

Effects of Hill Shapes on Tsunami Simulations

Kristian Pedersen and Gustav Baardsen

October 15, 2012

Abstract

In this project we study how the shape of hills at the sea bottom affects numerical and physical properties of simulations of earthquake-generated tsunamis. The tsunamis are modelled using two-dimensional wave equations, and a finite difference scheme is used to solve the partial differential equations.

Contents

1	Introduction	1
2	Mathematical model	2
3	Numerical scheme	2
3.1	Numerical scheme for the inner points	2
3.2	Numerical scheme for the first time step	3
3.3	Numerical scheme for the boundary	3
3.4	Approximating $q(x,y)$ outside the grid	4
4	Implementation	4
5	Numerical experiments	4
5.1	Verification	4
5.2	One-dimensional simulations	6
5.3	One-dimensional hill w/ gap simulations	10
5.4	Plots of hills w/ gap	13
5.5	One-dimensional hill w/ gap simulation conclusions	13
5.6	Two-dimensional simulations	13
6	Conclusions	15

1 Introduction

In this text we will address a two-dimensional, standard linear wave equation, with damping and reflecting boundaries. We will develop a scheme for solving it and apply it to the problem of a tsunami over an uneven seabed. Especial focus will be put on the question of which kinds of seabeds causes numerical instability.

2 Mathematical model

For (x, y) in the domain $D = [0, L_y] \times [0, L_x]$ and $t \in [0, T]$, we will study the partial differential equation

$$u_{tt} + bu_t = (q(x, y)u_x)_x + (q(x, y)u_y)_y + f(x, y, t), \quad (1)$$

where $(x, y) \in D$ and $t \in [0, T]$, with the boundary and initial conditions

$$\begin{aligned} \frac{\partial u}{\partial n} &= 0 & (x, y) \in \partial D, t \in [0, T], \\ u(x, y, 0) &= I(x, y) & x, y \in \partial D, \\ u_t(x, y, 0) &= V(x, y) & x, y \in \partial D. \end{aligned} \quad (2)$$

Here ∂D denotes the boundary of the domain D and $\partial/\partial n$ stands for differentiation in the normal direction out of the boundary.

3 Numerical scheme

3.1 Numerical scheme for the inner points

In operator notation, we want to use the following scheme for inner points:

$$[D_t D_t u + b D_{2t} u = D_x(q D_x u) + D_y(q D_y u) + f]_{ij}^n \quad (3)$$

Calculating each summand we find:

$$[D_t D_t u]_{ij}^n = \frac{u_{i,j}^{n+1} - 2u_{i,j}^n + u_{i,j}^{n-1}}{\Delta t^2}, \quad (4)$$

$$[b D_{2t} u]_{ij}^n = b \frac{u_{i,j}^{n+1} - u_{i,j}^{n-1}}{2\Delta t}, \quad (5)$$

$$[D_x q D_x u]_{ij}^n = \frac{q_{i+\frac{1}{2},j}(u_{i+1,j}^n - u_{i,j}^n) - q_{i-\frac{1}{2},j}(u_{i,j}^n - u_{i-1,j}^n)}{\Delta x^2}, \quad (6)$$

$$[D_y q D_y u]_{ij}^n = \frac{q_{i,j+\frac{1}{2}}(u_{i,j+1}^n - u_{i,j}^n) - q_{i,j-\frac{1}{2}}(u_{i,j}^n - u_{i,j-1}^n)}{\Delta x^2}, \quad (7)$$

$$[f]_{ij}^n = f(x_i, y_j, t_n). \quad (8)$$

Inserting these expressions into Eq. (3) and solving for $u_{i,j}^{n+1}$, we get the following scheme for inner points:

$$\begin{aligned}
u_{i,j}^{n+1} = & \{2u_{i,j}^n + u_{i,j}^{n-1} \\
& + \frac{\Delta t^2}{\Delta x^2} (q_{i+\frac{1}{2},j}(u_{i+1,j}^n - u_{i,j}^n) - q_{i-\frac{1}{2},j}(u_{i,j}^n - u_{i-1,j}^n)) \\
& + \frac{\Delta t^2}{\Delta y^2} (q_{i,j+\frac{1}{2}}(u_{i,j+1}^n - u_{i,j}^n) - q_{i,j-\frac{1}{2}}(u_{i,j}^n - u_{i,j-1}^n)) \\
& + \Delta t^2 f(x_i, y_j, t_n)\} / \left(1 + \frac{1}{2}b\Delta t\right)
\end{aligned} \tag{9}$$

3.2 Numerical scheme for the first time step

For the first time step $n = 0$, $u_{i,j}^{n-1}$ is not a part of the grid. To circumvent this problem, we have to apply the initial condition $u_t(x, y, 0) = V(x, y)$ for $x, y \in \partial D$. We discretize it and get:

$$[D_{2t}u = V]_{i,j}^0 \implies \frac{u_{i,j}^1 - u_{i,j}^{-1}}{2\Delta t} = V(x_i, y_j) \implies u_{i,j}^{-1} = u_{i,j}^1 - 2\Delta t V(x_i, y_j).$$

Inserting this into equation (4) and (5), we get:

$$[D_t D_t u]_{i,j}^n = 2 \left(\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t^2} - \frac{V(x_i, y_i)}{\Delta t} \right), \tag{10}$$

$$[b D_{2t} u]_{i,j}^n = b \frac{V(x_i, y_i)}{\Delta t}. \tag{11}$$

Inserting these expressions into Eq. (3) and solving for $u_{i,j}^{n+1}$, we get the following scheme for inner points in the first time step $n = 0$:

$$\begin{aligned}
u_{i,j}^{n+1} = & u_{i,j}^n + \left(1 - \frac{1}{2}b\Delta t\right) \Delta t V_{i,j} \\
& + \frac{\Delta t^2}{2\Delta x^2} (q_{i+\frac{1}{2},j}(u_{i+1,j}^n - u_{i,j}^n) - q_{i-\frac{1}{2},j}(u_{i,j}^n - u_{i-1,j}^n)) \\
& + \frac{\Delta t^2}{2\Delta y^2} (q_{i,j+\frac{1}{2}}(u_{i,j+1}^n - u_{i,j}^n) - q_{i,j-\frac{1}{2}}(u_{i,j}^n - u_{i,j-1}^n)) \\
& + \frac{\Delta t^2}{2} f(x_i, y_j, t_n).
\end{aligned} \tag{12}$$

3.3 Numerical scheme for the boundary

On the boundary, this scheme needs points outside the defined space domain. For example, we need values for $u_{-1,j}^n$, u_{i,N_y+1}^n , and $u_{0,-1}^n$. Values for these points can be obtained from the discretized versions of the Neumann condition.

The Neumann conditions are in discretized form

$$[D_{2x}u]_{0j}^n = [D_{2x}u]_{N_x j}^n = 0 \quad \text{for } j = 0, 1, \dots, N_y, \quad n = 0, 1, \dots, N_t, \quad (13)$$

$$[D_{2y}u]_{i0}^n = [D_{2y}u]_{iN_y}^n = 0 \quad \text{for } i = 0, 1, \dots, N_y, \quad n = 0, 1, \dots, N_t, \quad (14)$$

which leads to the equations

$$u_{1,j}^n = u_{-1,j}^n, \quad u_{N_x-1,j}^n = u_{N_x+1,j}^n \quad \text{for } j = 0, 1, \dots, N_y, \quad n = 0, 1, \dots, N_t \quad (15)$$

$$u_{i,1}^n = u_{i,-1}^n, \quad u_{i,N_y+1}^n = u_{i,N_x+1}^n \quad \text{for } i = 0, 1, \dots, N_y, \quad n = 0, 1, \dots, N_t. \quad (16)$$

A scheme for the borders can now be found by applying each equality to the inner point scheme for its boundary. The corners are found by applying two equalities, one for each of the adjacent border.

3.4 Approximating $q(x,y)$ outside the grid

In our implementation, we assume that values for the function $q(x,y)$ are given only at the grid points (x_i, y_j) . In the finite difference algorithm, we use mean values to evaluate the function at other points. To approximate the q function when evaluated outside the grid, we will apply the arithmetic and harmonic mean, defined respectively as

$$q_{i+\frac{1}{2},j} = \frac{q_{i,j} + q_{i+1,j}}{2} \quad (17)$$

$$(18)$$

and

$$q_{i+\frac{1}{2},j} = 2 \left(\frac{1}{q_{i,j}} + \frac{1}{q_{i+1,j}} \right)^{-1}. \quad (19)$$

When not states explicitly otherwise, we have used the arithmetic mean as the default option.

4 Implementation

We implemented the code in pure python. See the `wave2D_du0.py` for the full code.

TODO: Show the core of the program in minted enviroment

5 Numerical experiments

5.1 Verification

The implementation of the finite difference algorithm was verified with the following tests:

- In the first test the initial conditions were set to

$$\begin{aligned} u(x, y, t = 0) &\equiv I(x, y) = u_0, \\ u_t(x, y, t = 0) &\equiv V(x, y) = 0, \end{aligned} \quad (20)$$

where u_0 is a constant, and we used the restriction $f(x, y, t) = 0$. The resulting PDE gives the constant solution $u(x, y, t) = u_0$.

- In the second test case, the 2-dimensional code was tested with the simple one-dimensional plug function

$$I(x, y) = \begin{cases} 0 & \text{if } |x - L/2| > a, \\ 1 & \text{else,} \end{cases}$$

as initial condition. In the case with $V(x, y) = 0$, $f(x, y, t) = 0$, and $q(x, y) = c^2$, where c is a constant, the solution $u(x, y, t)$ gives exactly two moving squares, as long as the Carnot number $C \equiv c\Delta t/\Delta x$ is 1. This test was repeated by interchanging x and y in the initial condition.

- As another one-dimensional test, we used the one-dimensional initial conditions

$$\begin{aligned} I(x, y) &= \exp(a(x - L/2)^2), \\ V(x, y) &= 0, \end{aligned} \quad (21)$$

where a is a constant, together with the restrictions $b = 0$ and $f(x, y, t) = 0$. These conditions give the solution

$$\begin{aligned} u(x, y, t) &= \frac{1}{2} \exp(-a(x - ct - L_x/2)^2) \\ &= +\frac{1}{2} \exp(-a(x + ct - L_x/2)^2), \end{aligned} \quad (22)$$

at the limit when the boundaries are infinitely far away. This solution does not fulfill the boundary condition $\partial u/\partial n = 0$, but it was a useful test for the first few steps of the simulation.

- As suggested in the project description, we also used the manufactured solution

$$u(x, y, t) = \exp(-bt) \cos(\omega t) \cos\left(\frac{m_x x \pi}{L_x}\right) \cos\left(\frac{m_y y \pi}{L_y}\right), \quad (23)$$

where b is the damping parameter in the studied PDE, ω is a real constant, and m_x and m_y are integers. The solution $u(x, y, t)$ is a standing wave with the desired boundary condition $\partial u/\partial n = 0$. This manufactured solution was tested both with constant q and with the choice $q(x, y) = \exp(-x - y)$. For both cases we had to determine functions $f(x, y, t)$ such that the given manufactured solution fulfills the two-dimensional wave

Table 1: Convergence rates r_k for different convergence parameters h_k and two different choices of the function $q(x, y)$.

h_k	$q = \text{const.}$	$q(x, y) = \exp(-x - y)$
0.01	1.959	1.984
0.001	1.996	1.998

equation. According to the project formulation, the error ε of the solution $u(x, y, t)$ should converge as

$$\varepsilon = Dh^2, \quad (24)$$

where

$$D = D_t F_t^2 + D_x F_x^2 + D_y F_y^2, \quad (25)$$

$\Delta t = F_t h$, $\Delta x = F_x h$, $\Delta y = F_y h$, and D_i and F_i , $i \in \{t, x, y\}$, are constants.

In our implementation, the convergence rate r was estimated by assuming the relation between rates of subsequent errors ε_k with decreasing parameter h_k and rates of the convergence parameter h_k being of the form

$$\frac{\varepsilon_{h_{k-1}}}{\varepsilon_{h_k}} = \left(\frac{h_{k-1}}{h_k} \right)^r. \quad (26)$$

Here the considered error was chosen to be

$$\varepsilon_{h_k} = \max_{i,j} |u_e(x_i, y_j, t_N) - u_{h_k i,j}^N|, \quad (27)$$

where $u_e(x, y, t)$ is the exact solution and $u_{h_k i,j}^N$ is the numerical solution with convergence parameter h_k . Examples of calculated convergence rates are given in Table 5.1.

5.2 One-dimensional simulations

We did some simulations with one-dimensional systems to get a better view of the behaviour of the solutions. With the one-dimensional systems, it was also possible to use finer mesh grids. In the directories `movie_1dwave_box_Ba250` and `movie_1dwave_gaussian_Bs250` we have examples of simulations of one-dimensional systems with a box and a Gaussian shaped hill, respectively. With the same height of the hill, the simulation with a Gaussian perturbation gave a smoother function $u(x, y, t)$ than what was the case with the box-shaped perturbation.

Even if the solution of the Gaussian case is partially sharp, the numerical result becomes quite smooth when adding more space grid points. In the directories `movie_1dwave_gaussian_Bs275_Nx100` and

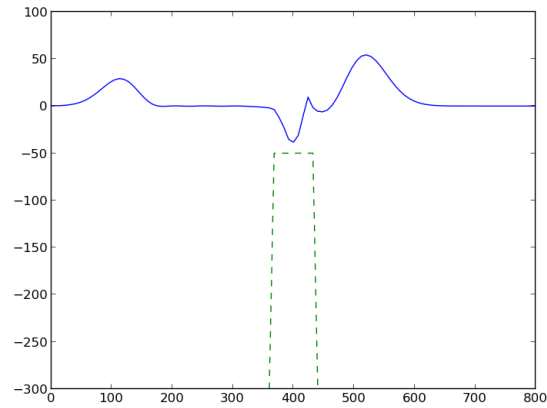


Figure 1: A snapshot of a one-dimensional simulation with a box-shaped hill at the bottom of the sea. The solution is clearly not smooth, and has probably numerical noise.

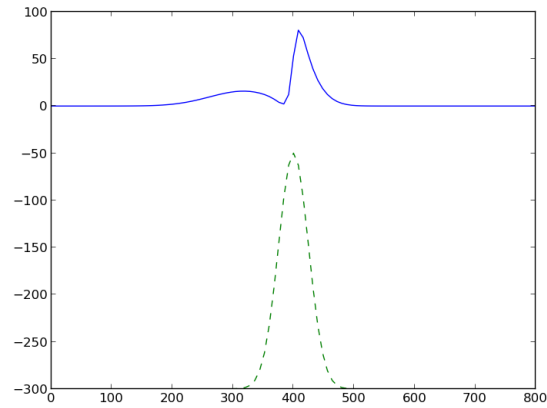


Figure 2: A picture from a one-dimensional simulation with a Gaussian-shaped hill at the seabed. This solution is smoother than the result of the simulation with a box-shaped hill, but is partially quite sharp.

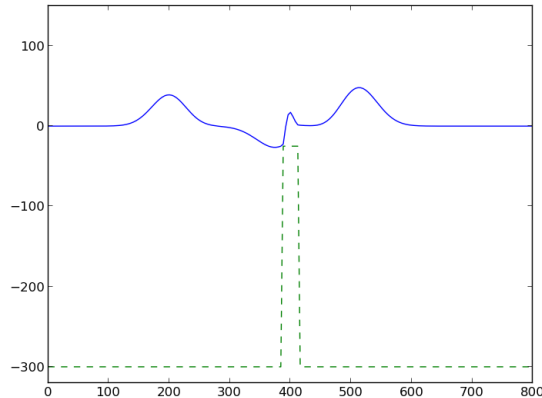


Figure 3: An example from a simulation with a narrow box hill. The resulting function is more spectacular than what was obtained with a wider box.

`movie_1dwave_gaussian_Bs275_Nx200`, there are simulation results obtained with the same Gaussian perturbation, whence the number of space grid points was 100 and 200 in the two runs, respectively. From these movies one clearly sees that the numerical solution becomes smooth when using a finer space mesh.

We also did a comparison between two simulations with the same box perturbation and 100 versus 200 mesh grid points. As in the case with a Gaussian hill, the numerical solution improved with a finer mesh. However, it seems that the system with a box hill may have a solution that is not smooth for any number of space mesh points. It is difficult to say, based on these simulations, what is numerical noise and what is due to a mathematically oscillating or non-smooth function. A first step in investigating that could be to compare numbers for $u_{i,j}^n$ of simulations with different numbers of grid points, using large N_x . It would perhaps also be instructive to look for exact numerical solutions for simple cases with a discontinuous q function.

Another question we asked was how the width of the box hill affects the quality of the numerical solution. A snap-shot from a simulation is shown in Fig. 10, and the movie is stored in the directory `movie_1dwave_box_Ba275_Bs15_Nx200`. At least, the solution is more spectacular with a narrower box perturbation. Still, it is not possible to say whether this is due to numerical noise or an oscillating shape of the exact solution. Intuitively, it is plausible that a geometry with sharp forms gives rise to high-frequency components, which in turn may lead to sharp and/or oscillating solutions. Generally, we observed that there were very clear reflections of part of the wave at the discontinuity points of the q function. When using smoother hills, the reflections became more spread out.

The latter observation was studied closer using discontinuous and smooth

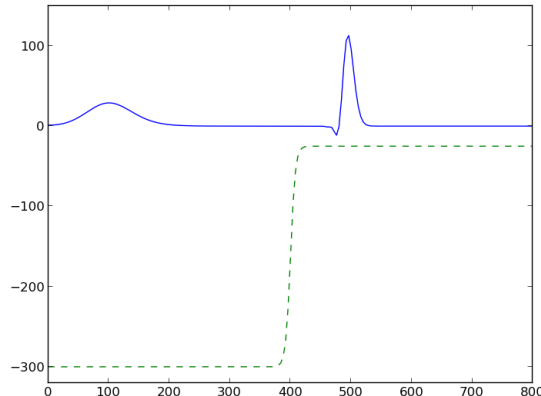


Figure 4: A smooth step function is used at the shore. The reflected wave pulse is more smeared out than in the case with a sharp step function. This result shows oscillations behind the main pulse, which may be numerical noise.

step functions. The smooth step function was of the form

$$B(x, y) = B_0 + \frac{B_a}{1 + \exp(-(x - B_{mx})/B_s)}, \quad (28)$$

where B_0 , B_a , B_{mx} , and B_s are parameters. These geometries might be used to simulate a tsunami approaching a shore where the sea level decreases suddenly. The related movies are stored in the directories `movie_1dwave_box_shore` and `movie_1dwave_smooth_step`. We found that the reflected wave was sharper peaked in the case with a discontinuous step function, whereas the smooth step function gave a reflected wave more smeared out. However, both simulations had similar sharp oscillations behind the main wave. This oscillation may be a numerical artefact. One can also observe how the wave moves slower on the shallower water. This is natural, as the function $q(x, y)$ is proportional to both the depth of the sea, $B(x, y)$, and the squared of the wave velocity, c^2 . From a physical point of view, the wave uses energy to keep the wave pulse higher, and therefore has less energy left for moving horizontally.

A simulation with the harmonic average of the q function is given in the directory `movie_2dwave_box_narrow2_harm` and the same simulation done with arithmetic average in the directory `movie_2dwave_box_narrow2`. The simulation using harmonic average seems to be smoother, but it needs to be investigated if the solution is closer to the exact one.

5.3 One-dimensional hill w/ gap simulations

In the other test we've run so far the hill has only been discontinuous at the edge. What happens if we introduce a discontinuous gap at the top of the hill? Letting

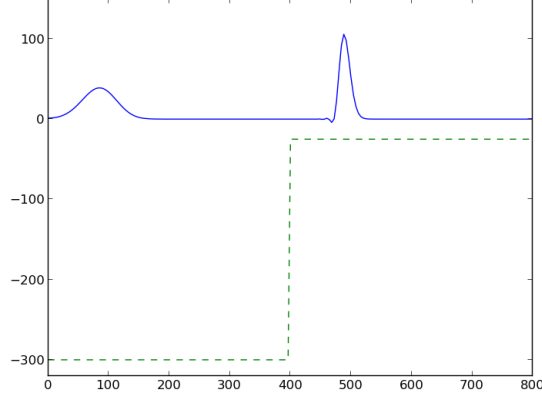


Figure 5: The shore is modelled using a sharp step function. In this case the reflected wave is sharp. As in the case with a smoother step function, here is also seen oscillations behind the main wave pulse.

$b(x,y)$ be the usual shape of the hill, M_x be the x-coordinate of top of the hill, B_m be the height of the seabed and ϵ be the length of the slit we introduce the function:

$$b_s(x, y) = \begin{cases} b(x, y) & \text{if } (M_x - \epsilon) \leq x \leq (M_x + \epsilon) \\ B_m & \text{otherwise} \end{cases}$$

For the choice of b we have several good choices. For our simulation we chose two which seemed radically different: The continuous gaussian bell shape and the discontinuous box shape. They were ideal as we'd already studied their behaviour without a gap.

The difference between M_x and B_m were chosen large to make the discontinuous border of the hill longer and hopefully make an eventual numerical instability more visible. ϵ is chosen to be small compared to the size of the hill. One might imagine a discontinuity at a single point might cause more trouble than one spanning several points, so $\epsilon = 0$ is also tested.

5.4 Plots of hills w/ gap

5.5 One-dimensional hill w/ gap simulation conclusions

For the gaussian hill introducing a gap did not seem to cause much numerical instability. While the wave behaved differently than it did for the gapless hill, it stayed relatively smooth throughout the simulation and seemed to fit somewhat with the physical properties of such a wave. For the box hill the gap caused more problems. While the wave was at times rather steep and the wave propagated into the hill at one point.

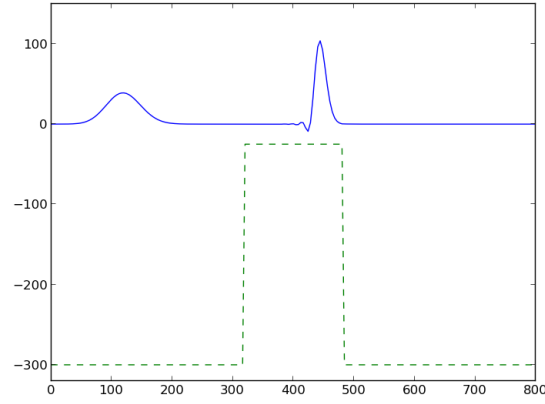


Figure 6: The last frame of the small box with no gap movie.

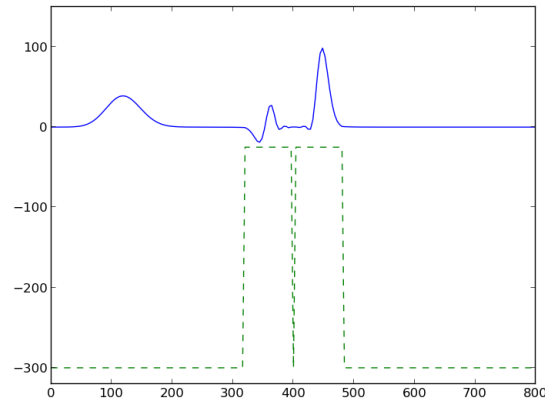


Figure 7: The last frame of the box with 1 point gap movie.

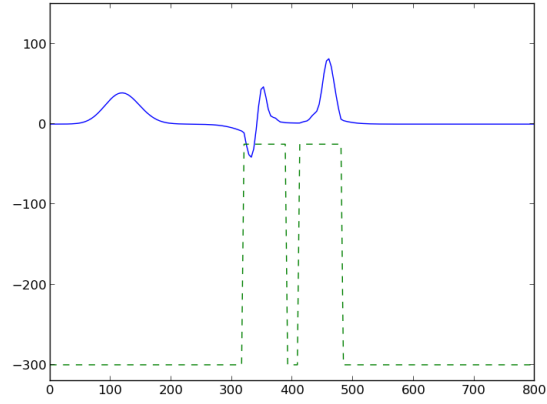


Figure 8: The last frame of the box with 10 point gap slit movie.

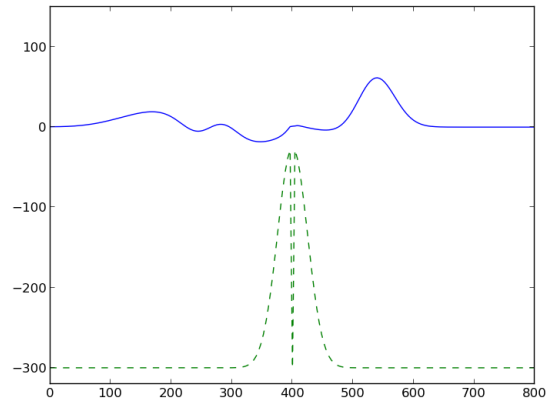


Figure 9: The last frame of the gaussian hill with 1 point gap movie.

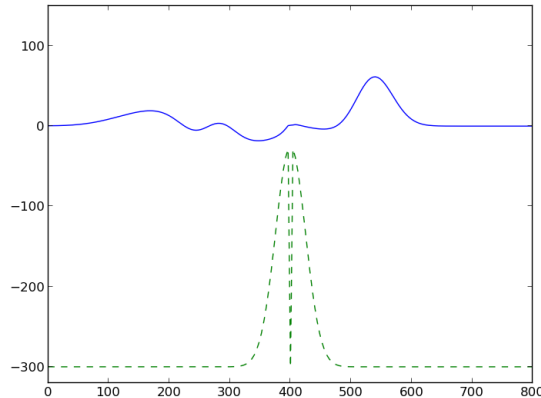


Figure 10: The last frame of the gaussian hill with 1 point gap movie.

One might then be tempted to conclude that having otherwise smooth hills mitigates the effect of a gap in the middle, but this is not necessarily true. The reflected wave from the gap did not in any of the tests seem to reflect back and interact with the main wave while it was in the gap, so the only difference the hill shape should have would be the shape of the wave hitting the gap. It does not seem unreasonable that a wave in the shape of the wave generated by passing over the box hill could also be generated using a smoother hill. If this is the case, would not a gap in this hill cause the same problems as we saw for the box? I find it more likely the steepness of the wave is causing the instability in the box hill case, though this could in turn be connected to the steepness of the hill.

The one point gap simulations mirrored the ten point gap simulations and did not seem to cause any additional problems. If anything both simulations seemed more stable than the ten point counterparts. It is still worth noting the huge impact such a point has on the solution. If the gap is not mirrored in the physical case (say it is a measuring error) one can get huge differences between the simulated solution and the real behaviour of the phenomena.

5.6 Two-dimensional simulations

Our main goal was to study finite difference results of the two-dimensional wave equation. We did simulations with box-, Gaussian-, and cosine hat-shaped hills at the bottom of the sea. In Figs. 11, 12, 13, and 14 we show snap-shots from simulations of the two-dimensional wave equation with four different hill shapes. We observe that the sharper the hill is, the more oscillations are seen around the perturbation region. The narrow box hill produces ring-shaped waves that spread around from the hill after the main wave pulse has passed the hill. We

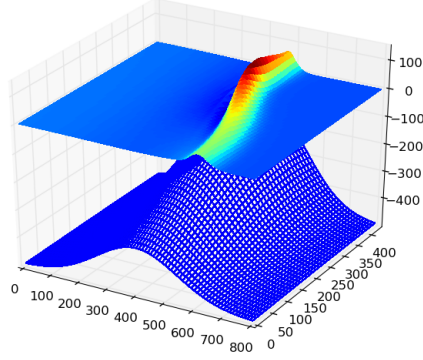


Figure 11: Snap-shot of a simulation of the two-dimensional wave equation with a Gaussian perturbation. The hill is smooth, and so is also the numerical result $u_{i,j}^n$.

were not able on the basis of these simulations to judge if these oscillations are due to numerical artefacts or the exact mathematical solution. However, considering that we needed more grid points in the one-dimensional case to get smooth functions, that would probably also be the case for the two-dimensional simulations. In the two-dimensional simulations we used only up to 80 grid points in the space coordinates. That number is probably too small. The two-dimensional simulations were quite slow to do, but we found that the finite difference algorithm consumed only a small fraction of the time. Most of the time was used to plot the results and store the figures. The simulations would therefore not have been much faster with compiled and parallelized loops. There would probably be room for optimizations in creating the pictures, for example by storing fewer of them, but that must be investigated in a later work.

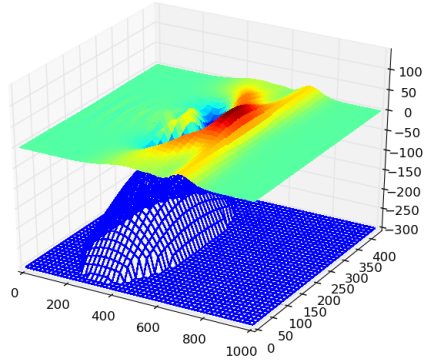


Figure 12: This figure is from a simulation of the two-dimensional wave equation using a cosine hat -shaped hill at the bottom of the sea. In this case the hill is steeper than in the case with the Gaussian perturbation. The resulting wave is clearly less smooth, and there are oscillations in the region of the perturbation after the main pulse has passed the hill.

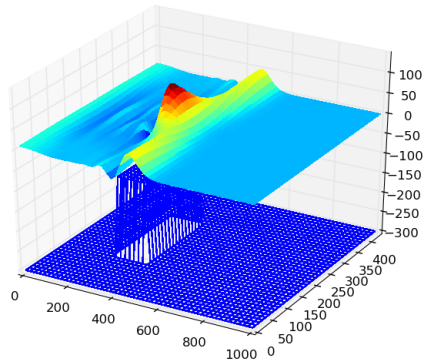


Figure 13: A picture from a two-dimensional wave equation simulation where the perturbation is a box. This system gives also oscillations behind the main pulse.

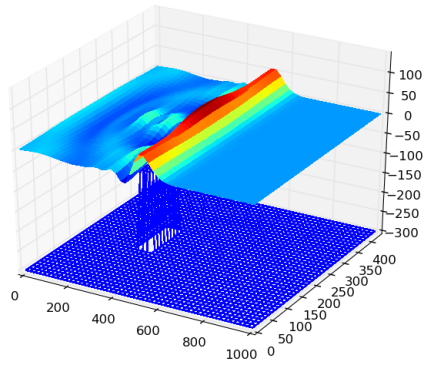


Figure 14: A snap-shot from a simulation of a two-dimensional wave equation with a sharp box-shaped hill. Observe that there is created a wave around the hill after the main wave pulse has passed the perturbation.

6 Conclusions