IN5270
WAVE PROJECT

ERIK JOHANNES B. L. G. HUSOM

OCTOBER 15, 2019

Contents

1. Discretization of equations

In this project we have the following 2D linear wave equation with damping:

$$\frac{\partial^2 u}{\partial t^2} + b\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(q(x,y)\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(q(x,y)\frac{\partial u}{\partial y}\right) + f(x,y,t) \tag{1}$$

The boundary condition is

$$\frac{\partial u}{\partial n} = 0 \tag{2}$$

and the inital conditions are

$$u(x,y,0) = I(x,y) \tag{3}$$
$$u_t(x,y,0) = V(x,y) \tag{4}$$

To use this in our computer calculations, we need a discretized version. Since we have a variable coefficient q, we write the inner derivatives (by using a centered derivative) as:

$$\phi_x = q[x,y]\frac{\partial u}{\partial x}, \quad \phi_y = q[x,y]\frac{\partial u}{\partial y} \tag{5}$$

Then we get

$$\left[\frac{\partial \phi_x}{\partial x}\right]_i^n \approx \frac{\phi_{x,i+1/2} - \phi_{x,i-1/2}}{\Delta x} \tag{6}$$
$$\left[\frac{\partial \phi_y}{\partial y}\right]_j^n \approx \frac{\phi_{y,j+1/2} - \phi_{y,j-1/2}}{\Delta y} \tag{7}$$

We then write

$$\phi_{x,i+1/2} = q_{i+1/2,j}\left[\frac{\partial u}{\partial x}\right]^n_{i+1/2} \approx q_{i+1/2,j}\frac{u_{i+1,j} - u^n_{i,j}}{\Delta x} \tag{8}$$

$$\phi_{x,i-1/2} = q_{i-1/2,j}\left[\frac{\partial u}{\partial x}\right]^n_{i-1/2} \approx q_{i-1/2,j}\frac{u_{i,j} - u^n_{i-1,j}}{\Delta x} \tag{9}$$

$$\phi_{y,j+1/2} = q_{i,j++1/2}\left[\frac{\partial u}{\partial y}\right]^n_{j+1/2} \approx q_{i,j+1/2}\frac{u_{i,j+1} - u^n_{i,j}}{\Delta y} \tag{10}$$

$$\phi_{y,j-1/2} = q_{i,j-1/2}\left[\frac{\partial u}{\partial y}\right]^n_{j-1/2} \approx q_{i,j-1/2}\frac{u_{i,j} - u^n_{i,j-1}}{\Delta y} \tag{11}$$

To obtain q at the half steps we use the arithmetic mean $q_{i+1/2} \approx \frac{1}{2}(q_i + q_{i+1}$ and $q_{i-1/2} \approx \frac{1}{2}(q_i + q_{i-1}$.

This is then used to discretize equation 1:

$$\begin{aligned}
\frac{u_{i,j}^{n+1}-2u_{i,j}^{n}+u_{i,j}^{n-1}}{\Delta t^2}+b\frac{u_{i,j}^{n+1}-u_{i,j}^{n-1}}{2\Delta t} &= \frac{1}{\Delta x}\left(q_{i+\frac{1}{2},j}\left(\frac{u_{i+1,j}^{n}-u_{i,j}^{n}}{\Delta x}\right)-q_{i-\frac{1}{2},j}\left(\frac{u_{i,j}^{n}-u_{i-1,j}^{n}}{\Delta x}\right)\right)\\
&+\frac{1}{\Delta y}\left(q_{i,j+\frac{1}{2}}\left(\frac{u_{i,j+1}^{n}-u_{i,j}^{n}}{\Delta y}\right)-q_{i,j-\frac{1}{2}}\left(\frac{u_{i,j}^{n}-u_{i,j-1}^{n}}{\Delta y}\right)\right)+f_{i,j}^{n}\\
\frac{u_{i,j}^{n+1}-2u_{i,j}^{n}+u_{i,j}^{n-1}}{\Delta t^2} &= -b\frac{u_{i,j}^{n+1}-u_{i,j}^{n-1}}{2\Delta t}\\
&+\frac{1}{\Delta x^2}\left(\frac{1}{2}\left(q_{i+1,j}-q_{i,j}\right)\left(u_{i+1,j}^{n}-u_{i,j}^{n}\right)-\frac{1}{2}\left(q_{i,j}-q_{i-1,j}\right)\left(u_{i,j}^{n}-u_{i-1,j}^{n}\right)\right)\\
&+\frac{1}{\Delta y^2}\left(\frac{1}{2}\left(q_{i,j+1}-q_{i,j}\right)\left(u_{i,j+1}^{n}-u_{i,j}^{n}\right)-\frac{1}{2}\left(q_{i,j}-q_{i,j-1}\right)\left(u_{i,j}^{n}-u_{i,j-1}^{n}\right)\right)+f_{i,j}^{n}\\
u_{i,j}^{n+1}-2u_{i,j}^{n}+u_{i,j}^{n-1} &= \frac{-b\Delta t}{2}(u_{i,j}^{n+1}-u_{i,j}^{n-1})\\
&+\frac{\Delta t^2}{2\Delta x^2}\left(\left(q_{i+1,j}-q_{i,j}\right)\left(u_{i+1,j}^{n}-u_{i,j}^{n}\right)-\left(q_{i,j}-q_{i-1,j}\right)\left(u_{i,j}^{n}-u_{i-1,j}^{n}\right)\right)\\
&+\frac{\Delta t^2}{2\Delta y^2}\left(\left(q_{i,j+1}-q_{i,j}\right)\left(u_{i,j+1}^{n}-u_{i,j}^{n}\right)-\left(q_{i,j}-q_{i,j-1}\right)\left(u_{i,j}^{n}-u_{i,j-1}^{n}\right)\right)+\Delta t^2 f_{i,j}^{n}\\
u_{i,j}^{n+1}+\frac{b\Delta t}{2}u_{i,j}^{n+1} &= 2u_{i,j}^{n}-u_{i,j}^{n-1}+\frac{b\Delta t}{2}u_{i,j}^{n-1}\\
&+\frac{\Delta t^2}{2\Delta x^2}\left(\left(q_{i+1,j}-q_{i,j}\right)\left(u_{i+1,j}^{n}-u_{i,j}^{n}\right)-\left(q_{i,j}-q_{i-1,j}\right)\left(u_{i,j}^{n}-u_{i-1,j}^{n}\right)\right)\\
&+\frac{\Delta t^2}{2\Delta y^2}\left(\left(q_{i,j+1}-q_{i,j}\right)\left(u_{i,j+1}^{n}-u_{i,j}^{n}\right)-\left(q_{i,j}-q_{i,j-1}\right)\left(u_{i,j}^{n}-u_{i,j-1}^{n}\right)\right)+\Delta t^2 f_{i,j}^{n}\\
\left(1+\frac{b\Delta t}{2}\right)u_{i,j}^{n+1} &= 2u_{i,j}^{n}-\left(1-\frac{b\Delta t}{2}\right)u_{i,j}^{n-1}\\
&+\frac{\Delta t^2}{2\Delta x^2}\left(\left(q_{i+1,j}-q_{i,j}\right)\left(u_{i+1,j}^{n}-u_{i,j}^{n}\right)-\left(q_{i,j}-q_{i-1,j}\right)\left(u_{i,j}^{n}-u_{i-1,j}^{n}\right)\right)\\
&+\frac{\Delta t^2}{2\Delta y^2}\left(\left(q_{i,j+1}-q_{i,j}\right)\left(u_{i,j+1}^{n}-u_{i,j}^{n}\right)-\left(q_{i,j}-q_{i,j-1}\right)\left(u_{i,j}^{n}-u_{i,j-1}^{n}\right)\right)+\Delta t^2 f_{i,j}^{n}\\
u_{i,j}^{n+1} &= \frac{1}{1+\frac{b\Delta t}{2}}\,(\;2u_{i,j}^{n}-\left(1-\frac{b\Delta t}{2}\right)u_{i,j}^{n-1}\\
&+\frac{\Delta t^2}{2\Delta x^2}\left(\left(q_{i+1,j}-q_{i,j}\right)\left(u_{i+1,j}^{n}-u_{i,j}^{n}\right)-\left(q_{i,j}-q_{i-1,j}\right)\left(u_{i,j}^{n}-u_{i-1,j}^{n}\right)\right)\\
&+\frac{\Delta t^2}{2\Delta y^2}\left(\left(q_{i,j+1}-q_{i,j}\right)\left(u_{i,j+1}^{n}-u_{i,j}^{n}\right)-\left(q_{i,j}-q_{i,j-1}\right)\left(u_{i,j}^{n}-u_{i,j-1}^{n}\right)\right)+\Delta t^2 f_{i,j}^{n}\;)
\end{aligned}$$

The modified scheme for the first step will find by using the initial conditions:

$$\begin{aligned}
u_t(x,y,0) &= V(x,y)\\
\frac{u_{i,j}^{0}-u_{i,j}^{-1}}{\Delta t} &= V_{i,j}\\
u_{i,j}^{-1} &= u_{i,j}^{0}-\Delta t V_{i,j}
\end{aligned}$$

The modified scheme at the boundary points is found by using the boundary conditions:

$$\frac{\partial u}{\partial n} = 0$$
$$\frac{u^n_{N_x+1,j} - u^n_{N_x-1,j}}{2\Delta x} = 0$$
$$u^n_{N_x+1,j} = u^n_{N_x-1,j}$$

2. Constant solution

The file `test.py` contains test of a constant solution for both the scalar and the vectorized version. With a constant solution $u(x, y, t) = c$, the derivatives become zero, so if we set $f = 0$, our exact solution becomes the same as the initial condition I, which has to be set to $I = c$. In my test case, I have chosen $c = 8$. The code also uses `nose.tools` to assert that the difference between the exact and numerical solution is zero.

3. Undampened waves

For undampened waves, we have the exact solution

$$u_e(x, y, t) = A\cos(k_x x)\cos(k_y y)\cos(\omega t), \tag{12}$$

where

$$k_x = \frac{m_x \pi}{L_x}, \tag{13}$$
$$k_y = \frac{m_y \pi}{L_y}. \tag{14}$$

The variables m_x and m_y are arbitrary numbers, and A is the amplitude. In order to find ω, we must insert the exact solution into the original wave equation. I have chosen to use $q = 1, f = 0, b = 0$. We get

$$\frac{\partial u_e}{\partial t} = -\omega A\cos(k_x x)\cos(k_y y)\sin(\omega t) \tag{15}$$
$$\frac{\partial^2 u_e}{\partial t^2} = \omega^2 Acos(k_x x)\cos(k_y y)\cos(\omega t) = \omega^2 u_e \tag{16}$$
$$\frac{\partial u_e}{\partial x} = -k_x A\sin(k_x x)\cos(k_y y)\cos(\omega t) \tag{17}$$
$$\frac{\partial^2 u_e}{\partial x**2} = k_x^2 Acos(k_x x)\cos(k_y y)\cos(\omega t) = k_x^2 u_e. \tag{18}$$

We get the similar result with y:

$$\frac{\partial^2 u_e}{\partial y^2} = k_x^2 Acos(k_x x)\cos(k_y y)\cos(\omega t) = k_y^2 u_e. \tag{19}$$

This gives us:

$$\omega^2 u_e = k_x^2 u_e + k_y^2 u_e \tag{20}$$
$$\omega = \sqrt{k_x^2 + k_y^2}. \tag{21}$$

I have implemented the undampened standing waves in the function `test_undampened`. I have chosen a series of h-values, in the list `h_list`, and calculates the error ϵ from the exact solution, compared to the numerical solution. The convergence rate r_i for each iteration i is then computed with the formula

$$r_i = \frac{\log \epsilon_i/\epsilon_{i-1}}{h_i/h_i - 1}. \tag{22}$$

The error ϵ is computed by

$$\epsilon = \sqrt{\Delta x \Delta y \Delta t \sum (u_e - u_{\text{num}})} \tag{23}$$

where u_e is the exact solution, and u_{num} is the numerical solution.

We expect the numerical convergence rate to be 2 (because the discretization are accurate to the second order), and my program shows that the numerical solution after some iterations converge to 2:

```
Convergence rate:
[1.99812038 2.00023556 2.00001278 1.99992403]
Test of undampened solution:
Difference:  7.597295288186423e-05
```

As we can see from the printout, the difference between the theoretical convergence rate 2, and the computed one, is of the order 10^{-5}.

4. Manufactured solution

I did not manage to get results that were consistent with the exact solutions. I prioritized to finish the physical problem solving instead of finding the bugs in this part of the project.

For the manufactured solution, I used `sympy` to obtain the following expression:

$$f(x,y,t) = A * kx * *2 * \cos(kx * x) * \cos(ky * y) * \cos(\omega * t) \tag{24}$$
$$+ A * ky * *2 * \cos(kx * x) * \cos(ky * y) * \cos(\omega * t) \tag{25}$$
$$- A * \omega * *2 * \cos(kx * x) * \cos(ky * y) * \cos(\omega * t) \tag{26}$$

5. Physical problem

In this task I will investigate how two different kinds of bottom surfaces will influence how the wave moves.

The first bottom shape, `B1`, is given by the formula:

$$B = B0 + B_a \exp\big(- (\frac{x - B_{mx}}{B_s})^2 - (\frac{y - B_{my}}{b * B_s})^2 \big), \tag{27}$$

where b is a scaling parameter, and the other parameters affect the specific shape of the bottom.

The other bottom shape I tested, `B2`, is given by:

$$B = B_0 + B_a \cos\big(\pi \frac{x - B_{mx}}{2B_s}\big) \cos\big(\pi \frac{y - B_{my}}{2B_s}\big). \tag{28}$$

Both of these bottom shapes are two of the shapes suggested in the project description. In each case I set

$$B_0 = 0, B_a = 2.5, B_{mx} = B_{my} = 1, B_s = 0.4, b = 1$$

I chose to simulate the waves for a period of $T = 2$, with a time step of $0.1h$, and $\Delta x = \Delta y = h$. I chose different values of h, which are presented for each of the plots in this report.

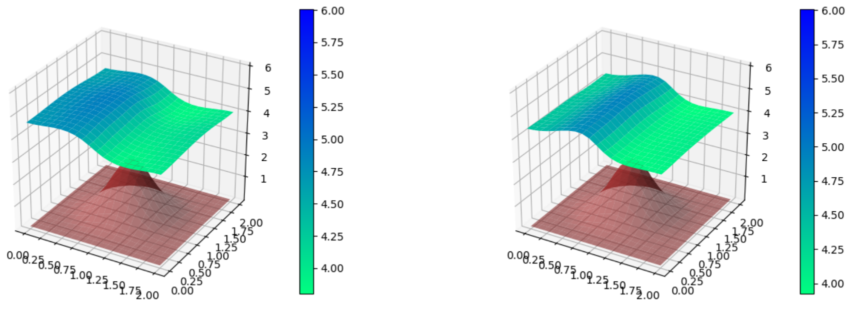

FIGURE 5.1. Bottom shape `B1`, $h = 0.1$, for the first (left) and last (right) timestep.

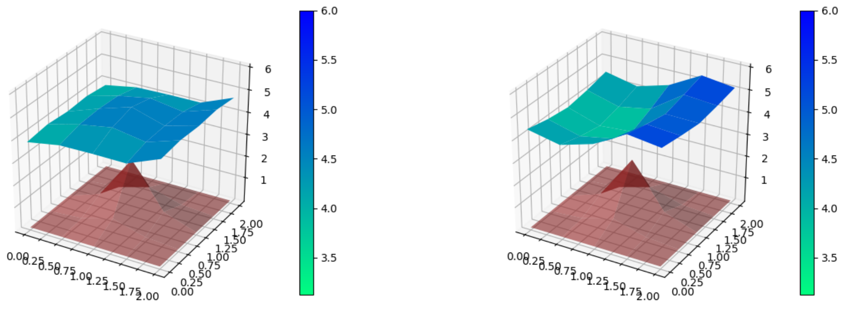

FIGURE 5.2. Bottom shape `B1`, $h = 0.4$, for the first (left) and last (right) timestep.

To simplify the presentation of the results, I have chosen to produce gifs of the wave motions, but I will also include some selected plots for different runs in this report. Here is a list of the gifs, which are uploaded to the same GitHub repository as this report (I was not able to produce gifs with $h < 0.1$, because the process got killed by my laptop):

- `wave1h01.gif`, shape `B1`, $h = 0.1$.
- `wave1h04.gif`, shape `B2`, $h = 0.4$.
- `wave2h01.gif`, shape `B1`, $h = 0.1$.
- `wave2h04.gif`, shape `B2`, $h = 0.4$.

Figure 5.1 shows the `B1` shape with $h = 0.1$, while figure 5.2 shows the same shape with $h = 0.4$. In figure 5.3 and 5.4 we see bottom shape `B2` with respectively $h = 0.1$ and $h = 0.4$. It is easier to inspect the results by looking at gifs (which are placed in the same repository as this report), and we can observe that the wave motion appears much smoother when we use a higher resolution for the simulations. In figure 5.5 we can see the to bottom shapes for an even higher resolution, with $h = 0.01$.

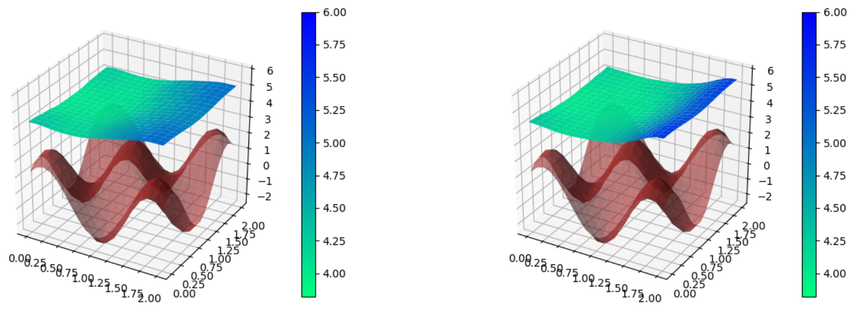

FIGURE 5.3. Bottom shape B2, $h = 0.1$, for the first (left) and last (right) timestep.

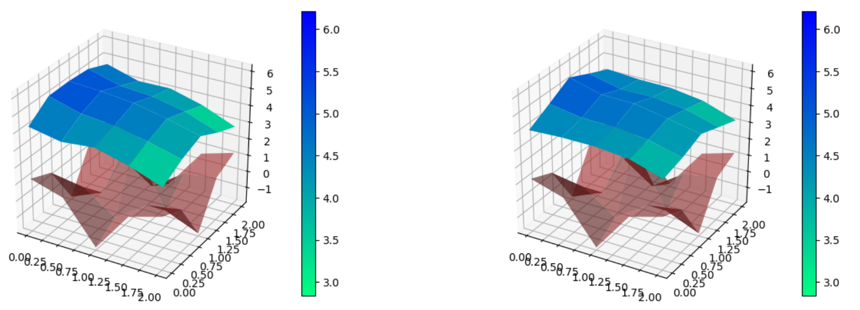

FIGURE 5.4. Bottom shape B2, $h = 0.4$, for the first (left) and last (right) timestep.

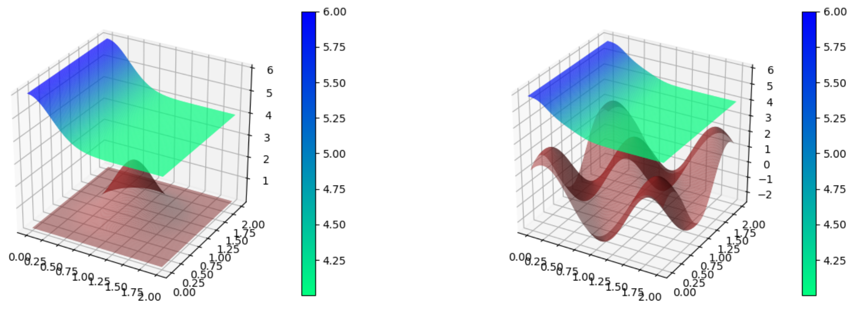

FIGURE 5.5. Bottom shape B1 (left), and bottom shape B2 (right), $h = 0.01$.