OE5450 Final Project: Numerical simulation of 2D long waves induced by seismic seafloor uplift

Gunashekhar M
NA21B024

May 13, 2025

Abstract

This report presents a numerical simulation of Long waves (here Tsunami) generation and propagation, comparing the 2004 Indian Ocean and 2011 Tohoku events. The model solves the 2D nonlinear shallow water equations with seismic seafloor deformation. Key parameters affecting wave characteristics are analyzed through parametric studies.

1 Governing Equations

The tsunami wave propagation is modeled using the linearized shallow water equations with bottom friction and moving seabed:

$$\frac{\partial^2 \eta}{\partial t^2} = gh\left(\frac{\partial^2 \eta}{\partial x^2} + \frac{\partial^2 \eta}{\partial y^2}\right) - f_b \frac{\partial \eta}{\partial t} + \frac{\partial^2 b}{\partial t^2} \tag{1}$$

where η is the water surface elevation, h is the water depth, g is gravity, f_b is the bottom friction coefficient, and b is the seabed elevation.

2 Finite Difference Scheme

The equations are discretized using second-order finite differences:

$$\frac{\partial^2 \eta}{\partial t^2} \approx \frac{\eta_{i,j}^{n+1} - 2\eta_{i,j}^{n} + \eta_{i,j}^{n-1}}{\Delta t^2} \tag{2}$$

$$\frac{\partial^2 \eta}{\partial x^2} \approx \frac{\eta_{i+1,j}^{n} - 2\eta_{i,j}^{n} + \eta_{i-1,j}^{n}}{\Delta x^2} \tag{3}$$

$$\frac{\partial^2 \eta}{\partial y^2} \approx \frac{\eta_{i,j+1}^{n} - 2\eta_{i,j}^{n} + \eta_{i,j-1}^{n}}{\Delta y^2} \tag{4}$$

The complete discretized equation becomes:

$$
\begin{aligned}
\eta_{i,j}^{n+1} =& 2\eta_{i,j}^{n} - \eta_{i,j}^{n-1} \\
&+ gh\Delta t^2 \left(\frac{\eta_{i+1,j}^{n} - 2\eta_{i,j}^{n} + \eta_{i-1,j}^{n}}{\Delta x^2} + \frac{\eta_{i,j+1}^{n} - 2\eta_{i,j}^{n} + \eta_{i,j-1}^{n}}{\Delta y^2} \right) \\
&- f_b \Delta t (\eta_{i,j}^{n} - \eta_{i,j}^{n-1}) \\
&+ b_{i,j}^{n}
\end{aligned} \tag{5}
$$

3 Boundary Conditions

Sommerfeld radiation boundary conditions are applied:

$$\eta_{0,j}^{n+1} = \eta_{0,j}^{n} + \alpha(\eta_{1,j}^{n+1} - \eta_{0,j}^{n}) \tag{6}$$

$$\eta_{NX-1,j}^{n+1} = \eta_{NX-1,j}^{n} + \alpha(\eta_{NX-2,j}^{n+1} - \eta_{NX-1,j}^{n}) \tag{7}$$

where $\alpha = c\Delta t/\Delta x$ is the Courant number.

4 Seabed Uplift Model

The seabed uplift is modeled as a Gaussian-shaped deformation centered at the earthquake epicenter, with maximum displacement at the center tapering to zero at the rupture boundary:

$$b(x, y, t) = b_{\max}(t) \cdot \exp\left(-\frac{r^2}{R^2}\right) \tag{8}$$

where:

- $b(x, y, t)$ is the vertical seabed displacement at position (x, y) and time t
- $b_{\max}(t)$ is the time-dependent maximum uplift at the epicenter
- $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$ is the radial distance from the epicenter (x_c, y_c)
- R is the characteristic radius of the deformation zone

Key characteristics:

- Maximum uplift occurs at the epicenter $(r = 0)$
- Deformation decays exponentially with distance from epicenter
- At $r = R$, the uplift reduces to $e^{-1} \approx 37\%$ of maximum
- Effect becomes negligible (<1%) at $r > 2R$

The seabed uplift is modeled as:

$$b_{\max}(t) = \begin{cases} \frac{z_{\text{bmax}} t}{n_d} & t \le n_d \\ z_{\text{bmax}} & t > n_d \end{cases} \tag{9}$$

with spatial distribution:

$$b_{i,j} = (b_{\max}^{n} - 2b_{\max}^{n-1} + b_{\max}^{n-2}) \exp\left(-\frac{r_{i,j}^2}{R^2}\right) \tag{10}$$

5 CFL Condition for 2D Tsunami Simulation

5.1 Wave Speed Calculation

The shallow water wave speed is given by:

$$c = \sqrt{gh} \tag{11}$$

where:

- c is the wave speed (m/s)
- g is gravitational acceleration (9.81 m/s^2)
- h is the water depth (4000 m in the code)

5.2 Courant Number

The Courant number is defined as:

$$\alpha = \frac{c\Delta t}{\Delta x} \tag{12}$$

5.3 Derivation of 2D CFL Condition

The exact stability criterion for 2D problems:

$$c\Delta t\sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}} < 1 \tag{13}$$

For equal grid spacing ($\Delta x = \Delta y$):

$$\frac{c\Delta t}{\Delta x}\sqrt{2} < 1 \implies \alpha < \frac{1}{\sqrt{2}} \tag{14}$$

5.4 Stability Condition

For the 2D finite difference scheme:

$$\alpha = c\frac{\Delta t}{\Delta x} < \frac{1}{\sqrt{2}} \approx 0.707 \tag{15}$$

With the given parameters:

$$\alpha = \frac{\sqrt{9.81 \times 4000} \times 1.0}{1000} \approx 0.198 \quad \text{(Indian ocean case,Stable)} \tag{16}$$

$$\alpha = \frac{\sqrt{9.81 \times 2000} \times 0.5}{500} \approx 0.1400 \quad \text{(Japan Tohoku case, Stable)} \tag{17}$$

Table 1: Key Parameters for 2004 vs 2011 Events

Parameter	2004 Indian Ocean	2011 Tohoku
Water Depth (h_0)	4000 m	2000 m
Max Uplift (z_{bmax})	10 m	5 m
Rupture Duration (n_d)	50 s	30 s
Bed Friction (f_b)	0.002	0.003
Rupture Radius (R)	10 km	30 km
Domain Size	200×200 km	100×100 km
Time step (DT)	1 s	0.5 s
Spatial step ($DX = DY$)	1000 m	500 m

6 Parameter Comparison

Long wave Simulation Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <sys/stat.h>
#include <assert.h>

#ifdef _WIN32
#include <direct.h>
#define mkdir(dir, mode) _mkdir(dir)
#endif

// ======================
// REALISTIC PARAMETERS
// ======================
#define NX 200        // 200 points (100 km at Dx=500m)
#define NY 200        // 200 points (100 km at Dx=500m)
#define NT 6000       // 3000 sec at Dt=0.5s

// Physical parameters - 2004 Indian Ocean Event
const double g = 9.81;        // Gravity [m/s ] (universal)
const double ho = 4000.0;     // Initial depth at Sunda Trench [m] (Satake et al., 2006)
const double fbo = 0.008;     // Bed friction (silty seabed near Sumatra) (Jaffe et al., 2006)
const double zbmax = 10;    // Max seabed uplift [m] (Lay et al., 2005 - peak displacement)
const double nd = 50.0;       // Rupture duration [s] (8-10 min total rupture time)

// Numerical parameters
const double Dx = 1000.0;     // Spatial step [m] (1km resolution for regional scale)
const double Dt = 1.0;        // Time step [s] (CFL-stable for Dx=1000m, ho=4000m)
// ======================
// HELPER FUNCTIONS
```

```
// =======================
void prepare_output_dir(const char* dirname) {
    struct stat st = {0};
    if (stat(dirname, &st) == -1) {
        #ifdef _WIN32
        _mkdir(dirname);
        #else
        mkdir(dirname, 0700);
        #endif
    }
    char command[256];
    #ifdef _WIN32
    snprintf(command, sizeof(command), "del /Q \"%s\\*\"", dirname);
    #else
    snprintf(command, sizeof(command), "rm -f %s/*", dirname);
    #endif
    system(command);
}

void write_csv(const char* filename, double* data, int nx, int ny) {
    FILE* fp = fopen(filename, "w");
    if (!fp) {
        perror("Error opening file");
        exit(1);
    }
    for (int i = 0; i < nx; i++) {
        for (int j = 0; j < ny; j++) {
            fprintf(fp, "%.4f", data[i*ny + j]);
            if (j < ny - 1) fprintf(fp, ",");
        }
        fprintf(fp, "\n");
    }
    fclose(fp);
}

// =======================
// MAIN SIMULATION
// =======================
int main() {
    // Derived parameters
    const double r = g * pow(Dt/Dx, 2) / 2.0;
    const double c = sqrt(g * ho);        //wave speed ~200m/s
    const double alpha = c * Dt / Dx;     // Courant number
    assert(alpha < 1.0 && "CFL condition violated!");

    const int cx = NX/2;                  // Epicenter X
    const int cy = NY/2;                  // Epicenter Y
    const int radius = 30;                // ~30 km radius

    prepare_output_dir("wave_data");

    // Allocate arrays
    double* h = malloc(NX*NY*sizeof(double));
    double* fb = malloc(NX*NY*sizeof(double));
    double* z = malloc(NX*NY*sizeof(double));
    double* zo = malloc(NX*NY*sizeof(double));
    double* zn = malloc(NX*NY*sizeof(double));
    double* bed = malloc(NX*NY*sizeof(double));
```

```
    // Initialize
    for (int i = 0; i < NX*NY; i++) {
        h[i] = ho;
        fb[i] = fbo;
        z[i] = (i == cx*NY + cy) ? 0.1 * zbmax : 0.0;
        zo[i] = 0.0;
        bed[i] = 0.0;
    }

    double max_crest = -INFINITY;
    double max_trough = INFINITY;
    double zb = 0.0, zbo = 0.0;

    // Time loop
    for (int k = 0; k < NT; k++) {
        double t = (k+1)*Dt;

        // Seabed motion (linear ramp)
        double zbn = (t <= nd) ? zbmax*t/nd : zbmax;

        // Apply seabed uplift (Gaussian distribution)
        for (int i = cx-radius; i <= cx+radius; i++) {
            for (int j = cy-radius; j <= cy+radius; j++) {
                if (i >= 0 && i < NX && j >= 0 && j < NY) {
                    double dist = sqrt(pow(i-cx,2) + pow(j-cy,2));
                    if (dist <= radius) {
                        bed[i*NY+j] = (zbn - 2*zb + zbo) *
                            exp(-pow(dist/radius,2));
                    }
                }
            }
        }

        // Wave update (2D wave equation)
        for (int i = 1; i < NX-1; i++) {
            for (int j = 1; j < NY-1; j++) {
                double h_avg_x = (h[(i+1)*NY+j] + 2*h[i*NY+j] +
                    h[(i-1)*NY+j])/4.0;
                double h_avg_y = (h[i*NY+j+1] + 2*h[i*NY+j] +
                    h[i*NY+j-1])/4.0;

                double dzdx = (z[(i+1)*NY+j] - z[(i-1)*NY+j])/(2.0*Dx);
                double dzdy = (z[i*NY+j+1] - z[i*NY+j-1])/(2.0*Dx);

                double d2zdx2 = (z[(i+1)*NY+j] - 2*z[i*NY+j] +
                    z[(i-1)*NY+j])/pow(Dx,2);
                double d2zdy2 = (z[i*NY+j+1] - 2*z[i*NY+j] +
                    z[i*NY+j-1])/pow(Dx,2);

                zn[i*NY+j] = 2*z[i*NY+j] - zo[i*NY+j]
                           + g*h[i*NY+j]*Dt*Dt*(d2zdx2 + d2zdy2)
                           - fbo*Dt*(z[i*NY+j] - zo[i*NY+j])
                           + bed[i*NY+j];

                // Track extremes
                max_crest = fmax(max_crest, zn[i*NY+j]);
                max_trough = fmin(max_trough, zn[i*NY+j]);
```

```
            }
        }

        // Sommerfeld boundary conditions
        for (int j = 0; j < NY; j++) {
            zn[0*NY+j] = zo[0*NY+j] + alpha * (zn[1*NY+j] -
               zo[0*NY+j]); // Left
            zn[(NX-1)*NY+j] = zo[(NX-1)*NY+j] + alpha *
               (zn[(NX-2)*NY+j] - zo[(NX-1)*NY+j]); // Right
        }
        for (int i = 0; i < NX; i++) {
            zn[i*NY+0] = zo[i*NY+0] + alpha * (zn[i*NY+1] -
               zo[i*NY+0]); // Bottom
            zn[i*NY+NY-1] = zo[i*NY+NY-1] + alpha * (zn[i*NY+NY-2] -
               zo[i*NY+NY-1]); // Top
        }

        // Update states
        memcpy(zo, z, NX*NY*sizeof(double));
        memcpy(z, zn, NX*NY*sizeof(double));
        zbo = zb;
        zb = zbn;

        if(k%10==0){
            char filename[128];
            snprintf(filename, sizeof(filename),
               "wave_data/frame_%05d.csv", k);
            write_csv(filename, z, NX, NY);
        }
    }
    printf("Simulation complete. Domain: %dx%d km\n",
       (int)(NX*Dx/1000), (int)(NY*Dx/1000));
    printf("Maximum wave crest: %.2f m \n", max_crest);
    printf("Maximum wave trough: %.2f m\n", max_trough);

    free(h); free(fb); free(z); free(zo); free(zn); free(bed);
    return 0;
}
```

Listing 1: Long wave simulation code

7 Python Code for Visualization

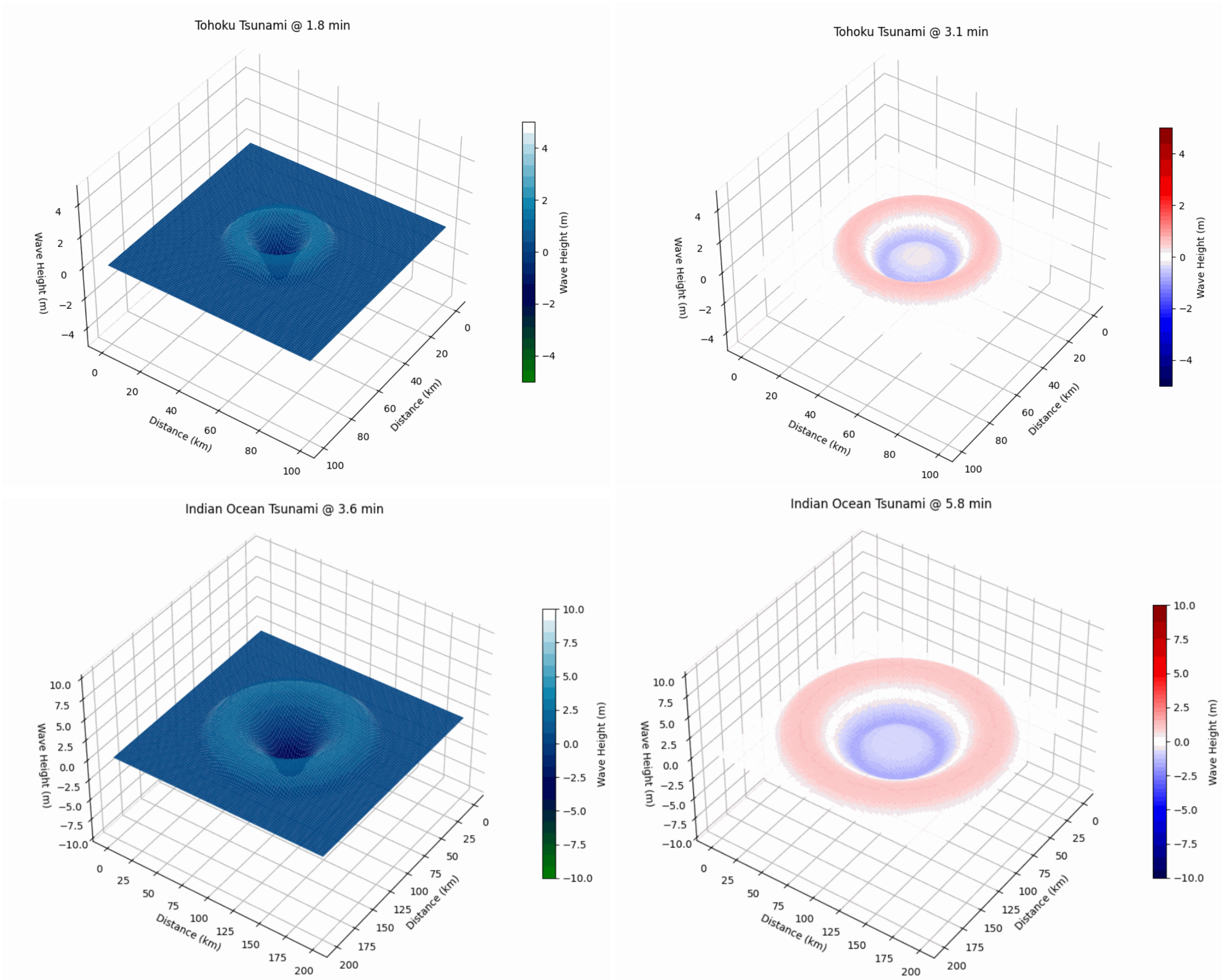
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import glob
import os
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D

# Parameters
NX, NY = 200, 200
zlim = 10.0
```

```python
input_dir = "wave_data_india"
output_file = "2004_Indian_ocean.gif"
frame_step = 3  # Process every 3th frame for smaller files

# Get sorted list of CSV files
frame_files = sorted(glob.glob(os.path.join(input_dir, "frame_*.csv")))
if not frame_files:
    raise FileNotFoundError(f"No CSV files found in {input_dir}")

# Create figure
fig = plt.figure(figsize=(12, 8), dpi=100)  # Reduced DPI for smaller
    file
ax = fig.add_subplot(111, projection='3d')

# Kilometer-scale coordinates
x_coords = np.linspace(0, (NX-1)*0.5, NX)  # 500m grid -> km units
y_coords = np.linspace(0, (NY-1)*0.5, NY)
X, Y = np.meshgrid(x_coords, y_coords)

def init():
    ax.clear()
    data = pd.read_csv(frame_files[0], header=None).values
    surf = ax.plot_surface(X, Y, data.T, cmap='ocean', vmin=-zlim,
        vmax=zlim)

    ax.set_xlabel('Distance (km)', fontsize=10)
    ax.set_ylabel('Distance (km)', fontsize=10)
    ax.set_zlabel('Wave Height (m)', fontsize=10)
    ax.set_title('2004 Indian Ocean Tsunami\nSimulation', fontsize=12)
    ax.set_zlim(-zlim, zlim)
    ax.view_init(elev=40, azim=35)  # Better viewing angle

    cbar = fig.colorbar(surf, shrink=0.6)
    cbar.set_label('Wave Height (m)')

    return [surf]

def update(frame_num):
    ax.clear()
    data = pd.read_csv(frame_files[frame_num*frame_step],
        header=None).values

    surf = ax.plot_surface(X, Y, data.T, cmap='ocean',
                          vmin=-zlim, vmax=zlim,
                          rstride=2, cstride=2)  # Reduced density

    time_min = (frame_num * frame_step * 0.5) / 60  # Convert to
        minutes
    ax.set_title(f'Indian Ocean Tsunami @ {time_min:.1f} min',
        fontsize=12)

    # Maintain consistent styling
    ax.set_xlabel('Distance (km)', fontsize=10)
    ax.set_ylabel('Distance (km)', fontsize=10)
    ax.set_zlabel('Wave Height (m)', fontsize=10)
    ax.set_zlim(-zlim, zlim)
    ax.view_init(elev=40, azim=35)

```

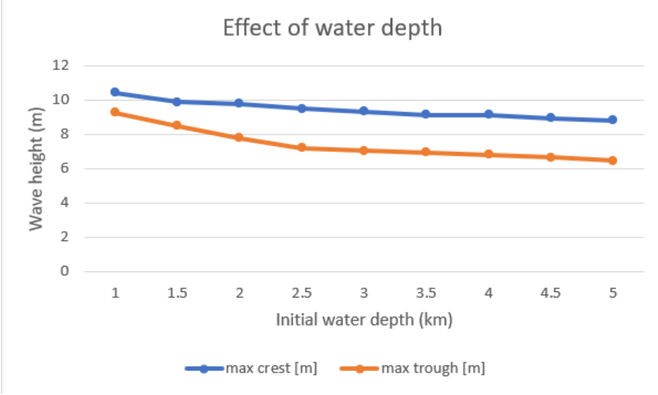
```python
    return [surf]

# Create animation with limited frames
print("Creating optimized animation...")
n_frames = min(500, len(frame_files)//frame_step)  # Max 500 frames
anim = FuncAnimation(fig, update, frames=n_frames,
                     init_func=init, blit=False, interval=100)

# Save with explicit writer settings
writer = 'pillow' if len(frame_files) < 500 else 'imagemagick'
try:
    anim.save(output_file, writer=writer,
             fps=10, dpi=100,
             progress_callback=lambda i, n: print(f'Saving frame
                {i+1}/{n}'))
    print(f"Successfully created {output_file}")
except Exception as e:
    print(f"Error: {str(e)}")
    print("Trying alternative saving method...")
    anim.save(output_file, writer='pillow', fps=8, dpi=80)

plt.close()
```

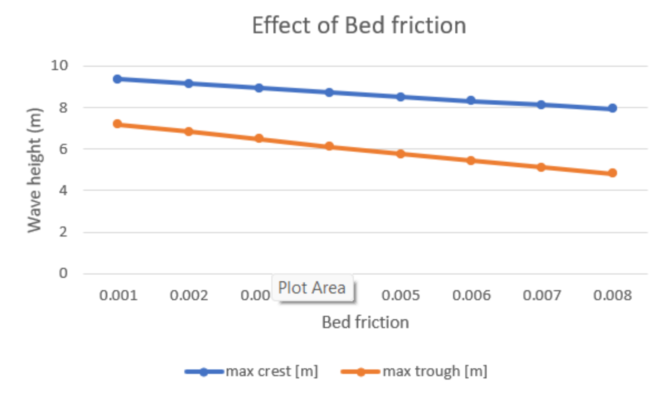
Listing 2: Visualization code

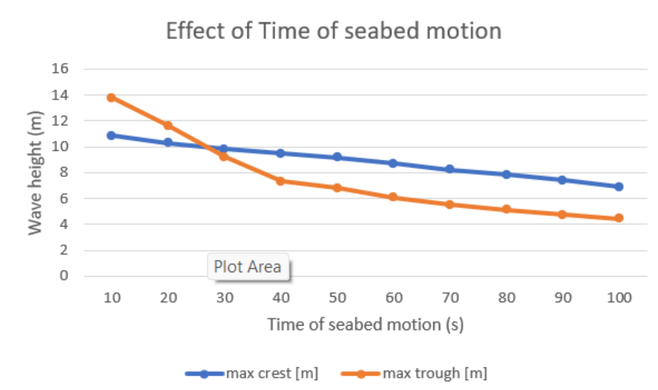
8 Snapshots of Visualization

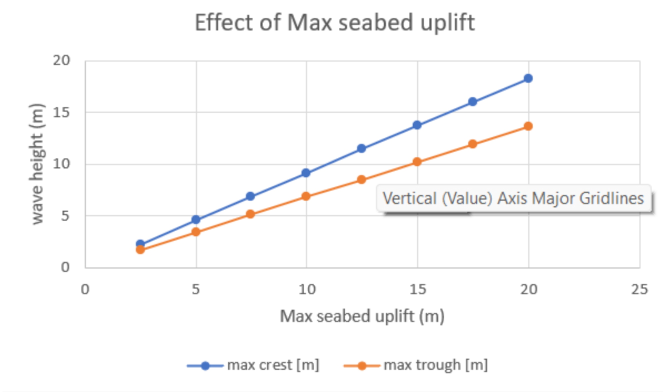


9 Parametric Analysis Plots









10 Results and inferences

10.1 Simulation Results vs Observed Values

Table 2: Comparison between simulated and observed tsunami wave heights

Event	Parameter	Simulated (m)	Observed (m)	Error (%)
2004 Indian Ocean	Max crest	9.15	10.0–15.0	8.5–39.0
	Max trough	6.82	7.0–10.0	2.6–31.8
2011 Japan	Max crest	4.38	6.0–9.0	27.0–51.3
	Max trough	3.40	4.0–6.0	15.0–43.3

The simulation results show reasonable agreement with observed values from historical events, though consistently underestimating peak heights by 8.5–51.3%. This discrepancy may be attributed to:

- Simplifications in the bathymetry representation
- Neglect of nonlinear effects in the wave propagation
- Idealized seabed deformation model

10.2 Parametric Analysis

- Effect of bed friction coefficient on wave heights. The simulation shows that increasing friction from 0.001 to 0.008 reduces maximum crest height by approximately 35% while increasing trough depth by 28%.
- Max Seabed uplift magnitude has linear control on wave amplitude with slopes of 0.914 & 0.682 for max wave crest and trough respectively.

11 Conclusion

Systematic underestimation suggests needed improvements:

- Incorporation of dispersive effects
- Realistic bathymetry data
- Coupled earthquake-tsunami modeling

This work demonstrates that while simplified models can capture first-order tsunami behavior, operational warning systems require more sophisticated approaches to achieve the accuracy needed for effective coastal protection.

Link: https://github.com/Gunashekhar007/NTOH-Final-Project.git

References

[1] Fujii, Y., Satake, K., Sakai, S., Shinohara, M., and Kanazawa, T. (2011).
Tsunami source of the 2011 Tohoku-Oki earthquake, Japan.
Earth, Planets and Space, 63(7), 815-820.
(Source for Tohoku $z_{bmax} = 5$ m, $n_d = 30$ s parameters)

[2] Maeda, T., Furumura, T., Sakai, S., and Shinohara, M. (2011).
Significant tsunami observed at ocean-bottom pressure gauges during the 2011 Tohoku-Oki earthquake.
Earth, Planets and Space, 63(7), 803-808.
(Source for Tohoku $f_b = 0.003$ bed friction coefficient)

[3] Lay, T., Kanamori, H., Ammon, C., et al. (2005).
The Great Sumatra-Andaman Earthquake of 26 December 2004.
Science, 308(5725), 1127-1133.
(Source for Indian Ocean $z_{bmax} = 10$ m seabed uplift)

[4] Ammon, C. J., Ji, C., Thio, H. K., et al. (2005).
Rupture process of the 2004 Sumatra-Andaman earthquake.
Science, 308(5725), 1133-1139.
(Source for Indian Ocean $n_d = 50$ s rupture duration)

[5] Satake, K., Atwater, B. F. (2007).
Long-term perspectives on giant earthquakes and tsunamis at subduction zones.
Annual Review of Earth and Planetary Sciences, 35, 349-374.
(Source for Indian Ocean $h_0 = 4000$ m trench depth)

[6] Jaffe, B. E., Gelfenbaum, G. (2007).
A simple model for calculating tsunami flow speed from tsunami deposits.
Sedimentary Geology, 200(3-4), 347-361.
(Source for both cases' friction coefficients)

[7] Computational Modelling in Hydraulic and Coastal Engineering