s & 0 \\ 0 & s & 2(1-s) & 0 \\ s & 0 & 0 & 2(1-s) \end{bmatrix}$$

$$B = \begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \end{bmatrix} = \begin{bmatrix} 0.2880 \\ 0.3840 \\ 0.3360 \\ 0.1920 \end{bmatrix}$$

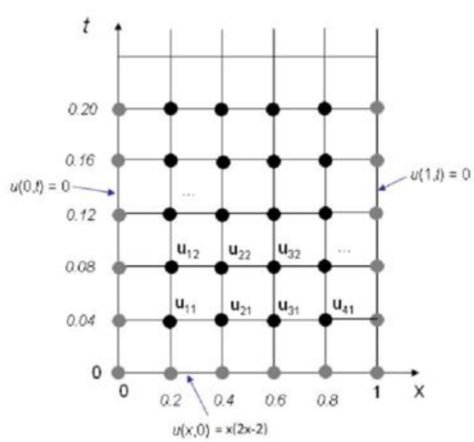


Figure 2: Grid for task 1.

$$X = A\backslash B \quad = \quad \begin{bmatrix} 0.0089 \\ 0.0124 \\ 0.0115 \\ 0.0063 \end{bmatrix}$$

When, ET =0.2

Now we combine all the values of the grid with the boundary values and get the result as:

V =

0	0.0089	0.0124	0.0115	0.0063	0
0	0.0179	0.0247	0.0226	0.0125	0
0	0.0359	0.0491	0.0443	0.0248	0
0	0.0719	0.0974	0.0870	0.0490	0
0	0.1439	0.1934	0.1710	0.0970	0
0	0.2880	0.3840	0.3360	0.1920	0

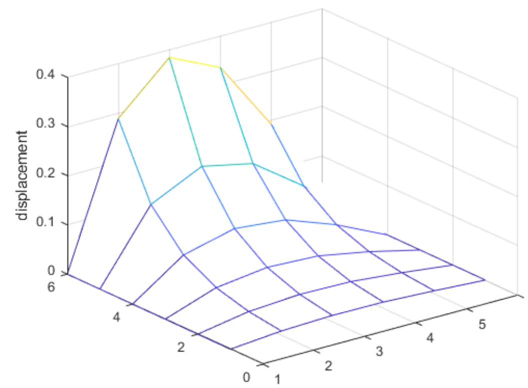


figure: graph of u(. ; .)

When ET = π in 2d graph

The result is:

V =

0	-0.0208	-0.0208	-0.0129	-0.0129	0
0	0.0368	0.0368	0.0228	0.0228	0
0	-0.0651	-0.0652	-0.0403	-0.0402	0
0	0.1150	0.1156	0.0719	0.0708	0
0	-0.1991	-0.2070	-0.1341	-0.1216	0
0	0.2880	0.3840	0.3360	0.1920	0

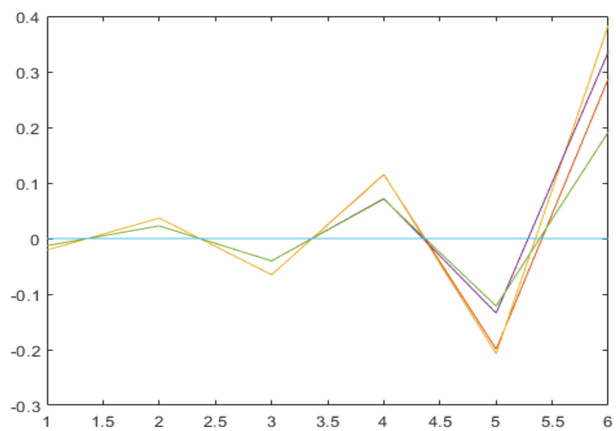


figure: graph of u (. ; π)

When ET = 2π in in 2d graph

The result is:

V =

0	-0.0000	-0.0000	-0.0000	-0.0000	0
0	0.0000	0.0000	0.0000	0.0000	0
0	-0.0002	-0.0002	-0.0001	-0.0001	0
0	0.0021	0.0021	0.0013	0.0013	0
0	-0.0265	-0.0283	-0.0189	-0.0161	0
0	0.2880	0.3840	0.3360	0.1920	0

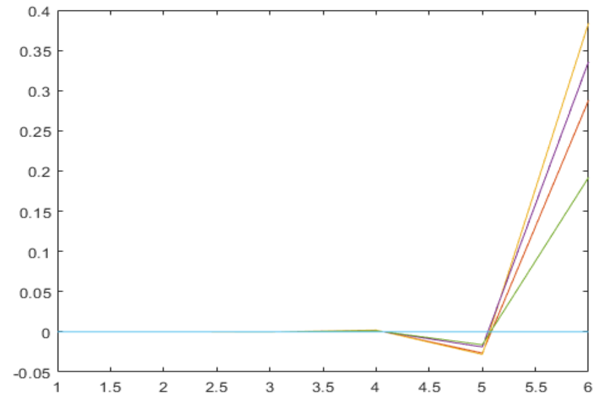


figure: graph of u (. ; 2π)

MATLAB codes for Task 1 is below:

```
% creating variables for grid X and Y directions
initial_time=0;
pie=3.1415;
pie_2=2*pie;
ET=0.2; % end time in time(Y-axis)
Nx=5; %grid division in X-direction
Ny=5;  % grid division in Y-direction
h=1/Nx; % width in X-direction
k=ET/Ny; % width in Y-direction
L=1; % Bars length
C=1; % String constant

% Initializing grid entries
grid_y= [k:k:ET];
grid_x=[h:h:(1-h)]; % grid of valid entries for u(x,0)

% defining coefficient
s=((C*k)/h)^2;

% initialization,declaration and defination of A Matrix
A=zeros(4,4);
I=2*(1-s)*eye(4,4); %
alphas_diag=[s s 0];
corner_diag=[s];
A=I+diag(alphas_diag,1)+diag(alphas_diag,-
1)+diag(corner_diag,3)+diag(corner_diag,-3);


%initialisation, declaration and defination of B Vector
displacement_i=(grid_x.*(grid_x-2).*(grid_x-1))'; % valid
displacement grid entries or "B" vector
initial_grid=displacement_i;
grid_points_n=zeros(4,5);
%ii=repmat(1,4,4);
 for i=1:Ny
     B=displacement_i;
     XX=A\B;
     displacement_i=XX;
 grid_points_n(:,i)=XX;
 end
grid_points_i=horzcat(initial_grid,grid_points_n);
```

```
boundries=zeros(Ny+1,1);
grid_points=flipud(grid_points_i');
V=horzcat(boundries,grid_points,boundries);
mesh(V);
display(V);
zlabel('displacement');
if(ET==pie||ET==pie_2||ET==initial_time)
        plot(V);
end
```

Task 2:

function [u] = waveSolution(f ,h,k,ET)

here, f = displacement

h = width in x direction

k = width in y direction

ET = end time

So, u(m,n)= waveSolution(f ,h,k,ET)

Where, m= (m-1)*h

n= (n-1)*k/T

for example,

when, h = 0.25 dx = 4

k = 0.5 dy = 4

ET = 2

f = exp(x)

given, L = 1 , c = 1

so, for $\frac{x}{1-e^{x}}$ we got,

waveSolution(@ (x) x ./ (exp (x) -1), 0.25, 2, 2)

grid_points =

-0.0200 -0.0256 -0.0183

0.8802 0.7707 0.6714

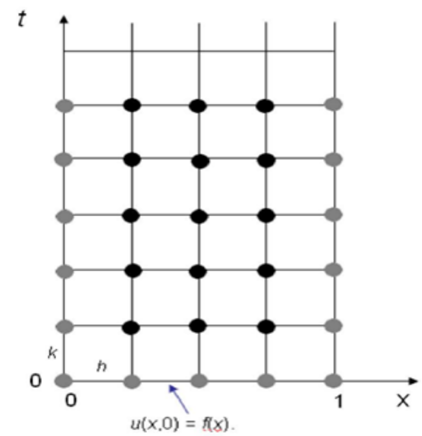


Figure 3: Grid for task 2.

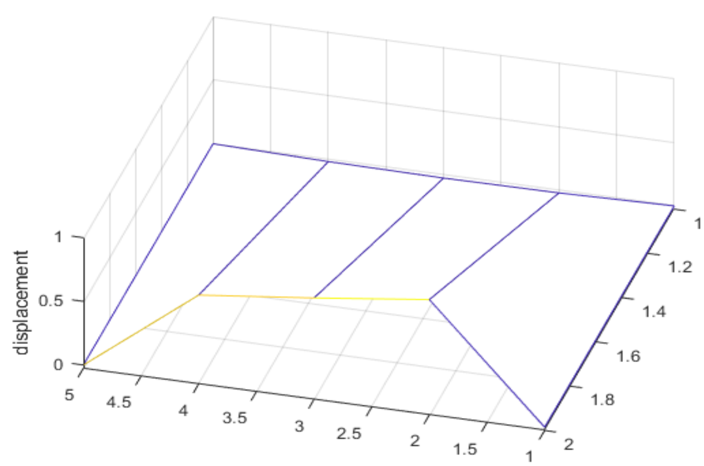


figure : waveSolution(@ (x) x ./ (exp (x) -1), 0.25, 2, 2)

Again,

when, h = 0.2   dx = 5

k = 0.5   dy = 4

ET = 2

f = sin(x)

given, L = 1 , c = 1

so, for sin(x) we got,

waveSolution(@sin, 0.2, 4, 2)

grid_points =

-0.0031	-0.0032	-0.0023	-0.0024
0.1987	0.3894	0.5646	0.7174

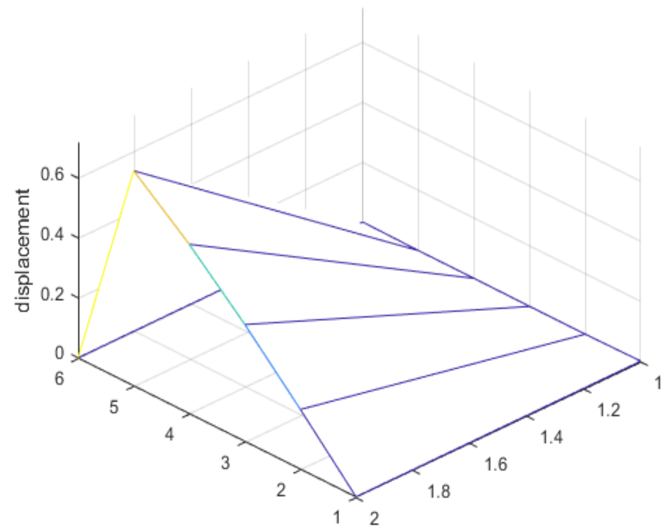


figure: waveSolution(@sin, 0.2, 4, 2)

Matlab codes for waveSolution(f ,h,k,ET) is below:

```
% This Matlab code will create a function which has as parameters a
% displacement function of spring, number of step with given lenght and
% number of steps with given END TIME

function [U] = waveSolution(f,h,k,ET);
L=1; % Bars length
C=1; % String constant

dx=(L/h)-1; % number of steps in X-direction with valid entries M-1
(Note: grid entries are without boundries)
dy=round(ET/k);     % number of steps in y-direction till given END
POINT

% Initializing valid grid entries
grid_x=[h:h:(1-h)]'; % grid of valid entries for u(x,0)

% defining coefficient
s=((C*k)/h)^2;       % calculating the constant from differential
equation

% initialization,declaration and defination of A Matrix
A=zeros(dx,dx);                % memory allocation with M-1 and N-1
entries
I=2*(1-s)*eye(dx,dx);          % initializing and defining main
diagonal with 2*(1-s) (M * times)
diag_1=repmat([s; s; 0],dx,1); % initializing and defining diagonal
with repeting s s 0 (M * times)
diag_2= repmat(s,dx,1);        % initializing and defining diagonal
with all s (M * times)

% definition of A matrix depending on the M entries
if (dx==1)
A=I;
```

```
else if (dx==2)
last_diag= diag_1(1:dx-1,1);
A=I+diag(last_diag,+1)+diag(last_diag,-1);
    else
            last_diag= diag_1(1:dx-1,1);
            last_diag_all_s=diag_2(1:dx-3,1);
            A=I+diag(last_diag,+1)+diag(last_diag,-
1)+diag(last_diag_all_s,+3)+diag(last_diag_all_s,-3);
    end
end

% %initialisation, declaration and defination of B Vector
displacement_i=f(grid_x);     % valid displacement grid entries or "B"
vector
initial_grid=displacement_i;
grid_points_n=zeros(dx,dy); % memory allocation for grid entries
without boundires for loop


 for i=1:dy
     B=displacement_i;
     XX=A\B;
     displacement_i=XX;
 grid_points_n(:,i)=XX;       % definition and declaration of grid
entries without boundries
 end

grid_points_i=horzcat(initial_grid,grid_points_n); % complete grid with
initial displacement entries

boundries=zeros(dy+1,1);      % N-1 times zeros for start and end point
fixed boundries
grid_points=flipud(grid_points_i');
V=horzcat(boundries,grid_points,boundries); % complete grid of [U]=((M-
1)*h,(N-1)*k)
mesh(V);
zlabel('displacement');
display(grid_points);
 end
```

<========== The End ==========>

Thank you