Miniprojekt 3
Beräkningsvetenskap 3
Fredrik Gustafsson
2018-03-07

1 Introduction

Waves transports energy from one place to another. However in order to do this they need a medium to propagate in. One such medium is for example a guitar string in which the waves energy also gives information about with tone the string will give. Analyzing how the wave will behave depending on some variables, tension, density per length unit, diameter etc. can be interesting. To do so one can use the finite difference method by doing the assumption that the wave follows the classic wave equation and giving it some initial conditions and some boundary values. By doing this with an implicit method one gets an opportunity to study the stability of the solution.

2 Theory

2.1 Derivation of FDM method and problem description

To implement the wave equation $\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$ in which u is the displacement in y direction and where c is a constant $c = \sqrt{\frac{T}{\rho}}$ with T being the tension and ρ the linear mass density. Since the FDM works with approximating the different derivatives we start with doing a taylor expansion for the space dependant, x dependant, part. Starting with doing a taylor exapnsion for $x + \Delta x$ gives

$$u^i(x+\Delta x) = u^i(X) + \frac{\partial u^i}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 u^i(x)}{\partial x^2}\Delta x^2 + \frac{1}{3!}\frac{\partial^3 u^i(x)}{\partial x^3}\Delta x^3 + O(\Delta x^4). \quad (1)$$

Doing the same thing for $x - \Delta x$ then gives

$$u^i(x-\Delta x) = u^i(x) - \frac{\partial u^i}{\partial x}(\Delta x) + \frac{1}{2}\frac{\partial^2 u^i(x)}{\partial x^2}(\Delta x)^2 - \frac{1}{3!}\frac{\partial^3 u^i(x)}{\partial x^3}\Delta x^3 + O(\Delta x^4). \quad (2)$$

Since we want to replace the expression $\frac{\partial^2 u}{\partial x^2}$ we summate equ. 1 and 2 and replacing $x + \Delta x$ and $x - \Delta x$ with $j + 1$ and $k - 1$ respectively to arrive at

$$\frac{\partial^2 u_j^i}{\partial x^2} = \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{\Delta x^2}. \quad (3)$$

Further by combining 1, 2 and 3 we get the truncation error

$$\tau(x) = u'' + O(\Delta x^2) - f(x) = O(\Delta x^2) \quad (4)$$

For the time-dependant part we do the exact same thing as we did for the space dependant one, but replacing x with t. This holds due to the fact that we want to expand u around $t \pm \Delta t$ just as for $x \pm \Delta x$

$$\frac{\partial^2 u_j^i}{\partial t^2} = \frac{u_j^{i+1} - 2u_j^i + u_j^{i-1}}{\Delta t^2}. \tag{5}$$

Substituting 3 and 4 into our original wave equation then gives us our final FDM approximation

$$\frac{u_j^{i+1} - 2u_j^i + u_j^{i-1}}{\Delta t^2} = c^2 \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{\Delta x^2}. \tag{6}$$

This approximation is of order 2 in time and order 2 in space. The initial condition consisted of $u(x,0) = \hat{u}\frac{x}{\hat{x}}$ $x \in [0, \hat{x}]$, $u(x,0) = \hat{u}\frac{x-L}{\hat{x}-L}$ $x \in [\hat{x}, L]$, where û is the initial displacement, x̂ the length from the end at which the string is plucked and L the total length of the string. These two initial values are fine as is however there was a third one $\frac{\partial u}{\partial t}(x,0) = 0$. Since the third one includes a derivative we need to work it out with the same method as before, forward/backward taylor expansion and the substitution of $t + \Delta t$ and $t - \Delta t$ as before giving us

$$\frac{\partial u_j^i}{\partial t} = \frac{u_j^{i+1} - u_j^{i-1}}{2\Delta t} = 0, \tag{7}$$

this in turn gives us $u_j^{i+1} = u_j^{i-1}$.

2.2 Stability analysis

To start with the stability analysis we first summerise equation 5 to arrive at a recursion formula

$$u_j^{i+1} = 2(1 - \lambda^2)u_j^i - u_j^{i-1} - \lambda^2(u_{j-1}^i + u_{j+1}^i) \tag{8}$$

in which $\lambda = c\frac{\Delta x}{\Delta t}$. In this form the stability analysis will become a bit easier. By then using the anzats $u_j^i = Q^i e^{jl\beta\Delta x}$ and inserting it into equ.7 using some trigonometry and simplifying we get the following equation

$$Q + 2(2\lambda^2 sin^2(\beta\Delta x/2) - 1) + Q^{-1} = 0. \tag{9}$$

The roots for this equation is given by

$$Q_\pm = 1 - 2\lambda^2 sin^2(\beta\Delta x/2) \pm \sqrt{(1 - 2\lambda^2 sin^2(\beta\Delta x/2))^2 - 1} \tag{10}$$

From this we arrive at

$$|2\lambda sin^2(\beta\Delta x/2) - 1| \leq 1 \tag{11}$$

Which is the same as$\lambda^2 sin^2(\beta\Delta x/2) \leq 1$. Since the sin^2 term is bounded by 1 we get the final condition of $\lambda \leq 1$ or, by substituting lambda back $c^2\Delta t^2 \leq \Delta x^2$, where $c = \frac{T}{\rho}$.

3 Method and implementation

To solve the wave equation an explicit recursive formula was implemented in matlab

$$u_j^{i+1} = 2u_j^i - u_j^{i-1} + \alpha(u_{j+1}^i - 2u_j^i + u_{j-1}^i \tag{12}$$

where $\alpha = (c\frac{\Delta t}{\Delta x})^2$. By modifying it a bit we get

$$u_{(i,j)} = 2u_{(i-1,j)} - u_{(i-2,j)} + \alpha(u_{(i-1,j+1)} - 2u_{(i-1,j)} + u_{(i-1,j-1)} \tag{13}$$

The solution for the problem will be discrete in both time and space. To do this we create a nxn matrix where the rows are the different times, row 0 being $t = 0$, and the columns being the discrete points in space, column 0 being $x = 0$. This matrix can then be filled with the initial conditions. For $t = 0$ the non derivative initial conditions will be used. The boundary conditions can also easily be implemented by setting the first and last column to zero. For the first time step corresponding to row 2. This is done using equation 9

$$u_{(2,j)} = 2u_{(1,j)} - u_{(0,j)} + \alpha(u_{(1,j+1)} - 2u_{(1,j)} + u_{(1,j-1)} \tag{14}$$

and since we know that $u_j^0 = u_j^2$ we arrive at

$$u_{(2,j)} = u_{(1,j)} + \frac{1}{2}\alpha(u_{(1,j+1)} - 2u_{(1,j)} + u_{(1,j-1)} \tag{15}$$

from which we can fill in the second row by looping over the columns j. Finally we can start to fill the matrix with the solutions by looping equ. 9 over i and j, rows and columns, starting from row 3.

To visualize the solution a simple function can be implemented that plots the solution for each timestep and when run will show how the wave will behave.

4 Results

By running the program with the standard parameters as shown in the instructions and having the diameter being $0.75mm$, initial amplitude,û, 1cm and x̂ being 1dm, with 100 nodes in the space disretization and taking 10^6 steps in time one gets figure 1 shown below.

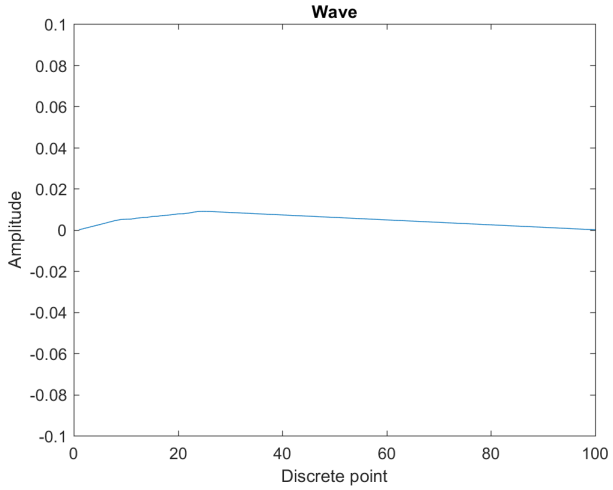

Figure 1. Snapshot of wave as simulated in matlab

This is a normal looking wave traveling along the string where the y-axis is the amplitude and the x-axis is the part of the string. By "playing around" with the variables to see what and how they affect things, while keeping the time step and space step fixed, and trying to see when they make the solution unstable table 1 could be constructed.

Table 1. Table showing the tunable variables, how they influence the wave and at which point the solution becomes unstable when having constant 100 steps in space and 10^6 steps in time

Variable	Influence	Values for which solution is unstable
Length(L)	Shorter create more interference	Bounded by $\hat{x}$ but won't become unstable
ρ	larger ρ decrease speed	ρ<24
String tension (T)	larger T increase speed	T >139550
String diameter	larger d decrease speed	d<$0.045 * 10^{-3}$
$\hat{u}$	Increase amplitude	non
$\hat{x}$	Smaller increase resonance	$\hat{x}$<0.004 breaks program

Figure 2 below shows the solution for the first couple of timesteps around $3 * 10^{-}4$s to $12 * 10^{-}4$s showing that the solution has started to become umstable and when continued to be ran, or by choosing parameters to make $c^2\Delta t^2 > \Delta x^2$, figure 3 can be observed. Figure 3 then shows solution being unstable from the beginning but a similar plot would be given if the solution generating figure 2 would continue to run, the amplitude oscilating between $\pm\infty$.

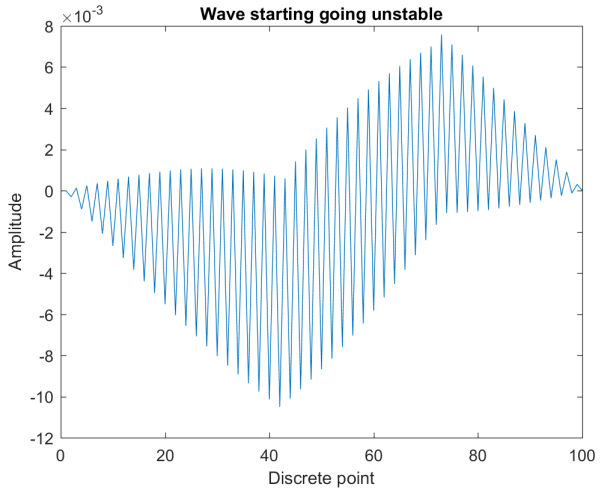

Figure 2. Solution starting to become unstable as simulation runs.

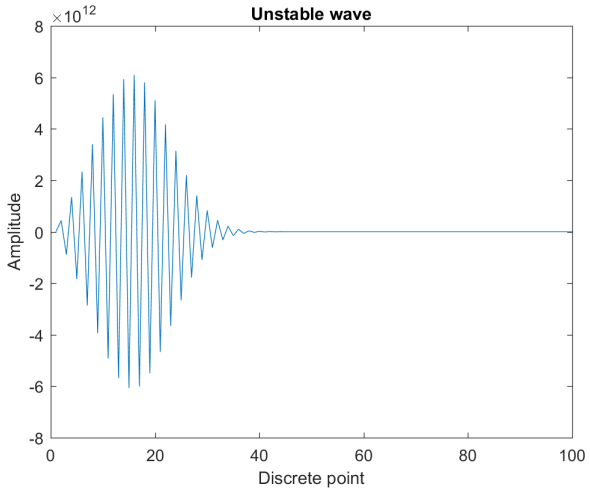

Figure 3. Solution unstable from beginning.

Another interesting aspect is when the point at which the string is "plucked" moves closer to the end point resonance starts to happening as in the wave doesn't die out after the wave is reflected this is shown in figure 5 below.

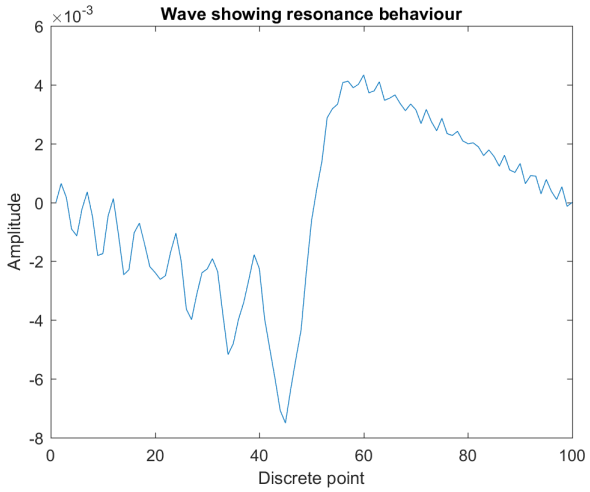

Figure 5. Wave showing resonance behaviour.

5 Discussion and conclussion

From testing the different variables one can see that the method is unstable which was to be expected since it's an explicit method. If an inplicit method had been used it should have been more stable, however this would also be more computational costly. As one can see exceeding some threshold values makes the solution unstable, although some of the values should be seen as physically impossible e.g ρ<24. These numbers could be theoretically calculated from the stability criteria when doing this and comparing with the practical values they are fairly close. This then shows that the stability criteria is correct and that as long as it is full filled the method is stable. Further one can also see that the $\hat{x}$ must be larger than $L/\Delta x$ otherwise the program breaks. All of this shows then that using FDM to solve the wave equation on a guitar string is more than doable.

If one were to do this again testing an implicit method would be interesting to see if we then can get around the limits for stability and also to compare the computation time. Another interesting thing would be to instead model the string such that it consists of n-number of harmonic oscillators and solve the coupled vibration problem this would give rise to and compare that with solving the wave quation since in many ways a guitar string could be said to consist of infinitely many harmonic oscillators.

6 Appendix

Matlab code:
To plot solution:

```
function plotWave(u)
%Plot the solution u to FDM method for wave on a string

close all;
[nt,nx]=size(u);
figure

hold on;
for i = 3 : nt %Starting at third row of u, since the earlier one was fillled in
    plot(u(i,:));
    axis([0 nx -0.1 0.1]); %Scale of axis, 0 nx is x-axis, -0.1 0.1 is amplitude
    pause(0.000800); %in order to see each step
    hold off;
end

end
```

Main program:

```
L =0.63; %Bounded by xl, longer gives one wavefront moving while shorter gives tw
roh = 7800; %unstable for roh<24 when taking 1000 time steps, higher roh gives s
T = 440; %lower decrease speed T=139550 or more unstable
d = 0.75*10^-3; %0.5-1, lower increase speed, unstable for d<0.045*10^-3
c = sqrt(T/(pi*d^2/4*roh));
uh = 0.01; %distance lifted, amplitude of wave
xl = 0.1; %distance from end, closer to zero gives more resonance, wave won't di

time = 1; %simuleringstid
nx = 100; %antal noder i rumet
nt = 1000000; % antal steg i tid
dx = L/(nx-1); % steglngd i rum
dt = time/(nt-1); % steglngd i tid
xld = round(xl/dx); %Discretisering f r brytpunkt f r BV

r = (c*(dt/dx))^2;

%Solution matrix
rx=ceil(L/dx)+1;
rt = ceil(time/dt)+1;
u=zeros(rt,rx);

%Implement BV non derivatives
```

```
for i =1:xld
    u(1,i) = uh*(i-1)*dx/xl;
end

for i = xld:nx
    u(1,i) = uh*(((i-1)*dx-L)/(xl-L));
end

%Implemend BV derivative
for i =2:nx-1
    u(2,i) = u(1,i)+1/2*r*(u(1,i+1)-2*u(1,i)+u(1,i-1)); %Solution first time ste
end

%Populate Matrix
for i = 3:nt
    for j = 2:nx-1
        u(i,j)=2*u(i-1,j)-u(i-2,j)+r*(u(i-1,j+1)-2*u(i-1,j)+u(i-1,j-1));
    end
end

%Visualize
plotWave(u)
```