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

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

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

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

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

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:

$$[D_t D_t u + bD_{2t}u = D_x(qD_x u) + D_y(qD_y u) + f]^n_{i\,j} \tag{3}$$

Calculating each summand we find:

$$[D_t D_t u]^n_{i\,j} = \frac{u^{n+1}_{i,j} - 2u^n_{i,j} + u^{n-1}_{i,j}}{\Delta t^2}, \tag{4}$$

$$[bD_2 tu]^n_{i\,j} = b\frac{u^{n+1}_{i,j} - u^{n-1}_{i,j}}{2\Delta t}, \tag{5}$$

$$[D_x q D_x u]^n_{i\,j} = \frac{q_{i+\frac{1}{2},j}(u^n_{i+1,j} - u^n_{i,j}) - q_{i-\frac{1}{2},j}(u^n_{i,j} - u^n_{i-1,j})}{\Delta x^2}, \tag{6}$$

$$[D_y q D_y u]^n_{i\,j} = \frac{q_{i,j+\frac{1}{2}}(u^n_{i,j+1} - u^n_{i,j}) - q_{i,j-\frac{1}{2}}(u^n_{i,j} - u^n_{i,j-1})}{\Delta x^2}, \tag{7}$$

$$[f]^n_{i\,j} = f(x_i, y_j, t_n). \tag{8}$$

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

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

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 intial 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:

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

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

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 $n = 0$:

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

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

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

$$[D_{2y}u]_{i\,0}^{n} = [D_{2y}u]_{0,N_y}^{n} = 0 \quad \text{for } i = 0, 1, \ldots, N_y, \quad n = 0, 1, \ldots, N_t, \tag{14}$$

which leads to the equations

$$u_{1,j}^{n} = u_{-1,j}^{n}, \quad u_{N_x-1,j}^{n} = u_{Nx+1,j}^{n} \quad \text{for } j = 0, 1, \ldots, N_y, \quad n = 0, 1, \ldots, N_t \tag{15}$$

$$u_{i,1}^{n} = u_{i,-1}^{n}, \quad u_{i,Ny+1}^{n} = u_{i,Nx+1}^{n} \quad \text{for } i = 0, 1, \ldots, N_y, \quad n = 0, 1, \ldots, N_t. \tag{16}$$

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 adjecent 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

$$q_{i+\frac{1}{2},j} = \frac{q_{i,j} + q_{i+1,j}}{2} \tag{17}$$

$$\tag{18}$$

and

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

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

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

  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

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

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

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

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

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

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 choise $q(x,y) = \exp(-x-y)$. For both cases we had to determine functions $f(x,y,t)$ such that the given manufactured solution fulfills the two-dimensional wave equation. According to the project formulation, the error ε of the solution $u(x,y,t)$ should converge as

$$\varepsilon = Dh^2, \tag{24}$$

where

$$D = D_t F_t^2 + D_x F_x^2 + D_y F_y^2, \tag{25}$$

$\Delta t = F_t h$, $\Delta x = F_x h$, $\Delta y = F_y h$, 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

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

Table 1: Convergence rates r_k for different convergence parameters h_k and two different choises 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

Here the considered error was chosen to be

$$\varepsilon_{h_k} = \max_{i,j} \left| u_e(x_i, y_j, t_N) - u^N_{h_k i,j} \right|, \tag{27}$$

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

6 Conclusions