Project Report

For

Finite Difference Simulation of 2D waves

Course INF 5620/9620 -Numerical Methods for the Partial Differential Equation

Hao, Zhao

Department of Geoscience & Department of Informatics, University of Oslo

Oct 12, 2015

1. Introduction

This numerical project is to implement the discretized 2D wave equation, and verify the discretization code with constant solution,1-D plug-wave, standing undamped & damped waves, manufactured solution and a physical problem of tsunami simulation.

2. Mathematical Problem

The project addresses the two-dimensional, standard, linear wave equation, with damping,

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

The associated boundary condition is

$$\frac{\partial \mathrm{u}}{\partial \mathrm{n}} = 0 \qquad (2)$$

In a rectangular spatial domain [0,Lx] x [0*Ly], and the corresponding initial condition are:

$$\mathrm{u}(\mathrm{x},\mathrm{y},0) = \mathrm{I}(\mathrm{x},\mathrm{y}) \text{ and } u_t(\mathrm{x},\mathrm{y},0) = \mathrm{V}(\mathrm{x},\mathrm{y})$$

2.1 Discretization

Based on the finite difference method, the 2D linear wave equation can be discretized to

$$\begin{aligned}\frac{u_{i,j}^{n+1} - 2*u_{i,j}^{n} + u_{i,j}^{n-1}}{\Delta t^2} + b*\frac{u_{i,j}^{n+1} - u_{i,j}^{n-1}}{2*\Delta t} \\ = \frac{1}{\Delta x^2}\left(\frac{1}{2}(q_{i,j}+q_{i+1,j})(u_{i,j}^{n} - u_{i+1,j}^{n}) - \frac{1}{2}(q_{i,j}+q_{i-1,j})(u_{i,j}^{n} - u_{i-1,j}^{n})\right) \\ + \frac{1}{\Delta y^2}\left(\frac{1}{2}(q_{i,j}+q_{i,j+1})(u_{i,j}^{n} - u_{i,j+1}^{n}) - \frac{1}{2}(q_{i,j}+q_{i,j-1})(u_{i,j}^{n} - u_{i,j-1}^{n})\right) \\ + f(x,y,t) \qquad (4)\end{aligned}$$

By solving this equation, it yields to

$$u_{i,j}^{n+1} = (\frac{b}{2} * \Delta t + 1)^{-1} * \left(\left(\frac{b}{2} * \Delta t - 1 \right) * u_{i,j}^{n-1} + 2 * u_{i,j}^{n} + f * \Delta t^2 + \frac{\Delta t^2}{\Delta x^2} * A + \frac{\Delta t^2}{\Delta y^2} * B \right) \quad (5)$$

Where A and B are listed below

$$A = \frac{1}{2}(q_{i,j} + q_{i+1,j})(u_{i,j}^{n} - u_{i+1,j}^{n}) - \frac{1}{2}(q_{i,j} + q_{i-1,j})(u_{i,j}^{n} - u_{i-1,j}^{n})$$
$$B = \frac{1}{2}(q_{i,j} + q_{i,j+1})(u_{i,j}^{n} - u_{i,j+1}^{n}) - \frac{1}{2}(q_{i,j} + q_{i,j-1})(u_{i,j}^{n} - u_{i,j-1}^{n}) \quad (6)$$

To solve the first time step, the special form is derived from equation (5), by using the initial condition:

$$u_t(\mathrm{x}, \mathrm{y}, 0) = \mathrm{V}(\mathrm{x}, \mathrm{y}) = \frac{u_{i,j}^{1} - u_{i,j}^{-1}}{\Delta t^2} \quad (7)$$

And it yields to

$$u_{i,j}^{1} = \left(1 - \frac{b}{2} * \Delta t\right) * \Delta t * V + u_{i,j}^{0} + \frac{1}{2}\left(\frac{\Delta t^2}{\Delta x^2} * A + \frac{\Delta t^2}{\Delta y^2} * B\right) + \frac{1}{2} * f * \Delta t^2 \quad (8)$$

Where A and B are listed below

$$A = \frac{1}{2}(q_{i,j} + q_{i+1,j})(u_{i,j}^{0} - u_{i+1,j}^{0}) - \frac{1}{2}(q_{i,j} + q_{i-1,j})(u_{i,j}^{0} - u_{i-1,j}^{0})$$
$$B = \frac{1}{2}(q_{i,j} + q_{i,j+1})(u_{i,j} - u_{i,j+1}^{0}) - \frac{1}{2}(q_{i,j} + q_{i,j-1})(u_{i,j}^{0} - u_{i,j-1}^{0}) \quad (9)$$

The equation (5) and (8) are the scheme for calculation the inner points of the 2D discretized wave equation, for the 4 boundaries of the 2D rectangular space, the scheme is derived with the help of Neumann boundary condition

$$\frac{\partial \mathrm{u}}{\partial \mathrm{n}} = 0$$

which yields to

$u_{i-1,j}^{n} = u_{i+1,j}^{n}$ at i = 0

$u_{i+1,j}^{n} = u_{i-1,j}^{n}$ at i = Lx

$u_{i,j-1}^{n} = u_{i,j+1}^{n}$ at j = 0

$u_{i,j+1}^{n} = u_{i,j-1}^{n}$ at j = Ly

Based on the above derivation, the solver for the 2D linear wave equation is implemented. The corresponding scalar version and vectorized version are generated.

2.2 Implementation of Discretized 2D linear wave equation

Here I briefly describe the implementation of the discretized 2D linear wave equation with the example of the scalar solver version. The vectorized version follows the same concept but used the operations on slices of arrays to save the computation needs which are well commented and self-explained in the program.

For the scalar version of the discretized 2D linear wave equation, the below calculations are carried out:

(1) Define the 5 coefficient variables for the discretized equation, the time step 1 and the subsequent time steps are assigned the different coefficient variable sets:

If step1:

D1 = 1; D2 = 0; D3 = 0.5; D4 = 0.5; D5 = 1-0.5*b*dt;

else:

D1 = 2/(1+0.5*b*dt); D2 = (1-0.5*b*dt)/(1+0.5*b*dt); D3 = 1/(1+0.5*b*dt); D4 = 1/(1+0.5*b*dt); D5 = 0;

(2) Start the looping of every location and make the calculation of the wave field. To handle the Neumann Boundary condition, the below indexing solution is implemented:

ip1 = i+1;im1 = i-1;jp1 = j+1;jm1 = j-1

ip1 = im1 if ip1 > Ix[-1] else ip1; im1 = ip1 if im1 < Ix[0] else im1

jp1 = jm1 if jp1 > Iy[-1] else jp1; jm1 = jp1 if jm1 < Iy[0] else jm1

(3) Based on the above indexing solution, the u_xx and u_yy, which is the expression A,B in the previous equation (6) and (9), and the wavefiled from time step 1 to subsequent steps are calculated as following expression:

u_xx = 0.5*(q[i,j] + q[ip1,j])*(u_1[ip1,j] - u_1[i,j]) - 0.5*(q[i,j] + q[im1,j])*(u_1[i,j] - u_1[im1,j])
u_yy = 0.5*(q[i,j] + q[i,jp1])*(u_1[i,jp1] - u_1[i,j]) - 0.5*(q[i,j] + q[i,jm1])*(u_1[i,j] - u_1[i,jm1])

u[i,j] = D1*u_1[i,j] - D2*u_2[i,j] + D3*(Cx2*u_xx + Cy2*u_yy) + D4*(dt2*f(x[i], y[j], t[n])) + D5*dt*V(x[i], y[j])

3. Verification

The following contents presented the verification of the solver of the of Discretized 2D linear wave equation.

3.1 Verification of constant solution

With the constant solution u(x,y,t) = constant. The initial condition can be derived as I(x,y,0) = constant, V(x,y) = 0, Ut =0, Ux =0, Uy = 0,and f(x,y,t)=0, thus the constant solution fulfills both the exact 2D wave equation and discrete 2-D wave equation. In the test, the arbitrary constant value 10 is used for the constant solution, and random chosen c = 1.0 for numerical test. This calculation is verified with the nose test, and we haven't observed any error between the exact solution and discrete solution.

3.2 Exact 1D Plug-wave solution in 2D Verification

To simulate the 1D plug-wave in the 2D case, the V(x,y) = 0, f(x,y,t)=0, the velocity field c(x,y)= 1.0, and the following I(x,y) is applied to simulate the plug-wave traveling in x direction (the time step is chosen to make the corresponding courant number to be unit):

$$I(x,y) = \begin{cases} 0, & \left|x - \frac{Lx}{2}\right| > 0.1 \\ 1, & \left|x - \frac{Lx}{2}\right| < 0.1 \end{cases}$$

Similarly, to simulate the 1D plug-wave propagating to Y direction in the 2D case, the V(x,y) = 0, f(x,y,t)=0, the velocity field c(x,y)= 1.0, and the following I(x,y) is applied: (the time step is chosen to make the corresponding courant number to be unit):

$$I(x,y) = \begin{cases} 0, & \left|y - \frac{Ly}{2}\right| > 0.1 \\ 1, & \left|y - \frac{Ly}{2}\right| < 0.1 \end{cases}$$

The picture1 shows the simulated Plug-wave in the 2D which propagates from both X and Y direction.

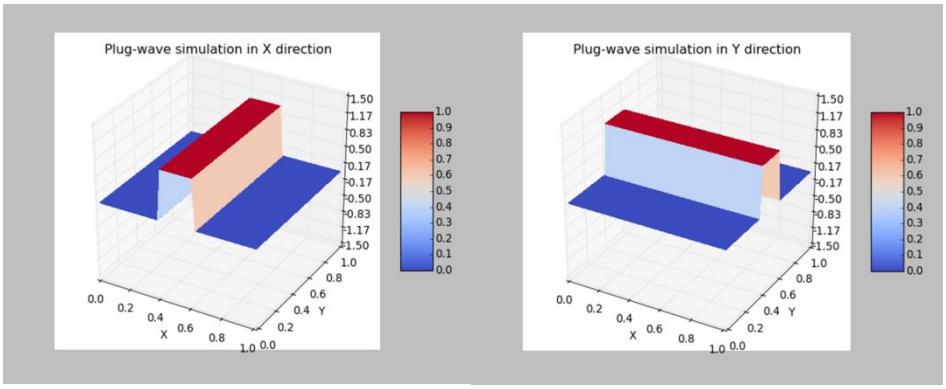

Figure1. Plug-wave simulation in 2D,

Wave Propagating from X direction (Left), Propagating from Y Direction (Right)

3.3 Simulation of Standing and Undamped Waves

The standing and undamped 2D wave is simulated with the exact standing wave solution (10) and its corresponding discrete equation.

$$u^e(x,y,t) = ACos(K_x x)Cos(K_y y)Cos(Wt)\ , \tag{10}$$

$$k_x = \frac{Mx * Pi}{Lx}\ , k_y = \frac{My * Pi}{Ly}$$

In the simulation, we used the no damping factor (b=0), constant wave velocity(c=1.0), and the frequency (W = Sqrt(Kx^2 +Ky^2)*C) is used to make the numerical solution admits the exact standing wave solution. Also the corresponding initial condition is derived by sympy and used in the simulation. The fiugre2 shows a snapshot of undamped standing wave simulation.

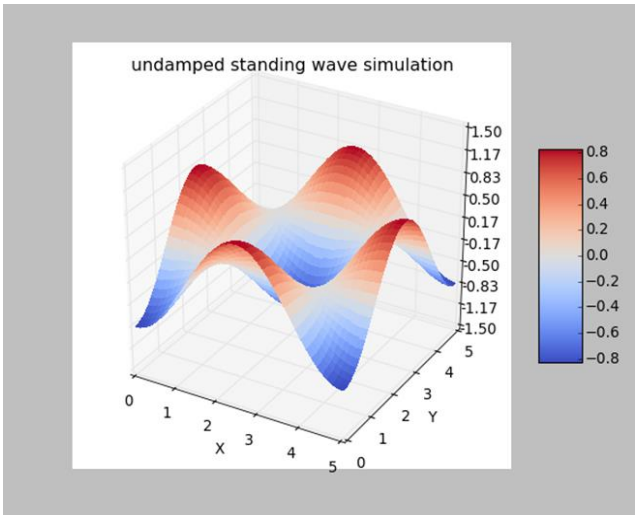

Figure2. Standing and Undamped Wave simulation

To further verify the numerical solution is converging to the exact solution, the error is calculated based on the infinite Norm in the program. And the corresponding convergence test has been done with the following designed variable dx, dy and dt parameters. The numerical error is derived correspondingly, and the calculated convergence rate is 2.00 which agree to the expected.

	dx	dy	dt	Error
Plan-1	0.5	0.5	0.08	0.02848
Plan-2	0.25	0.25	0.04	0.00710
Plan-3	0.125	0.125	0.02	0.00177

Table-1. Convergence Test plans and corresponding Numerical Error

3.4 Manufactured solution Verification

In order to test the solution of the 2D wave equation problem with damping and variable wave velocity. The damping standing waves with exact solution (11) is used in this test.

$$u^e(x,y,t) = \big(ACos(Wt) + BSin(Wt)\big)e^{-ct}Cos(K_x x) * Cos\big(K_y y\big)\,, \tag{11}$$

$$k_x = \frac{Mx * Pi}{Lx}\,, k_y = \frac{My * Pi}{Ly}$$

To make an accurate numerical solution of the exact damping standing waves and variable wave velocity (the velocity function: C=X, is used as the variable wave velocity function in this test), the initial condition I(x,y), V(x,y) and source term F(x,y,t) are derived by Sympy within the jobs and used in the simulation. The fiugre3 shows a snapshot of damped standing wave simulation, and the corresponding convergence test (table-2) gives the reasonable convergence rate: 2.02.

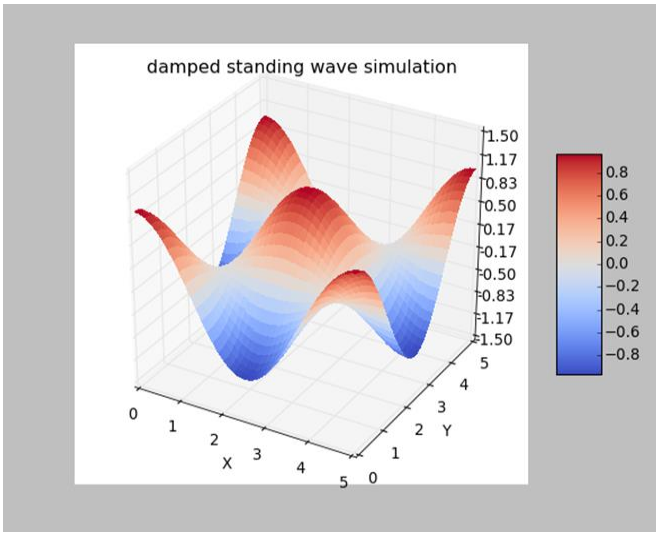

Figure3. Standing and damped Wave simulation

	dx	dy	dt	Error
Plan-1	0.5	0.5	0.04	0.10269
Plan-2	0.25	0.25	0.02	0.02601
Plan-3	0.125	0.125	0.01	0.00640

Table-2. Convergence Test plans and corresponding Numerical Error

3.5 Investigate a Physical Problem

In the last test, I simulated a tsunami wave passing through a shallow water area and the corresponding ocean surface elevation variation based on the discrete 2D wave equation schemes. To simulate the Tsunami wave propagation, the smooth Gaussian function (12) is used to simulate the wave propagating along the X direction. (12)

$$I(x, y, t = 0) = 5 * e^{-\left(\frac{x^2}{10}\right)}$$

Meanwhile, to simulate the subsurface hill, three different bottom shapes are modelled by 2D Gaussian function (13), Cosine hat function (14) and a simple Box function (15).

$$B(x, y) = B_0 + B_a e^{-\left(\frac{x-Bmx}{B_s}\right)^2 - \left(\frac{y-Bmy}{b*B_s}\right)^2} \quad (13)$$

$$B(x, y) = B_0 + B_a Cos\left(\pi \frac{x - Bmx}{2B_s}\right) Cos\left(\pi \frac{y - Bmy}{2B_s}\right) \quad (14)$$

$$B(x, y) = B_0 + B_a \quad (15)$$

For x and y inside the rectangle Bmx -Bs <= X <= Bmx +Bs, Bmy –b*Bs <= y <= Bmy +b*Bs and B(x,y) = Bo, when x and y outside the rectangle.

In the simulation, the Bo = 0, Ba=5, Bmx = 20, Bmy = 50, Bs = 10 and Bb =1 is used for these subsurface hill numerical modelling. And the corresponding spatial variant wave velocity is defined by the equation (16), where the H0 represent the average water bottom depth, and 10m used in this simulation:

$$c(x, y) = \sqrt{g * (H0 - B(x, y)}$$

Based on all the initial conditions and input, the tsunami wave passing through a shallow water area is simulated. The following pictures 4, 5 and 6 present the simulation snapshots. The results also proves the Gaussian function and cosine hat function has better modelling result than the simple box function since the smooth subsurface hill generated less numerical noise in simulation.

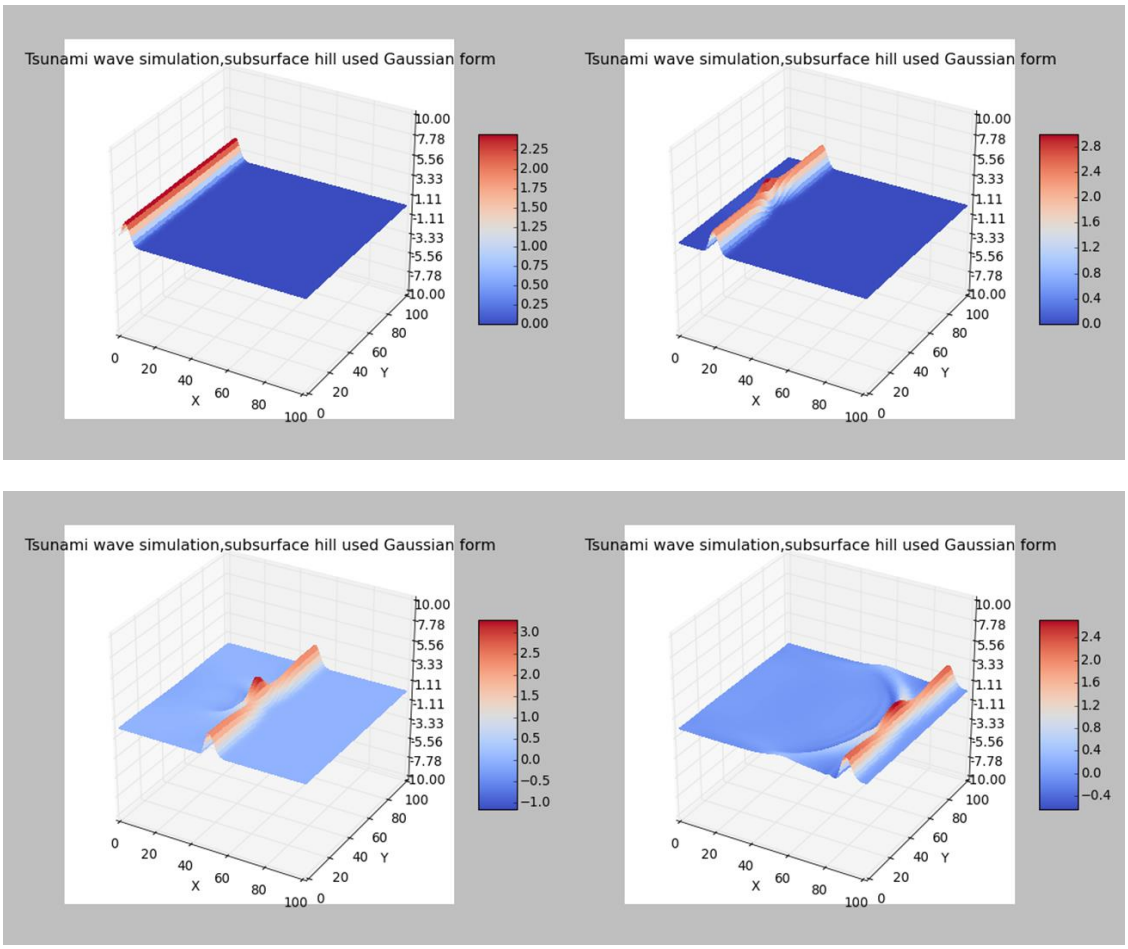

Figure4. Simulation of Tsunami wave passing through the shallow water

The subsurface hill modelled by Gaussian function.

Top-Left: wave before reach the shallow bottom. **Top-right**: wave start touching the shallow bottom.

Bot-Left: wave reaching the half of the X distance. **Bot-right**: wave reaching the whole X distance

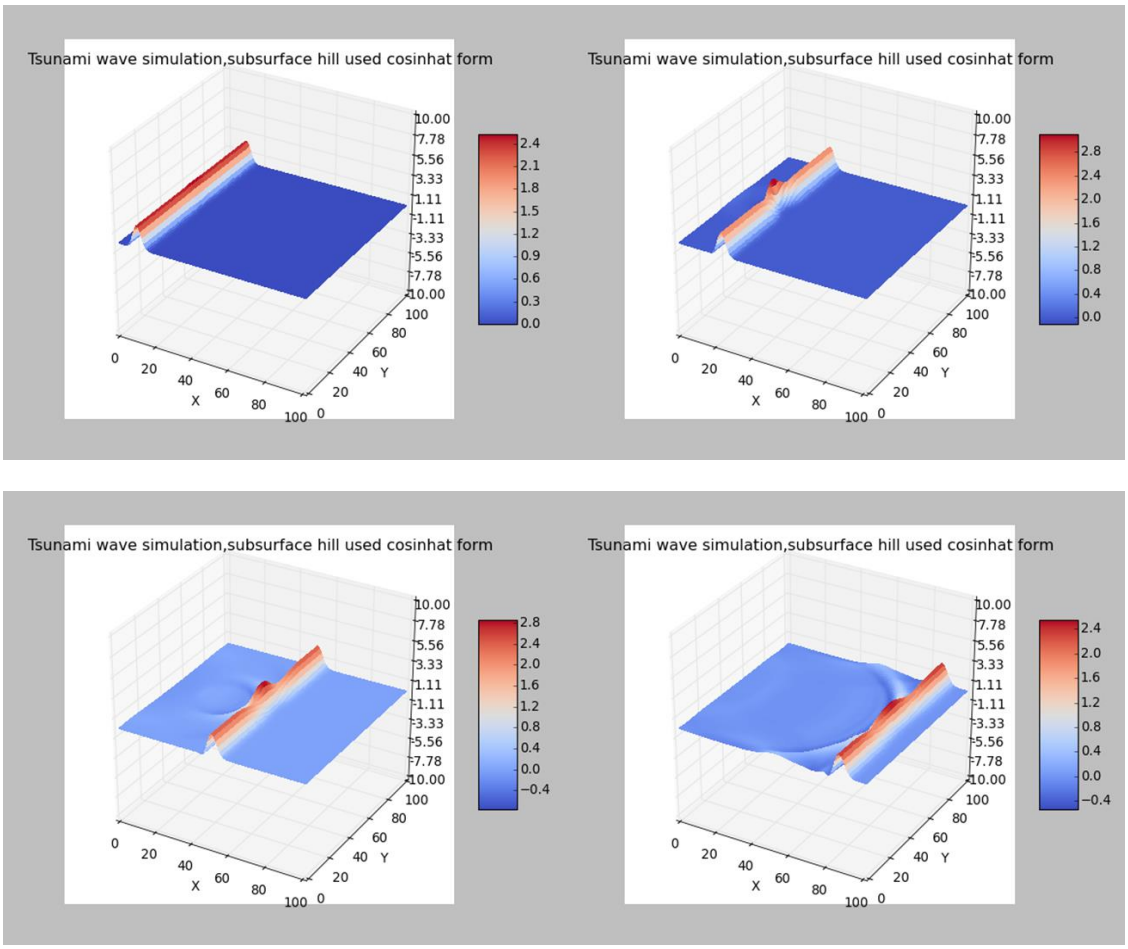

Figure5. Simulation of Tsunami wave passing through the shallow water

The subsurface hill modelled by cosine hat function.

Top-Left: wave before reach the shallow bottom. **Top-right**: wave start touching the shallow bottom.

Bot-Left: wave reaching the half of the X distance. **Bot-right**: wave reaching the whole X distance

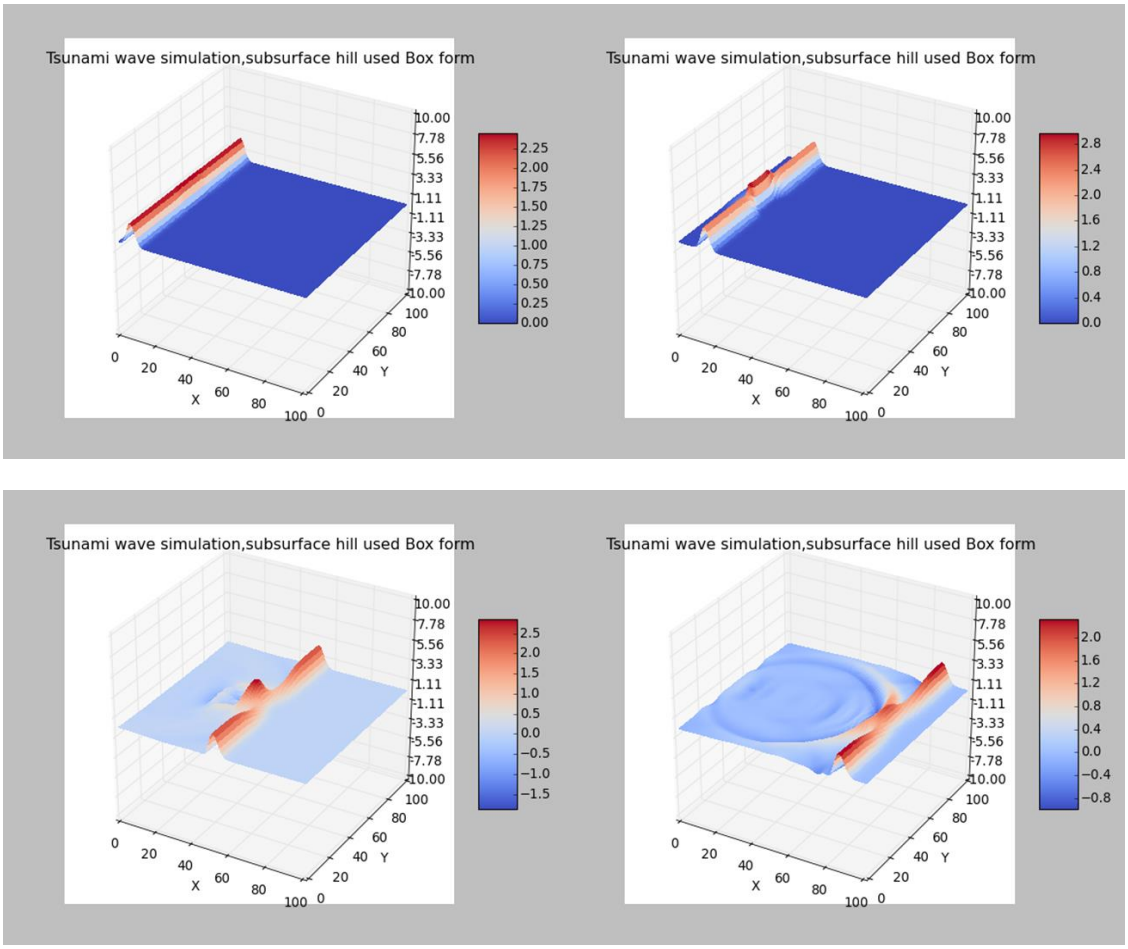

Figure6. Simulation of Tsunami wave passing through the shallow water

The subsurface hill modelled by Box function.

Top-Left: wave before reach the shallow bottom. **Top-right**: wave start touching the shallow bottom.

Bot-Left: wave reaching the half of the X distance. **Bot-right**: wave reaching the whole X distance