Labwork Report

Finite Elements
AP3001-FE

Numerical Research on Tsunami based on Spherical FEM model

Authors:

Yuning Zhang(5274796)
Danning Chen()

November 12, 2020

Delft University of Technology

1 Introduction and Problem Statement

As one of the most destructive disaster created by nature, tsunami is hard to understand and predict due to the complex environmental condition happened in nature. Here we report a brief numerical research of megatsunami based on finite element model.

A tsunami can be formalized by a time independent wave equation on the ocean surface of earth, with boundary conditions on the continential borders. We use the real-world geological data from the GSHHG project [1] to build the geometry of the ocean, and meshing it with an open source tool, the Gmsh[2].

Then, 3D linear Lagrange interpolating is used to approximate the target function and build the discreted Galerkin Equation. Plus some numerical technique to evalute integral terms on spherical surface[3], we can develop a functional FEM framework to solve the equation and predict the behavior of a tsunami.

2 Mathematical Model

2.1 Problem Formulation

The tsunami is modelled by a wave equation on the spherical surface of earth, with its domain to be the ocean part of earth surface.

$$-\Delta u - k^2 u = f \quad in\ \Omega$$

where u is the target function define the wave of tsunami. k is a given constant characterizing the wavelength of tsunami. f is the incentive function which produces the wave.

The coastline is denoted as Γ_1. We assume that the tsunami can only propagate in the ocean area thus a Dirichlet boundary should be applied.

$$u = 0 \quad on\ \Gamma_1$$

The Γ_2 is the edge of the map when we map the 3D surface of earth onto a 2D plane. An absorption boundary condition is applied as above, indicating that only outward directional wave can travel through the boundary and there is no wave come in from outside.

$$\frac{\partial u}{\partial n} + iku = 0, \quad on\ \Gamma_2$$

Tsunami is driven by an incentive function f, which is set to be a Gaussian bell curve.

$$f = A \exp\left(-\frac{d^2}{2\sigma^2}\right)$$

In which A, σ are control parameters, and d is the geodesic distance between the input location and the epicenter of the tsunami. On 3D spherical surface, the geodesic distance is

$$d = R \arccos(\mathbf{n} \cdot \mathbf{n_0})$$

where $\mathbf{n}$ is the normalized vector of input location $\mathbf{x}$ and $\mathbf{n_0}$ is the normalized vector of the given epicenter $\mathbf{x}_0$.

2.2 Weak formulation

On 3D spherical surface, the ocean surface is closed. Thus the boundary Γ_2 produced by the 2D projection is eliminated. To give a general form for both projected and spherical cases, we preserved the Γ_2 boundary condition in the derivation of weak formulation.

First, we multiply a test function ϕ with the PDE, and integrate it over Ω. Note here $\phi = 0$ on Γ_2.

$$\int_\Omega \phi(-\nabla^2 u - k^2 u - f) d\Omega = 0$$

Integrate by part,

$$\int_\Omega \nabla\phi \cdot \nabla u - \nabla \cdot (\phi \nabla u) d\Omega + \int_\Omega \phi(-k^2 u - f) d\Omega = 0$$

By Gauss's theorem,

$$\int_\Omega \nabla\phi \cdot \nabla u - \phi(k^2 u + f) d\Omega - \int_\Gamma \phi \frac{\partial u}{\partial n} d\Gamma = 0$$

Apply the Boundary condition of ϕ and u,

$$\int_\Omega \nabla\phi\cdot\nabla u - \phi(k^2u+f)d\Omega + ik\int_{\Gamma_2}\phi u d\Gamma = 0$$

Above equation, together with essential boundary condition $u = 0$ on Γ_1, is the weak formulation of the problem.

2.3 Galerkin Equation on Spherical Surface

We approximate u with a set of basis functions.

$$u \approx u_n = \sum_{i=1}^{n} \alpha_i\varphi_i$$

Notice u_n need to satisfy the essential boundary condition, so all basis functions should fulfill $\varphi_i = 0$ on Γ_1And we choose the test function ϕ to be φ_j, Now the weak formulation turns to be

$$\sum_{i=1}^{n}\alpha_i(\int_\Omega \nabla\varphi_i\cdot\nabla\varphi_j - k^2\varphi_i\varphi_j d\Omega + ik\int_{\Gamma_2}\varphi_j\varphi_i d\Gamma) = \int_\Omega f\varphi_j d\Omega \quad j = 1,2,...,$$

On spherical surface, we can drop the Γ_2 terms. Then the above formula can be safely writen in matrix form as

$$M\alpha = \mathbf{f}$$

where

$$M_{ij} = \int_\Omega \nabla\varphi_i\cdot\nabla\varphi_j - k^2\varphi_i\varphi_j d\Omega, \quad f_j = \int_\Omega f\varphi_j d\Omega$$

2.4 Linear Lagrange Interpolating

The derivation above is independent from choice of coordinates. While the spherical coordinates seems more promising for spherical surface, it may lead to singularity problems on the north and south poles. Thus we use Cartesian coordinates in the problem.

In 2D problems, the exact result of finite element integral can be obtained with a linear trignalar element. On the spherical surface, the element can be handled as semi-planar given that the elements size is relatively small compared with the sphere radius. With a generalized derivation of Sylvester's formula, we can recover most of the properties in 2D planar cases[3].

Use a linear Lagrange interpolating to approximate the target function, which is define to be

$$\varphi_i \text{ is linear}, \quad \varphi_i(\mathbf{x}_j) = \delta_{ij}$$

Givn a triangle element S_n, denote the three nodes to be $(x_1^{(n)}, y_1^{(n)}, z_1^{(n)}), (x_2^{(n)}, y_2^{(n)}, z_2^{(n)}), (x_3^{(n)}, y_3^{(n)}, z_3^{(n)})$ and linear functions to be $\varphi_1^{(n)}, \varphi_2^{(n)}, \varphi_3^{(n)}$ respectively. We have

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1^{(n)} & x_2^{(n)} & x_3^{(n)} \\ y_1^{(n)} & y_2^{(n)} & y_3^{(n)} \\ z_1^{(n)} & z_2^{(n)} & z_3^{(n)} \end{pmatrix} \begin{pmatrix} \varphi_1^{(n)} \\ \varphi_2^{(n)} \\ \varphi_3^{(n)} \end{pmatrix}$$

Denote

$$\Delta_{det}^{(n)} = \begin{vmatrix} x_1^{(n)} & x_2^{(n)} & x_3^{(n)} \\ y_1^{(n)} & y_2^{(n)} & y_3^{(n)} \\ z_1^{(n)} & z_2^{(n)} & z_3^{(n)} \end{vmatrix}$$

we have

$$\varphi_i^{(n)}(x,y,z) = \frac{a_i^{(n)}x + b_i^{(n)}y + c_i^{(n)}z}{\Delta_{det}^{(n)}}$$

with

$$a_i^{(n)} = y_j^{(n)}z_k^{(n)} - y_k^{(n)}z_j^{(n)}, \quad b_i^{(n)} = x_k^{(n)}z_j^{(n)} - x_j^{(n)}z_k^{(n)}, \quad c_i^{(n)} = x_j^{(n)}y_k^{(n)} - x_k^{(n)}y_j^{(n)}$$

and

$$i, j, k = 1, 2, 3$$

Thus we can write

$$\nabla \varphi_i^{(n)}(x, y, z) = \frac{1}{\Delta_{det}^{(n)}} \begin{pmatrix} a_i^{(n)} \\ b_i^{(n)} \\ c_i^{(n)} \end{pmatrix} = \frac{1}{\Delta_{det}^{(n)}} \begin{pmatrix} y_j^{(n)} z_k^{(n)} - y_k^{(n)} z_j^{(n)} \\ x_k^{(n)} z_j^{(n)} - x_j^{(n)} z_k^{(n)} \\ x_j^{(n)} y_k^{(n)} - x_k^{(n)} y_j^{(n)} \end{pmatrix} = \frac{\mathbf{x}_j \times \mathbf{x}_k}{\Delta_{det}^{(n)}}$$

If we define $\mathbf{N}^{(n)}$ to be the outward pointing normal to the triangle, then

$$\mathbf{N^{(n)}}(\mathbf{x} - \mathbf{x_i}^{(n)}) = 0$$

and $\mathbf{N}$ can be writen as

$$\mathbf{N}^{(n)} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ x_2^{(n)} - x_1^{(n)} & y_2^{(n)} - y_1^{(n)} & z_2^{(n)} - z_1^{(n)} \\ x_3^{(n)} - x_1^{(n)} & y_3^{(n)} - y_1^{(n)} & z_3^{(n)} - z_1^{(n)} \end{vmatrix}$$

The length of the normal vector is just

$$\Delta_{cros}^{(n)} = |\mathbf{N}^{(n)}|$$

which is exactly twice the area of the element triangle.

2.5 Element Matrix and Element Vector

Next we can extend Holand-Bell Theorem into three dimensional trianglar elements, using

$$\int \varphi_1^\alpha \varphi_2^\beta \varphi_3^\gamma d\Omega = \frac{\alpha!\beta!\gamma!}{(\alpha + \beta + \gamma + 2)!} \Delta_{cros}$$

For an element S_n, let $i, j, k = 1, 2, 3$ to be the local indexes for the linear functions

$$M_{ij}^{(n)} = \int_\Omega \nabla \varphi_i^{(n)} \cdot \nabla \varphi_j^{(n)} - k^2 \varphi_i^{(n)} \varphi_j^{(n)} d\Omega \simeq \left(\frac{a_i^{(n)} a_j^{(n)} + b_i^{(n)} b_j^{(n)} + c_i^{(n)} c_j^{(n)}}{2(\Delta_{det}^{(n)})^2} - \frac{k^2}{24} \right) \Delta_{cros}^{(n)}$$

And the element matrix would be

$$M^{(n)} \simeq \begin{pmatrix} M_{11}^{(n)} & M_{12}^{(n)} & M_{13}^{(n)} \\ M_{21}^{(n)} & M_{22}^{(n)} & M_{23}^{(n)} \\ M_{31}^{(n)} & M_{32}^{(n)} & M_{33}^{(n)} \end{pmatrix}$$

The element vector can be write as

$$f^{(n)} \simeq \frac{\Delta_{cros}}{6} \begin{pmatrix} f(\mathbf{x_1}^{(n)}) \\ f(\mathbf{x_2}^{(n)}) \\ f(\mathbf{x_3}^{(n)}) \end{pmatrix}$$

Using above derivation, we can get the numerical approximation of each integral terms appearing

2.6 Assembly of Finite Elements

Given the element matrix and vectors, we can assembly the large matrix used in the Galerkin's Equation. Suppose there are N elemetns contained in the mesh. The key issue here is that we need to find the map from local indices of an element to the global one.

$$g : (n, i) \mapsto k$$

where $n = 1, 2, \ldots N$ is the index of the elements and $i = 1, 2, 3$ is the indices of the three local nodes contained in that element. And k is the global indices of nodes in the mesh. So the large matrix element can be assembled as a conditional sum

$$M_{pq} = \sum_n \sum_{i|g(n,i)=p} \sum_{j|g(n,j)=q} M_{ij}^{(n)}$$

The assembly of element vector is similar, where we need to find a map

$$f_k = \sum_n \sum_{i|g(n,i)=k} f_i^{(n)}$$

In practice, we iterate over the N elements to build the element matrix and vector for each one, then we find the global index for each local node and add the term in local element matrix to corresponding entry in large matrix.

3 Implementation

3.1 Geometry Building

The earth coastline deta is obtained from the GSHHG project[1]. We use a `Python` script to convert the data to the `.geo` format supported by the Gmsh software. The original data is stored in the form of latitude ϕ and longitude λ, which can be converted into Cartesian coordinates with

$$x = R\,cos(\phi)cos(\lambda),\ y = R\,cos(\phi)sin(\lambda),\ z = R\,sin(\phi)$$

in which R is the radius of earth. Since the numerics in the `.geo` file is dimensionless, it's quite natural to set $R = 6.371 \times 10^6$ so that the nuit length is 1 kilometer.

The coordinates in the coastline data is converted into the `Point` entity defined by Gmsh . And the borders of continents are represented by `BSpline`, which is a basis spline controled by the list of points on the coastline. This promises basic smoothness using a minimum amount of points.

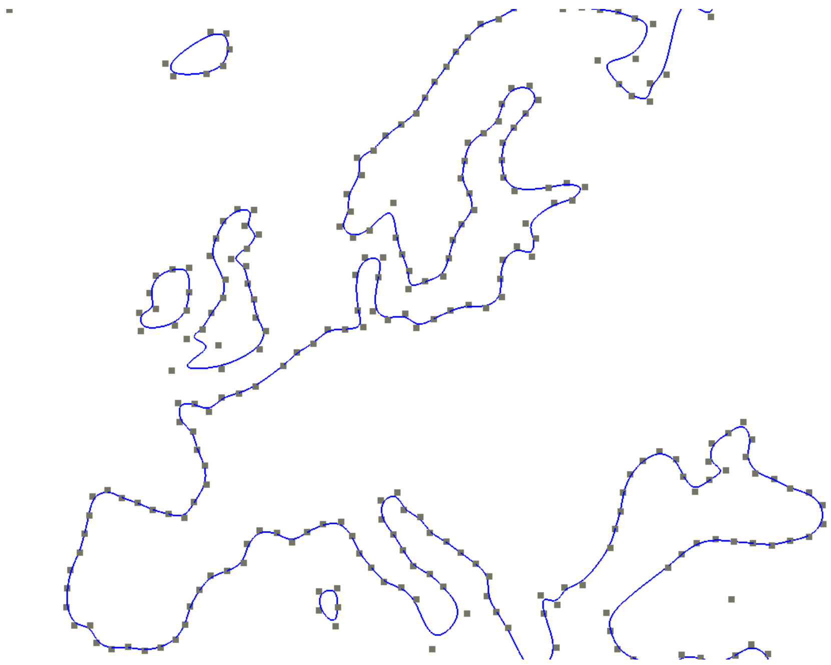


Figure 1: Contour of Europe interpolated by B-sline

The set of all continential boundaries constitutes the "Coast", defined to be a `Physical Line`. And the spherical surface closed by all the curves defines a `Physical Surface` where the meshing will be performed, that's exactly the "Sea".

3.2 Mesh Generation

We use Gmsh to build the meshing of the ocean surface, that is the `Physical Surface` we defined before. We expect that the mesh should be dense near the coast and sparse in the center of great ocean. To achieve this, we use two `Field` entities to adjust the size of mesh elements at different location.

First we define an `Attractor Field` depicting the distance from a field point to the nearst coastline. This field determines that the element size will increase when the meshing point goes further from the coastline.

However, we should notice that the element size can be too small thus inefficient near the boundary. Also, generally the minimum meshing sizet should not surpass 1/6 wavelength when running FEM to simulate the wave behavior.

So another `Threshold Field` is applied to set a limitation on the element size. Here we set the lower bound of element size to be 150km and maximum size to be 900km.

The mesh is generated by `Frontal-Delaunay` algorithm, with 12633 nodes and 23068 elements, as shown in Fig.2.

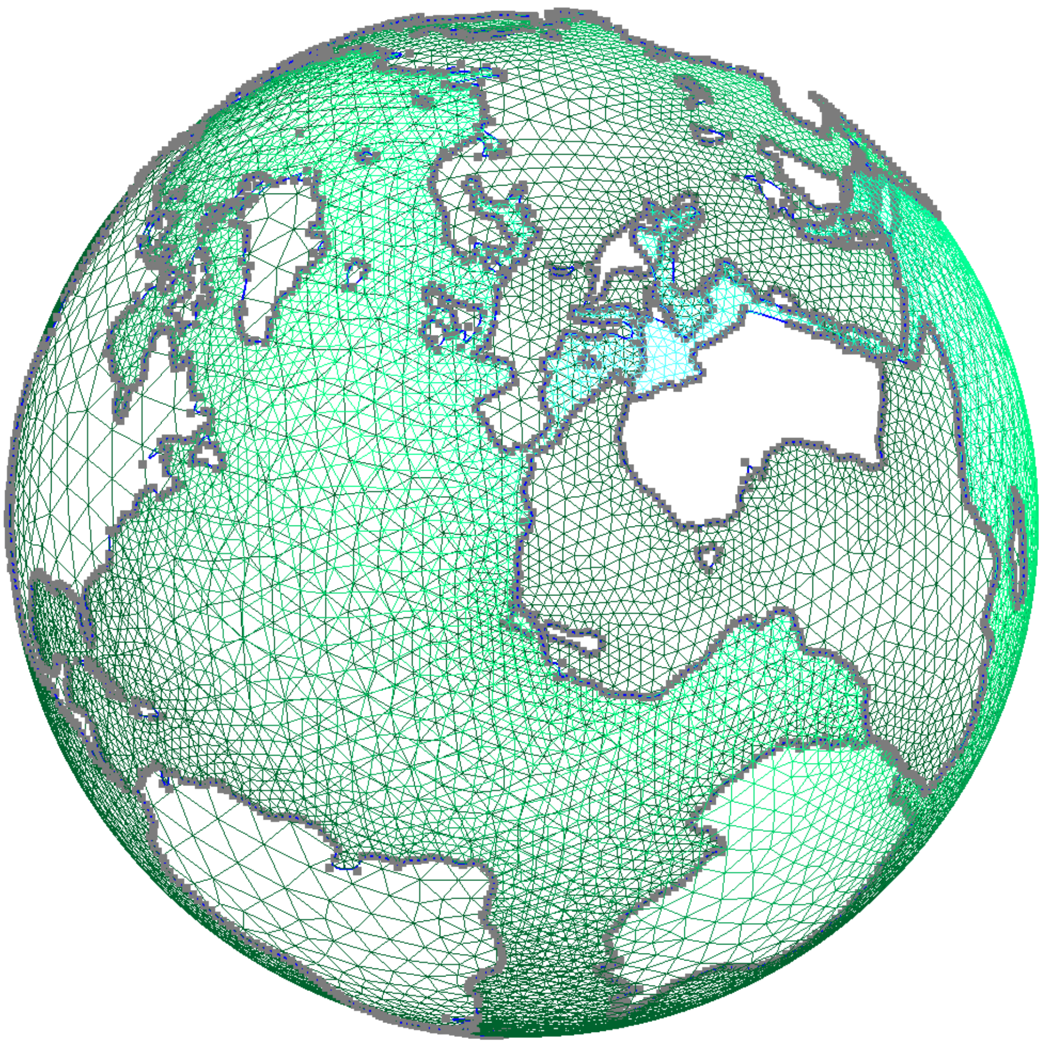


Figure 2: World Ocean Meshing with Gmsh

The mesh file is exported in the `.msh` format, at version 2.2.

3.3 Algorithm Implementation

The algorithm is implemented in `Julia`, based on the code provided in the sample. The Gmsh format data is processed using the `CompScienceMeshes.jl` library. We added some necessary modifications in the module of elementary matrix and incentive function.

The basic procedures implemented in the FEM are listed below

1. Load meshing file and parse the data.
2. Detect boundary nodes and associate nodes with elements.
3. Iterate over elements and build element matrix and vectors.

4. Assembly the matrix and vectors in Galerkin Equation
5. Solve the equation with numerical inversion of the matrix
6. Post-processing and visualization

The embeded array-based syntax and operation makes vector calculation quite convenient. The inversion of matrix, which is the most computation-intensive part of FEM, is implemented using the build-in function of Julia.

4 Results and Discussion

4.1 Result Demonstration

We start the simulation using following parameters, with an epicenter at North Atlantic (40.4,-43.2).

Name	Description	value
R	radius of the earth	6.371×10^6
A	amplitude of incentive function f	100
σ	spreading of incentive function f	10^5
ϕ_0	latitude of the epicenter	$40.4\pi/180$
λ_0	longitude of the epicenter	$-43.2\pi/180$
k	wavelength parameter in wave equation	$2\pi/(4 \times 10^6)$

Table 1: Parameter List of Modeling

With more than 10^4 element, the solution of this FEM model can be finished in 10 sec on a 8-core desktop PC, which has proved the efficiency of the algorithm.

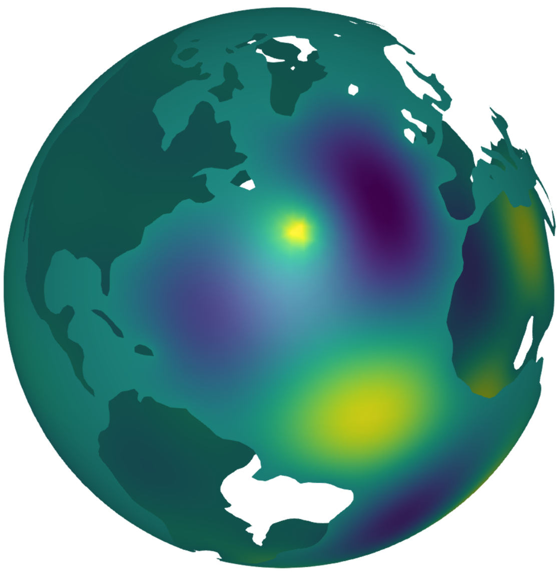
The result of simulation is showed in the Fig.3. We picked four different view from North Atlantic Ocean,South Atlantic Ocean, Indian Ocean, and Pacific Ocean respectively.

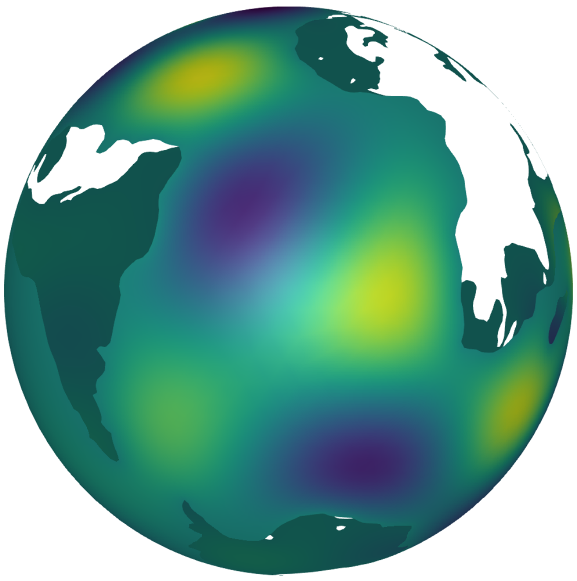
Due to the input of incentive function, a bright spot is visuable at the epicenter of North Atlantic in 3a. The tsunami wave traveled through the entire Atlantic Ocean and impacted the coastline of Indian Ocean. The wavelength is set to be 4000 km thus we can see huge crests and troughs across the entire ocean surface.

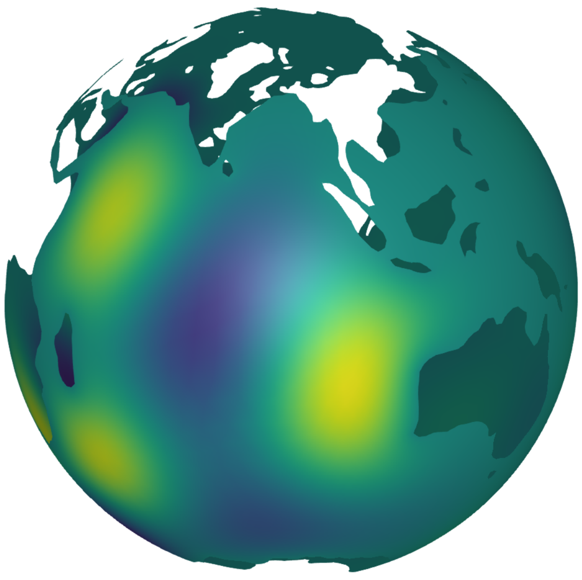
As expected, the tsunami didn't affect the Pacific Ocean as shown in 3d. One reason is the incentive function f is set to be a Gaussian curve with $\sigma = 100$ km, decaying fast with the geodesic distance to the epicenter. Another reason is the fact that Pacific Ocean is surrounded by serveral continents, with the west side adjacent to Asia and Australia and east side closed by North and South America. With a long wavelength (4000 km), the tsunami wave can't travel through the narrow channels to Pacific, and diffraction is also disabled.

4.2 Further Improvement

A more reasonable and conventional way to meshing the world ocean is using ocean bathymetry data[4], which has been implemented in many oceanographical researches [5]. Based on bathymetry, the mesh size is supposed to be smaller in shallow ocean area and larger in deep sea region. This can be achieved by importing a `Structured Field` in `Gmsh`. Also, we can add a smaller limitations on minimum and maximum meshing size, in order to get a dense mesh.


(a) North Atlantic Ocean


(b) South Atlantic Ocean


(c) Indian Ocean

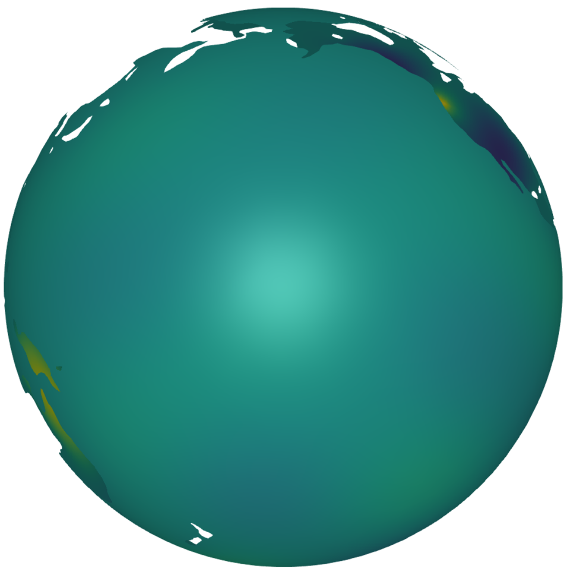

(d) Pacific Ocean

Figure 3: Tsunami visualization above different oceans area

References

P. Wessel and W. H. Smith, "A global, self-consistent, hierarchical, high-resolution shoreline database," *Journal of Geophysical Research: Solid Earth*, vol. 101, no. B4, pp. 8741–8743, 1996.

C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities," *International journal for numerical methods in engineering*, vol. 79, no. 11, pp. 1309–1331, 2009.

F. X. Giraldo, "Lagrange–galerkin methods on spherical geodesic grids," *Journal of Computational Physics*, vol. 136, no. 1, pp. 197–213, 1997.

C. Amante and B. Eakins, "Etopo1 1 arc-minute global relief model: Procedures, data sources and analysis. memorandum nesdis ngdc-24," *National Geophysical Data Center, NOAA*, 2009.

S. Legrand, E. Deleersnijder, E. Hanert, V. Legat, and E. Wolanski, "High-resolution, unstructured meshes for hydrodynamic models of the great barrier reef, australia," *Estuarine, coastal and shelf science*, vol. 68, no. 1-2, pp. 36–46, 2006.

A File Inventory Introduction

- `./mega_tsunami.jl`: Julia code to run the main algorithm of PDE
- `./gshhs2geo.py`: Python script to convert GSHHS data into .geo format
- `./mesh/gshhs/`: data folder containing data from GSHHS program
- `./mesh/world.geo`: Gmsh .geo format file that contains the model geometry
- `./mesh/world.msh`: Gmsh .msh format file that contains mesing of the model