Generation and propagation of a Tsunami wave : a new mesh adaptation technique

Georges Sadaka

Generation and propagation of a *Tsunami* wave : a new mesh adaptation technique

Georges Sadaka*

Abstract : The importance of the study of the propagation of a *Tsunami* wave came from the complex phenomenon and its natural disasters which represents a major risk for populations. To model this phenomena, we will consider a *simplified* Boussinesq [1] system of BBM [2] type (sBBM) derived by D. Mitsotakis in [9], over a flat bottom then over a variable bottom in space and in time and apply this system, first, using a mesh generated using a photo of the Mediterranean sea, second, using a mesh generated using an imported xyz bathymetry for the sea near Java island and then we will consider a realistic example of the *Tsunami* wave near Java island which happened in 2006.
We choose here to use FreeFem++ [8] software which simplifies the construction of the domain, in particular, one of the advantage of FreeFem++ is that we can build a mesh using a photo and we can easily export bathymetric data in order to consider more realistic simulations where a special adapt mesh technique applied for these two methods is detailed in the sequel.

Contents

1 Introduction

We consider here the numerical simulation of the sBBM in 2D over a variable bottom in space $D(x, y)$ and in time $\zeta(x, y, t)$:

$$\begin{aligned} &\eta_t + \nabla\cdot((D+\eta+\zeta)V) + \zeta_t + \tilde{A}\nabla\cdot\left(D^2\nabla\zeta_t\right) + \nabla\cdot\left\{AD^2\left[\nabla\left(\nabla D\cdot V\right) + \nabla D\nabla\cdot V\right] - bD^2\nabla\eta_t\right\} = 0,\\ &V_t + g\nabla\eta + \tfrac{1}{2}\nabla|V|^2 + BgD\left[\nabla\left(\nabla D\cdot\nabla\eta\right) + \nabla D\Delta\eta\right] - dD^2\Delta V_t - BD\nabla\zeta_{tt} = 0, \end{aligned} \tag{1}$$

where

$$\widehat{a} = \left(\theta - \frac{1}{2}\right), \qquad \widehat{b} = \frac{1}{2}\left((\theta-1)^2 - \frac{1}{3}\right), \qquad \tilde{A} = \nu\widehat{a} - (1-\nu)\widehat{b}, \qquad A = -\widehat{b}, \qquad B = 1-\theta,$$

*Université de Picardie Jules Verne, LAMFA CNRS UMR 7352, 33, rue Saint-Leu, 80039 Amiens, France, http://lamfa.u-picardie.fr/sadaka/ ✉ georges.sadaka@u-picardie.fr

1. [1]
2. Benjamin, Bona and Mahony (BBM)

$$b = \frac{1}{2}\left(\theta^2 - \frac{1}{3}\right)(1-\nu), \qquad d = \frac{1}{2}\left(1-\theta^2\right)(1-\mu) \qquad \text{and} \qquad g = 9.81 \text{ is the gravity.}$$

This system is an approximation to the three-dimensional Euler equations describing the irrotational free surface flow of an ideal fluid $\Omega \subset \mathbb{R}^3$, which is bounded below by $-h(x,y,t) = -D(x,y) - \zeta(x,y,t)$ and above by the free surface elevation $\eta(x,y,t)$ (cf. Figure 1).

The variables in (1) : $\mathbf{X} = (x,y) \in \Omega$ and $t > 0$ are proportional to position along the channel and time, respectively. $\eta = \eta(\mathbf{X},t)$ being proportional to the deviation of the free surface departing from its rest position and $V = V(\mathbf{X},t) = \begin{pmatrix} u(\mathbf{X},t) \\ v(\mathbf{X},t) \end{pmatrix} = (u,v)^T = (u;v)$ being proportional to the horizontal velocity of the fluid at some height. $\nabla\cdot = \begin{pmatrix} \partial_x \cdot \\ \partial_y \cdot \end{pmatrix}$ is the gradient, $\nabla\cdot(\star\,;\cdot) = \partial_x\star + \partial_y\cdot$ is the divergence and $\Delta\cdot = \partial_{xx}\cdot + \partial_{yy}\cdot$ is the laplacian.

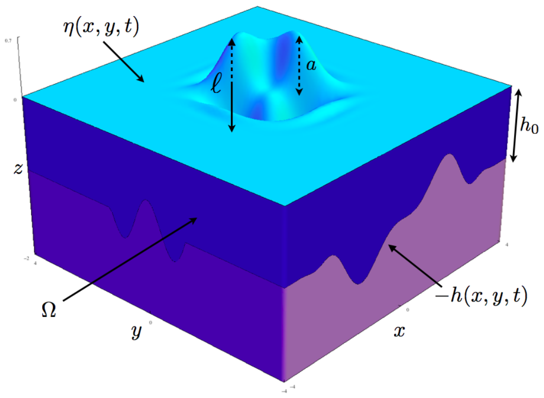


Figure 1 – The domain Ω.

Remark 1. In our study, we suppose that $\eta = \mathcal{O}(a)$, where here the amplitude a is the difference between the water surface and the zero level. Also we set $\lambda = \mathcal{O}(\ell)$ be the wave length. In addition, we limit ourselves to the case where $\eta + D > 0$ (there is no dry zone), since we are in a big deep water wave regime.

This paper is organized as follows: in Section 2, we present the space and time discretization of equation (1). In Section 3, we present a method to build a mesh using a photo and then using an imported *xyz* bathymetry, in which, we will also present a special adaptive mesh technique around the initial data used for the generation of a *Tsunami* wave. In Section 4, we first check the convergence of our code, which establishes the adequacy of the chosen finite element discretization, then we simulate the propagation of a wave, that looks like a *Tsunami* wave generated by an Earthquake, in the Mediterranean sea over the sBBM system (1) with a flat bottom using the mesh generated from a photo of the Mediterranean sea, then with a variable bottom in space using the mesh generated from the xyz bathymetry of the sea near Java island and finally, with a realistic example of the *Tsunami* wave near Java island which happened in 2006.

2 Discretization of the sBBM system

In this section, we present the spatial discretization of (1) using finite element method with $\mathbb{P}_1$ continuous piecewise linear functions and for the time marching scheme an explicit second order Runge-Kutta scheme.

2.1 Spatial discretization

We let Ω be a convex, plane domain, and $\mathcal{T}_h$ be a regular, quasiuniform triangulation of Ω with triangles of maximum size $h < 1$. Setting $V_h = \{v_h \in C^0(\bar{\Omega}); v_h|_T \in \mathbb{P}_1(T), \forall T \in \mathcal{T}_h\}$ be a finite-dimensional subspace of $H^1(\Omega)$, where $\mathbb{P}_1$ is the set of all polynomials of degree ≤ 1 with real coefficients and denoting by $\langle\cdot;\cdot\rangle$ the L^2 inner product on Ω, we consider the weak formulation of system (1) :

Find $\eta_h, u_h, v_h \in V_h$ such that $\forall \phi_h^\eta, \phi_h^u, \phi_h^v \in V_h$, we have :

$$\begin{aligned}
&\left\langle \eta_{ht} - b\nabla\cdot\left(D^2\nabla\eta_{ht}\right) + \nabla\cdot\left((D+\eta_h+\zeta)(u_h;v_h)\right) + \zeta_t;\phi_h^\eta\right\rangle + \left\langle \tilde{A}\nabla\cdot\left(D^2\nabla\zeta_t\right);\phi_h^\eta\right\rangle \\
&\qquad\qquad + \left\langle \nabla\cdot\left\{AD^2\left[\nabla\left(\nabla D\cdot(u_h;v_h)\right) + \nabla D\nabla\cdot(u_h;v_h)\right]\right\};\phi_h^\eta\right\rangle = 0, \\
&\left\langle u_{ht} - dD^2\Delta u_{ht} + g\eta_{xh} + u_h u_{hx} + v_h v_{hx} - BD\zeta_{xtt};\phi_h^u\right\rangle + Bg\left\langle D\Big[(\nabla D\cdot\nabla\eta_h)_x + D_x\Delta\eta_h\Big];\phi_h^u\right\rangle = 0, \\
&\left\langle v_{ht} - dD^2\Delta v_{ht} + g\eta_{yh} + u_h u_{hy} + v_h v_{hy} - BD\zeta_{ytt};\phi_h^u\right\rangle + Bg\left\langle D\left[(\nabla D\cdot\nabla\eta_h)_y + D_y\Delta\eta_h\right];\phi_h^v\right\rangle = 0.
\end{aligned} \tag{2}$$

For simplicity, we set $\phi_h^\eta = \Phi^\eta,\ \phi_h^u = \Phi^u,\ \phi_h^v = \Phi^v,\ \eta_h = \mathcal{E},\ u_h = \mathcal{U}, v_h = \mathcal{V}$, so that system (2) can be rewrite in the following way :

$$\left\{\begin{aligned}
\left\langle \partial_t\mathcal{E} - b\nabla\cdot\left(D^2\nabla\partial_t\mathcal{E}\right);\Phi^\eta\right\rangle &= -\langle (D+\zeta+\mathcal{E})\nabla\cdot(\mathcal{U};\mathcal{V}) + (D_x+\zeta_x+\mathcal{E}_x)\mathcal{U} + (D_y+\zeta_y+\mathcal{E}_y)\mathcal{V} + \zeta_t \\
&\quad + \tilde{A}\nabla\cdot\left(D^2\nabla\zeta_t\right) + A\nabla\cdot\left\{D^2\left[\nabla\left(\nabla D\cdot(\mathcal{U};\mathcal{V})\right) + \nabla D\nabla\cdot(\mathcal{U};\mathcal{V})\right]\right\};\Phi^\eta\Big\rangle \\
&= \mathcal{F}\left(\mathcal{E},\mathcal{U},\mathcal{V},\Phi^\eta\right), \\
\left\langle (I_d - dD^2\Delta)\partial_t\mathcal{U};\Phi^u\right\rangle &= -\left\langle g\mathcal{E}_x + \mathcal{U}\mathcal{U}_x + \mathcal{V}\mathcal{V}_x + BgD\Big[(\nabla D\cdot\nabla\mathcal{E})_x + D_x\Delta\mathcal{E}\Big] - BD\zeta_{xtt};\Phi^u\right\rangle \\
&= \mathcal{G}\left(\mathcal{E},\mathcal{U},\mathcal{V},\Phi^u\right), \\
\left\langle (I_d - dD^2\Delta)\partial_t\mathcal{V};\Phi^v\right\rangle &= -\left\langle g\mathcal{E}_y + \mathcal{U}\mathcal{U}_y + \mathcal{V}\mathcal{V}_y + BgD\Big[(\nabla D\cdot\nabla\mathcal{E})_y + D_y\Delta\mathcal{E}\Big] - BD\zeta_{ytt};\Phi^v\right\rangle \\
&= \mathcal{H}\left(\mathcal{E},\mathcal{U},\mathcal{V},\Phi^u\right).
\end{aligned}\right. \tag{3}$$

Next, we discretize system (3). First, integrating by parts, the left hand side in (3) gives :

$$-\left\langle b\nabla\cdot\left(D^2\nabla\partial_t\mathcal{E}\right);\Phi^\eta\right\rangle = b\left\langle D^2\nabla\partial_t\mathcal{E};\nabla(\Phi^\eta)\right\rangle - \int_{\Gamma_n} bD^2\Phi^\eta\frac{\partial(\partial_t\mathcal{E})}{\partial n}\partial\gamma,$$

$$-\left\langle dD^2\Delta\partial_t\mathcal{U};\Phi^u\right\rangle = d\left\langle D^2\nabla\partial_t\mathcal{U};\nabla\Phi^u\right\rangle + d\left\langle 2D\nabla D\cdot\nabla\partial_t\mathcal{U};\Phi^u\right\rangle - \int_{\Gamma_n} dD^2\Phi^u\frac{\partial(\partial_t\mathcal{U})}{\partial n}\partial\gamma,$$

and

$$-\left\langle dD^2\Delta\partial_t\mathcal{V};\Phi^v\right\rangle = d\left\langle D^2\nabla\partial_t\mathcal{V};\nabla\Phi^v\right\rangle + d\left\langle 2D\nabla D\cdot\nabla\partial_t\mathcal{V};\Phi^v\right\rangle - \int_{\Gamma_n} dD^2\Phi^v\frac{\partial(\partial_t\mathcal{V})}{\partial n}\partial\gamma.$$

Now, dealing with the right-hand side of the first equation in system (3), we have :

$$\left\langle \nabla\cdot\left(D^2\nabla\zeta_t\right);\Phi^\eta\right\rangle = \left\langle \left(D^2\zeta_{xt}\right)_x + \left(D^2\zeta_{yt}\right)_y;\Phi^\eta\right\rangle = \left\langle 2DD_x\zeta_{xt} + D^2\zeta_{xxt} + 2DD_y\zeta_{yt} + D^2\zeta_{yyt};\Phi^\eta\right\rangle,$$

and

$$\begin{aligned}
&\left\langle \nabla\cdot\left\{D^2\left[\nabla\left(\nabla D\cdot(\mathcal{U};\mathcal{V})\right) + \nabla D\nabla\cdot(\mathcal{U};\mathcal{V})\right]\right\};\Phi^\eta\right\rangle \\
&= \left\langle \nabla\cdot\left\{D^2\left[\Big((D_x\mathcal{U}+D_y\mathcal{V})_x;(D_x\mathcal{U}+D_y\mathcal{V})_y\Big) + (D_x\nabla\cdot(\mathcal{U};\mathcal{V});D_y\nabla\cdot(\mathcal{U};\mathcal{V}))\right]\right\};\Phi^\eta\right\rangle \\
&= \langle \nabla\cdot\left(D^2D_{xx}\mathcal{U} + D^2D_x\mathcal{U}_x + D^2D_{xy}\mathcal{V} + D^2D_y\mathcal{V}_x + D^2D_x\nabla\cdot(\mathcal{U};\mathcal{V});D^2D_{xy}\mathcal{U} + D^2D_x\mathcal{U}_y\right. \\
&\qquad \left. + D^2D_{yy}\mathcal{V} + D^2D_y\mathcal{V}_y + D^2D_y\nabla\cdot(\mathcal{U};\mathcal{V})\right);\Phi^\eta\rangle \\
&= \langle (2DD_xD_{xx} + 2DD_yD_{xy} + D^2D_{xyy} + D^2D_{xxx})\mathcal{U} + (2DD_xD_{xy} + 2DD_yD_{yy} + D^2D_{yyy} \\
&\quad + D^2D_{xxy})\mathcal{V} + (4DD_x^2 + 3D^2D_{xx} + 2DD_y^2 + D^2D_{yy})\mathcal{U}_x + 2(D^2D_{xy} + DD_xD_y)\mathcal{U}_y + (4DD_y^2 \\
&\quad + 3D^2D_{yy} + 2DD_x^2 + D^2D_{xx})\mathcal{V}_y + 2(DD_xD_y + D^2D_{xy})\mathcal{V}_x;\Phi^\eta\rangle + \left(\left\langle 2D^2D_x\mathcal{U}_{xx};\Phi^\eta\right\rangle\right. \\
&\quad \left. + \left\langle D^2D_y\mathcal{U}_{xy};\Phi^\eta\right\rangle + \left\langle D^2D_x\mathcal{U}_{yy};\Phi^\eta\right\rangle + \left\langle D^2D_y\mathcal{V}_{xx};\Phi^\eta\right\rangle + \left\langle D^2D_x\mathcal{V}_{xy};\Phi^\eta\right\rangle + \left\langle 2D^2D_y\mathcal{V}_{yy};\Phi^\eta\right\rangle\right);
\end{aligned}$$

On the other hand, we have :

$$\left\langle 2D^2D_x\mathcal{U}_{xx};\Phi^\eta\right\rangle = -\left\langle 2D^2D_x\mathcal{U}_x;\Phi_x^\eta\right\rangle - \left\langle (4DD_x^2 + 2D^2D_{xx})\mathcal{U}_x;\Phi^\eta\right\rangle + \int_{\Gamma_n} 2D^2D_x\Phi^\eta\frac{\partial\mathcal{U}}{\partial n}\partial\gamma,$$

$$\left\langle D^2D_y\mathcal{U}_{xy};\Phi^\eta\right\rangle = -\left\langle D^2D_y\mathcal{U}_x;\Phi_y^\eta\right\rangle - \left\langle (2DD_y^2 + D^2D_{yy})\mathcal{U}_x;\Phi^\eta\right\rangle + \int_{\Gamma_n} D^2D_y\Phi^\eta\frac{\partial\mathcal{U}}{\partial n}\partial\gamma,$$

$$\left\langle D^2D_x\mathcal{U}_{yy};\Phi^\eta\right\rangle = -\left\langle D^2D_x\mathcal{U}_y;\Phi_y^\eta\right\rangle - \left\langle (2DD_xD_y + D^2D_{xy})\mathcal{U}_y;\Phi^\eta\right\rangle + \int_{\Gamma_n} D^2D_x\Phi^\eta\frac{\partial\mathcal{U}}{\partial n}\partial\gamma,$$

$$\left\langle D^2D_y\mathcal{V}_{xx};\Phi^\eta\right\rangle = -\left\langle D^2D_y\mathcal{V}_x;\Phi_x^\eta\right\rangle - \left\langle (2DD_xD_y + D^2D_{xy})\mathcal{V}_x;\Phi^\eta\right\rangle + \int_{\Gamma_n} D^2D_y\Phi^\eta\frac{\partial\mathcal{V}}{\partial n}\partial\gamma,$$

$$\left\langle D^2D_x\mathcal{V}_{xy};\Phi^\eta\right\rangle = -\left\langle D^2D_x\mathcal{V}_x;\Phi_y^\eta\right\rangle - \left\langle (2DD_xD_y + D^2D_{xy})\mathcal{V}_x;\Phi^\eta\right\rangle + \int_{\Gamma_n} D^2D_x\Phi^\eta\frac{\partial\mathcal{V}}{\partial n}\partial\gamma,$$

$$\left\langle 2D^2D_y\mathcal{V}_{yy};\Phi^\eta\right\rangle = -\left\langle 2D^2D_y\mathcal{V}_y;\Phi^\eta_y\right\rangle - \left\langle (4DD_y^2+2D^2D_{yy})\mathcal{V}_y;\Phi^\eta\right\rangle + \int_{\Gamma_n} 2D^2D_y\Phi^\eta\frac{\partial\mathcal{V}}{\partial n}\partial\gamma,$$

and, consequently,

$$\begin{aligned}\mathcal{F}(\mathcal{E},\mathcal{U},\mathcal{V},\Phi^\eta) = &-\langle (D+\zeta+\mathcal{E})\nabla\cdot(\mathcal{U};\mathcal{V}) + (D_x+\zeta_x+\mathcal{E}_x)\mathcal{U} + (D_y+\zeta_y+\mathcal{E}_y)\mathcal{V} + \zeta_t;\Phi^\eta\rangle \\ &-\tilde{A}\left\langle 2DD_x\zeta_{xt} + D^2\zeta_{xxt} + 2DD_y\zeta_{yt} + D^2\zeta_{yyt};\Phi^\eta\right\rangle - A\left\langle (2DD_xD_{xx} + 2DD_yD_{xy} + D^2D_{xyy}\right. \\ &+D^2D_{xxx})\mathcal{U} + (2DD_xD_{xy} + 2DD_yD_{yy} + D^2D_{yyy} + D^2D_{xxy})\mathcal{V} + D^2D_{xx}\mathcal{U}_x + D^2D_{xy}\mathcal{U}_y - 2DD_xD_y\mathcal{V}_x \\ &\left.+(D^2D_{yy} + 2DD_x^2 + D^2D_{xx})\mathcal{V}_y;\Phi^\eta\right\rangle + A\left(\left\langle 2D^2D_x\mathcal{U}_x + D^2D_y\mathcal{V}_x;\Phi^\eta_x\right\rangle + \left\langle D^2D_y\mathcal{U}_x + D^2D_x\mathcal{U}_y\right.\right. \\ &\left.\left.+D^2D_x\mathcal{V}_x + 2D^2D_y\mathcal{V}_y;\Phi^\eta_y\right\rangle\right) - A\int_{\Gamma_n}\left((3D^2D_x + D^2D_y)\Phi^\eta\frac{\partial\mathcal{U}}{\partial n} + (D^2D_x + 3D^2D_y)\Phi^\eta\frac{\partial\mathcal{V}}{\partial n}\right)\partial\gamma.\end{aligned}$$

For the right-hand side of second equation in system (3), we have :

$$\begin{aligned}\mathcal{G}(\mathcal{E},\mathcal{U},\mathcal{V},\Phi^u) &= -\left\langle g\mathcal{E}_x + \mathcal{U}\mathcal{U}_x + \mathcal{V}\mathcal{V}_x + BgD\left[(D_x\mathcal{E}_x + D_y\mathcal{E}_y)_x + D_x(\mathcal{E}_{xx}+\mathcal{E}_{yy})\right] - BD\zeta_{xtt};\Phi^u\right\rangle \\ &= -\langle g\mathcal{E}_x + \mathcal{U}\mathcal{U}_x + \mathcal{V}\mathcal{V}_x + Bg(DD_{xx}\mathcal{E}_x + DD_{xy}\mathcal{E}_y) - BD\zeta_{xtt};\Phi^u\rangle - Bg\langle 2DD_x\mathcal{E}_{xx} + DD_y\mathcal{E}_{xy} + DD_x\mathcal{E}_{yy};\Phi^u\rangle \\ &= -\langle g\mathcal{E}_x + \mathcal{U}\mathcal{U}_x + \mathcal{V}\mathcal{V}_x + Bg(DD_{xx}\mathcal{E}_x + DD_{xy}\mathcal{E}_y) - BD\zeta_{xtt};\Phi^u\rangle + Bg\langle 2DD_x\mathcal{E}_x;\Phi^u_x\rangle + Bg\left\langle (2D_x^2 + 2DD_{xx})\mathcal{E}_x;\Phi^u\right\rangle \\ &+Bg\left\langle DD_y\mathcal{E}_x + DD_x\mathcal{E}_y;\Phi^u_y\right\rangle + Bg\left\langle (D_y^2 + DD_{yy})\mathcal{E}_x + (D_xD_y + DD_{xy})\mathcal{E}_y;\Phi^u\right\rangle - \int_{\Gamma_n} Bg(3DD_x + DD_y)\Phi^u\frac{\partial\mathcal{E}}{\partial n}\partial\gamma \\ &= -\left\langle g\left(I_d - B\left(DD_{xx} + 2D_x^2 + DD_{yy} + D_y^2\right)\right)\mathcal{E}_x + \mathcal{U}\mathcal{U}_x + \mathcal{V}\mathcal{V}_x - BgD_xD_y\mathcal{E}_y - BD\zeta_{xtt};\Phi^u\right\rangle + Bg\langle 2DD_x\mathcal{E}_x;\Phi^u_x\rangle \\ &+Bg\left\langle DD_y\mathcal{E}_x + DD_x\mathcal{E}_y;\Phi^u_y\right\rangle - \int_{\Gamma_n} Bg(3DD_x + DD_y)\Phi^u\frac{\partial\mathcal{E}}{\partial n}\partial\gamma.\end{aligned}$$

Finally, for the right-hand side of the third equation in system (3), we have :

$$\begin{aligned}\mathcal{H}(\mathcal{E},\mathcal{U},\mathcal{V},\Phi^v) &= -\left\langle g\mathcal{E}_y + \mathcal{U}\mathcal{U}_y + \mathcal{V}\mathcal{V}_y + BgD\left[(D_x\mathcal{E}_x + D_y\mathcal{E}_y)_y + D_y(\mathcal{E}_{xx}+\mathcal{E}_{yy})\right] - BD\zeta_{ytt};\Phi^v\right\rangle \\ &= -\langle g\mathcal{E}_y + \mathcal{U}\mathcal{U}_y + \mathcal{V}\mathcal{V}_y + Bg(DD_{xy}\mathcal{E}_x + DD_{yy}\mathcal{E}_y) - BD\zeta_{ytt};\Phi^v\rangle - Bg\langle DD_y\mathcal{E}_{xx} + DD_x\mathcal{E}_{xy} + 2DD_y\mathcal{E}_{yy});\Phi^v\rangle \\ &= -\langle g\mathcal{E}_y + \mathcal{U}\mathcal{U}_y + \mathcal{V}\mathcal{V}_y + Bg(DD_{xy}\mathcal{E}_x + DD_{yy}\mathcal{E}_y) - BD\zeta_{ytt};\Phi^v\rangle + Bg\langle DD_y\mathcal{E}_x;\Phi^v_x\rangle + Bg\langle (D_xD_y + DD_{xy})\mathcal{E}_x;\Phi^v\rangle \\ &+Bg\left\langle DD_x\mathcal{E}_x + 2DD_y\mathcal{E}_y;\Phi^v_y\right\rangle + Bg\left\langle (D_xD_y + DD_{xy})\mathcal{E}_x + (2D_y^2 + 2DD_{yy})\mathcal{E}_y;\Phi^v\right\rangle - \int_{\Gamma_n} Bg(DD_x + 3DD_y)\Phi^v\frac{\partial\mathcal{E}}{\partial n}\partial\gamma \\ &= -\left\langle -Bg(2D_xD_y + DD_{xy})\mathcal{E}_x + \mathcal{U}\mathcal{U}_y + \mathcal{V}\mathcal{V}_y + g\left(I_d - B\left(DD_{yy} - 2D_y^2\right)\right)\mathcal{E}_y - BD\zeta_{ytt};\Phi^v\right\rangle + Bg\langle DD_y\mathcal{E}_x;\Phi^v_x\rangle \\ &+Bg\left\langle DD_x\mathcal{E}_x + 2DD_y\mathcal{E}_y;\Phi^v_y\right\rangle - \int_{\Gamma_n} Bg(DD_x + 3DD_y)\Phi^v\frac{\partial\mathcal{E}}{\partial n}\partial\gamma.\end{aligned}$$

Remark 2. We note that, during the simulation, when there is a steep gradient, we obtain a blow-ups. In order to avoid this problem, we need to change the bottom (make it smoother) or/and to get rid of the high order derivatives for the bottom as in [9]. That's why, we take into account in the sequel, the smoothness of the bottom and the fact that the derivatives of the bottom of order greater then one are neglected.

Thus, we will now deal with the following system:

$$\left\{\begin{aligned} \left\langle \partial_t\mathcal{E};\Phi^\eta\right\rangle + b\left\langle D^2\nabla\partial_t\mathcal{E};\nabla(\Phi^\eta)\right\rangle - \int_{\Gamma_n} bD^2\Phi^\eta\frac{\partial(\partial_t\mathcal{E})}{\partial n}\partial\gamma &= \mathcal{F}(\mathcal{E},\mathcal{U},\mathcal{V},\Phi^\eta) \\ \left\langle \partial_t\mathcal{U};\Phi^u\right\rangle + d\left\langle D^2\nabla\partial_t\mathcal{U};\nabla\Phi^u\right\rangle + d\langle 2D\nabla D\cdot\nabla\partial_t\mathcal{U};\Phi^u\rangle - \int_{\Gamma_n} dD^2\Phi^u\frac{\partial(\partial_t\mathcal{U})}{\partial n}\partial\gamma &= \mathcal{G}(\mathcal{E},\mathcal{U},\mathcal{V},\Phi^u) \\ \left\langle \partial_t\mathcal{V};\Phi^v\right\rangle + d\left\langle D^2\nabla\partial_t\mathcal{V};\nabla\Phi^v\right\rangle + d\langle 2D\nabla D\cdot\nabla\partial_t\mathcal{V};\Phi^v\rangle - \int_{\Gamma_n} dD^2\Phi^v\frac{\partial(\partial_t\mathcal{V})}{\partial n}\partial\gamma &= \mathcal{H}(\mathcal{E},\mathcal{U},\mathcal{V},\Phi^v) \end{aligned}\right. \quad (4)$$

with

$$\begin{aligned}\mathcal{F}(\mathcal{E},\mathcal{U},\mathcal{V},\Phi^\eta) = &-\langle (D+\zeta+\mathcal{E})\nabla\cdot(\mathcal{U};\mathcal{V}) + (D_x+\zeta_x+\mathcal{E}_x)\mathcal{U} + (D_y+\zeta_y+\mathcal{E}_y)\mathcal{V} + \zeta_t;\Phi^\eta\rangle \\ &-\tilde{A}\langle 2DD_x\zeta_{xt} + 2DD_y\zeta_{yt};\Phi^\eta\rangle - A\left\langle -2DD_xD_y\mathcal{V}_x + 2DD_x^2\mathcal{V}_y;\Phi^\eta\right\rangle + A\left(\left\langle 2D^2D_x\mathcal{U}_x + D^2D_y\mathcal{V}_x;\Phi^\eta_x\right\rangle + \left\langle D^2D_y\mathcal{U}_x\right.\right. \\ &\left.\left.+D^2D_x\mathcal{U}_y + D^2D_x\mathcal{V}_x + 2D^2D_y\mathcal{V}_y;\Phi^\eta_y\right\rangle\right) - A\int_{\Gamma_n}\left((3D^2D_x + D^2D_y)\Phi^\eta\frac{\partial\mathcal{U}}{\partial n} + (D^2D_x + 3D^2D_y)\Phi^\eta\frac{\partial\mathcal{V}}{\partial n}\right)\partial\gamma,\end{aligned}$$

$$\begin{aligned}\mathcal{G}(\mathcal{E},\mathcal{U},\mathcal{V},\Phi^u) = &-\left\langle g\left(I_d - B\left(2D_x^2 + D_y^2\right)\right)\mathcal{E}_x + \mathcal{U}\mathcal{U}_x + \mathcal{V}\mathcal{V}_x - BgD_xD_y\mathcal{E}_y - BD\zeta_{xtt};\Phi^u\right\rangle + Bg\langle 2DD_x\mathcal{E}_x;\Phi^u_x\rangle \\ &+Bg\left\langle DD_y\mathcal{E}_x + DD_x\mathcal{E}_y;\Phi^u_y\right\rangle - \int_{\Gamma_n} Bg(3DD_x + DD_y)\Phi^u\frac{\partial\mathcal{E}}{\partial n}\partial\gamma,\end{aligned}$$

and

$$\begin{aligned}\mathcal{H}(\mathcal{E},\mathcal{U},\mathcal{V},\Phi^v) = &-\left\langle -2BgD_xD_y\mathcal{E}_x + \mathcal{U}\mathcal{U}_y + \mathcal{V}\mathcal{V}_y + g\left(I_d - 2BD_y^2\right)\mathcal{E}_y - BD\zeta_{ytt};\Phi^v\right\rangle + Bg\langle DD_y\mathcal{E}_x;\Phi^v_x\rangle \\ &+Bg\left\langle DD_x\mathcal{E}_x + 2DD_y\mathcal{E}_y;\Phi^v_y\right\rangle - \int_{\Gamma_n} Bg(DD_x + 3DD_y)\Phi^v\frac{\partial\mathcal{E}}{\partial n}\partial\gamma.\end{aligned}$$

2.2 Time marching scheme

Our method is based on an explicit second order Runge-Kutta scheme. For that, let us denote by $(\mathcal{E}^{n+1},\mathcal{U}^{n+1},\mathcal{V}^{n+1})$ and $(\mathcal{E}^{n},\mathcal{U}^{n},\mathcal{V}^{n})$ the approximate values at time $t = t^{n+1}$ and $t = t^n$, respectively and by δt the time step size. Then, owing to (4), the unknown fields at time $t = t^{n+1}$ are defined as the solution of the following system:

$$\begin{cases} \langle \mathcal{E}^{n+1};\Phi^\eta\rangle = \langle \mathcal{E}^{n} + \dfrac{\mathcal{E}^{k1}+\mathcal{E}^{k2}}{2};\Phi^\eta\rangle, \\ \langle \mathcal{U}^{n+1};\Phi^u\rangle = \langle \mathcal{U}^{n} + \dfrac{\mathcal{U}^{k1}+\mathcal{U}^{k2}}{2};\Phi^u\rangle, \\ \langle \mathcal{V}^{n+1};\Phi^v\rangle = \langle \mathcal{V}^{n} + \dfrac{\mathcal{V}^{k1}+\mathcal{V}^{k2}}{2};\Phi^v\rangle, \end{cases} \tag{5}$$

where

$$\begin{aligned} \left\langle \mathcal{E}^{k1};\Phi^\eta\right\rangle + b\left\langle D^2\nabla\mathcal{E}^{k1};\nabla(\Phi^\eta)\right\rangle - \int_{\Gamma_n} bD^2\Phi^\eta\frac{\partial(\mathcal{E}^{k1})}{\partial n}\partial\gamma &= \delta t\cdot\mathcal{F}\left(\mathcal{E}^n,\mathcal{U}^n,\mathcal{V}^n,\Phi^\eta\right), \\ \left\langle \mathcal{U}^{k1} + 2dD\nabla D\cdot\nabla\mathcal{U}^{k1};\Phi^u\right\rangle + d\left\langle D^2\nabla\mathcal{U}^{k1};\nabla\Phi^u\right\rangle - \int_{\Gamma_n} dD^2\Phi^u\frac{\partial(\mathcal{U}^{k1})}{\partial n}\partial\gamma &= \delta t\cdot\mathcal{G}\left(\mathcal{E}^n,\mathcal{U}^n,\mathcal{V}^n,\Phi^u\right), \\ \left\langle \mathcal{V}^{k1} + 2dD\nabla D\cdot\nabla\mathcal{V}^{k1};\Phi^v\right\rangle + d\left\langle D^2\nabla\mathcal{V}^{k1};\nabla\Phi^v\right\rangle - \int_{\Gamma_n} dD^2\Phi^v\frac{\partial(\mathcal{V}^{k1})}{\partial n}\partial\gamma &= \delta t\cdot\mathcal{H}\left(\mathcal{E}^n,\mathcal{U}^n,\mathcal{V}^n,\Phi^v\right) \end{aligned} \tag{6}$$

and

$$\begin{aligned} \left\langle \mathcal{E}^{k2};\Phi^\eta\right\rangle + b\left\langle D^2\nabla\mathcal{E}^{k2};\nabla(\Phi^\eta)\right\rangle - \int_{\Gamma_n} bD^2\Phi^\eta\frac{\partial(\mathcal{E}^{k2})}{\partial n}\partial\gamma &= \delta t\cdot\mathcal{F}\left(\mathcal{E}^n+\mathcal{E}^{k1},\mathcal{U}^n+\mathcal{U}^{k1},\mathcal{V}^n+\mathcal{V}^{k1},\Phi^\eta\right), \\ \left\langle \mathcal{U}^{k2} + 2dD\nabla D\cdot\nabla\mathcal{U}^{k2};\Phi^u\right\rangle + d\left\langle D^2\nabla\mathcal{U}^{k2};\nabla\Phi^u\right\rangle - \int_{\Gamma_n} dD^2\Phi^u\frac{\partial(\mathcal{U}^{k2})}{\partial n}\partial\gamma &= \delta t\cdot\mathcal{G}\left(\mathcal{E}^n+\mathcal{E}^{k1},\mathcal{U}^n+\mathcal{U}^{k1},\mathcal{V}^n+\mathcal{V}^{k1},\Phi^u\right), \\ \left\langle \mathcal{V}^{k2} + 2dD\nabla D\cdot\nabla\mathcal{V}^{k2};\Phi^v\right\rangle + d\left\langle D^2\nabla\mathcal{V}^{k2};\nabla\Phi^v\right\rangle - \int_{\Gamma_n} dD^2\Phi^v\frac{\partial(\mathcal{V}^{k2})}{\partial n}\partial\gamma &= \delta t\cdot\mathcal{H}\left(\mathcal{E}^n+\mathcal{E}^{k1},\mathcal{U}^n+\mathcal{U}^{k1},\mathcal{V}^n+\mathcal{V}^{k1},\Phi^v\right). \end{aligned} \tag{7}$$

3 Mesh generation and initial data

In this section, we present the technique used in order to generate a mesh using a photo of the Mediterranean sea then using an imported xyz bathymetry. Also, we will present a way in order to obtain the initial data and explain the special adapt mesh technique that we will use in our numerical simulation.

3.1 Mesh generated using a photo

We present here the method to build a mesh from a photo inspired from a FreeFem++ script made by F. Hecht [7] and another one made by O. Pantz [12].

Owing to Google Earth® and for better resolution, we take severals parts of the Mediterranean sea (cf. Figure 2) that are subsequently assembled using Photoshop® to obtain a complete picture of the Mediterranean sea (cf. Figure 3).

Figure 2 – The pictures of two parts of the Mediterranean sea.

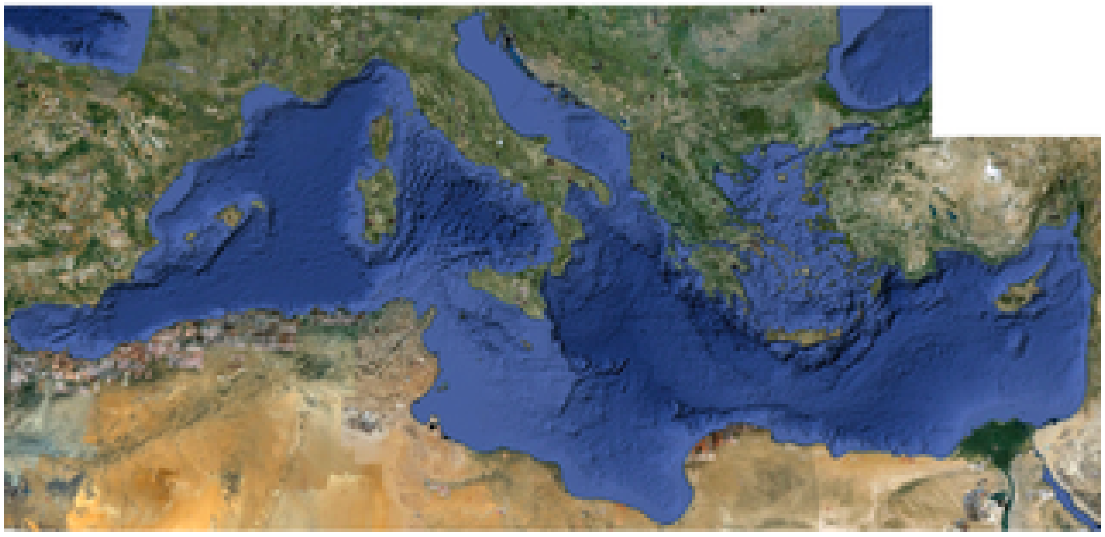


Figure 3 – The Mediterranean sea after assembly with Photoshop®.

Using Photoshop®, we can also eliminate the land areas that circumvent the Mediterranean sea (cf. Figure 4). We note that we must smoothen the borders in Photoshop® and put the black color inside our domain and the white color outside.

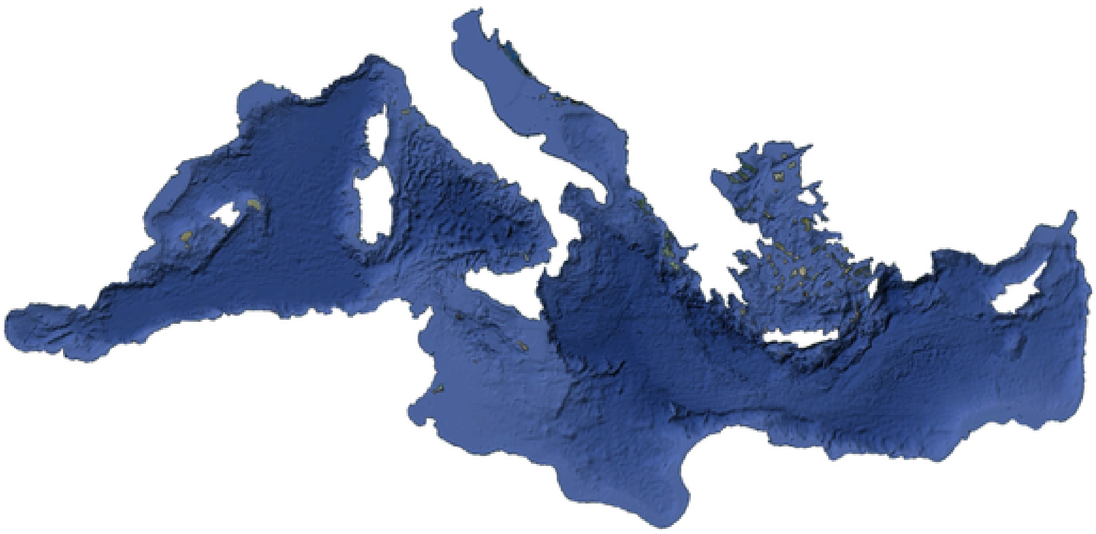


Figure 4 – The Mediterranean sea.

Then we convert the jpg photo to a pgm photo which can be read by FreeFem++ using in a terminal window :

```
convert Medit_sea.jpg Medit_sea.pgm
```

In order to generate the mesh of the Mediterranean sea domain, we read the pgm file using

```
load "ppm2rnm"
load "isoline"
real[int,int] f1("Medit_sea.pgm");
int nx = f1.n, ny=f1.m;
mesh Sh=square(nx-1,ny-1,[(nx-1)*(x),(ny-1)*(1-y)]);
fespace Vh(Sh,P1);  Vh fxy;
fxy[]=f1;
fxy=(fxy>=0.5)-(fxy<0.5);// to get value between -1 and 1
real[int,int] Curves(3,1);
int[int] be(1);
```

```
int nc=isoline(Sh,fxy,iso=0.,close=0,Curves,beginend=be,smoothing=.005,ratio
    ➥=0.1);
```

The function obtained from the pgm file has values between 0 and 1, where the value of 0.5 represents the contour between two different colors. We note that, we can regularize this contour (thanks to O. Pantz [12]), before using the isoline function which computes the number of closed curves of our image, by solving :

$$\int_\Omega \left(2\varepsilon\nabla(\delta u)\nabla(\phi)+\frac{4}{\varepsilon}u^2\delta u\phi+\alpha\delta u\phi\right)+\int_\Omega\left(\varepsilon\nabla(u)\nabla(\phi)-\frac{2}{\varepsilon}(1-u^2)u\phi\right)=\int_\Omega \alpha(fxy-u)\phi, \tag{8}$$

where in our example, we take $\varepsilon = 1$, $\alpha = 1$ and $\delta x = 1.5$. We also note that when ε is close to zero, the solution u takes the values equal to 1 and -1 and α sets the balance between length of the curve and fitting the actual interface: As α increase, the approximation become better.
After regularizing, we update $u = 0$ till $u = u + \delta u$, and we adapt the mesh around the contour (cf. Figure 5), by using

```
Sh=adaptmesh(Sh,u,nbvx=1e7);
```

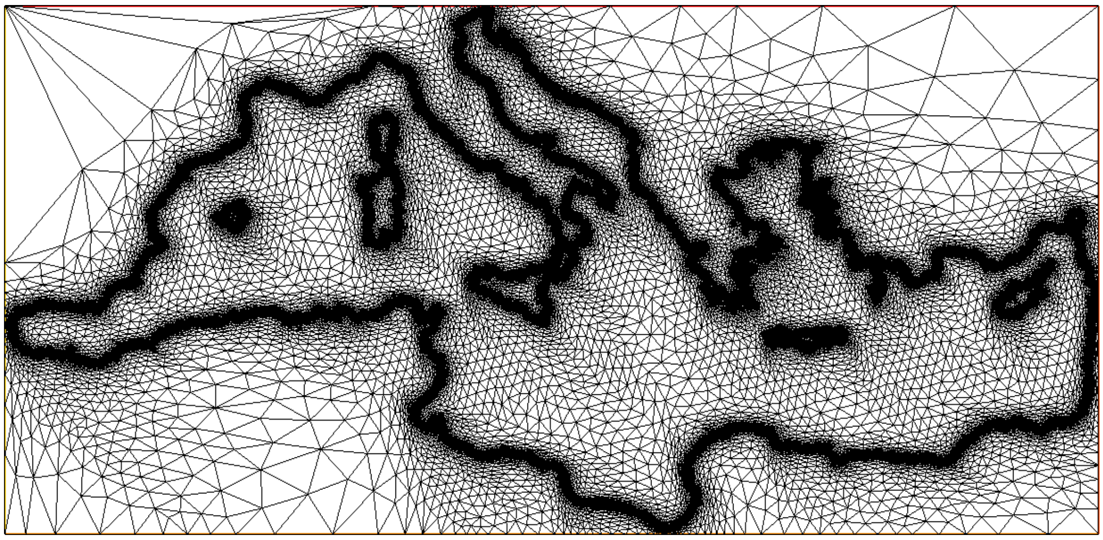


Figure 5 – The adapted mesh around the contour of the Mediterranean sea.

then, we solve again the regularizing problem, then we update the function $u = u + \delta u$ and finally we interpolate the solution on the initial mesh (cf. Figure 6) before using isoline :

```
mesh Shinit=Sh; fespace Vhinit(Shinit,P1);
Vhinit fxyinit;
fxyinit=u;
int nc=isoline(Shinit,fxyinit,iso=0.,close=0,Curves,beginend=be,smoothing
    ➥=.005,ratio=0.1);
```

Now, we show in Figure 7, the mesh created around Crete island, where we see in the top left, the mesh after first regularization and in the top right, the function fxy after interpolation, in the down left, three isoline level -1, 0 and 1 of the fxy function, where here the mesh generated at the 0 level is shown in the down right. The complete script is written in [15].
We note that, we take into account the area of the Mediterranean sea, which is almost 2.5 million km^2 and we did a scale for our final mesh :

```
real Areasea=2.5e6;//area of the Mediterranean sea in Km^2
real scale=sqrt(Areasea/Th.area);
Th=movemesh(Th,[x*scale,y*scale]);
```

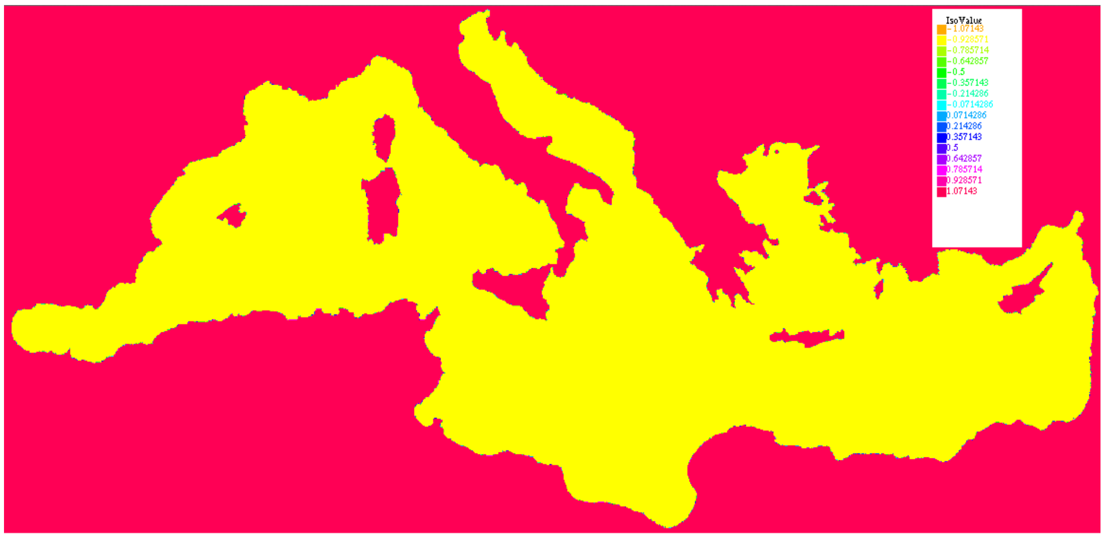


Figure 6 – The function fxy of the contour of Mediterranean sea after interpolation.

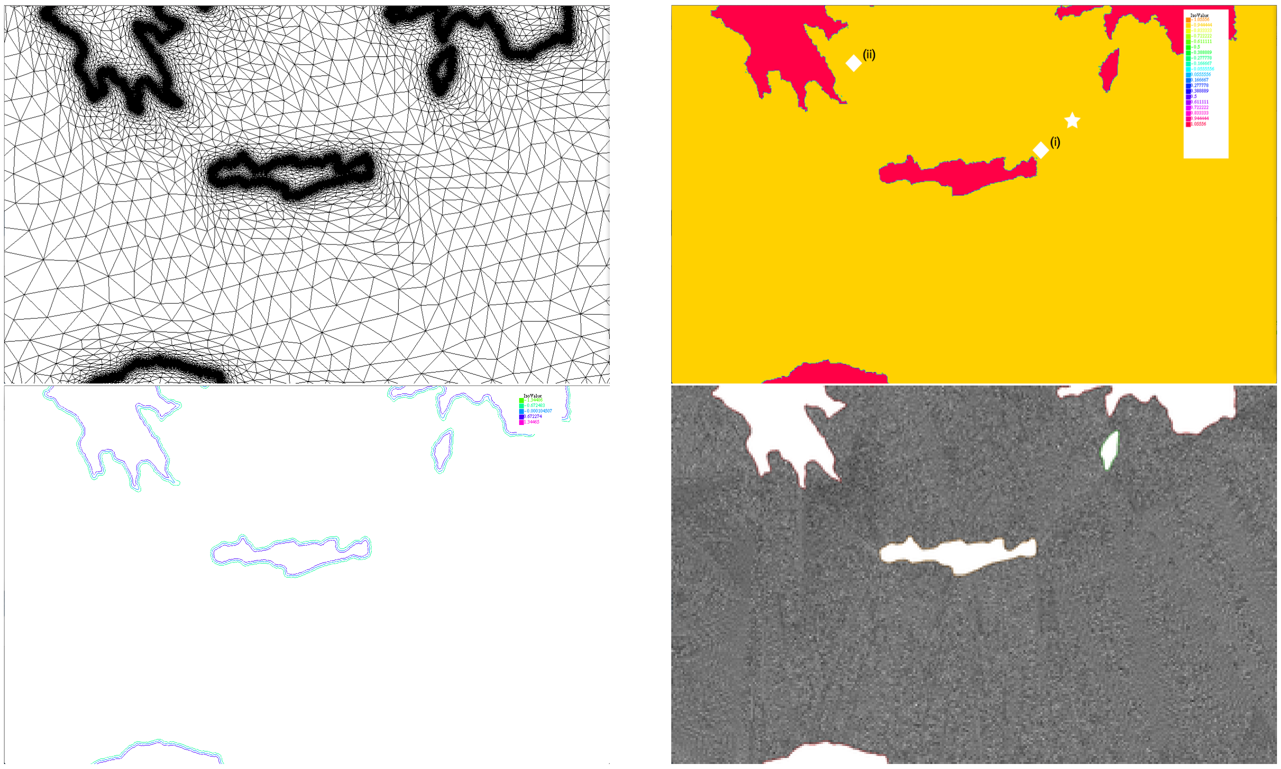


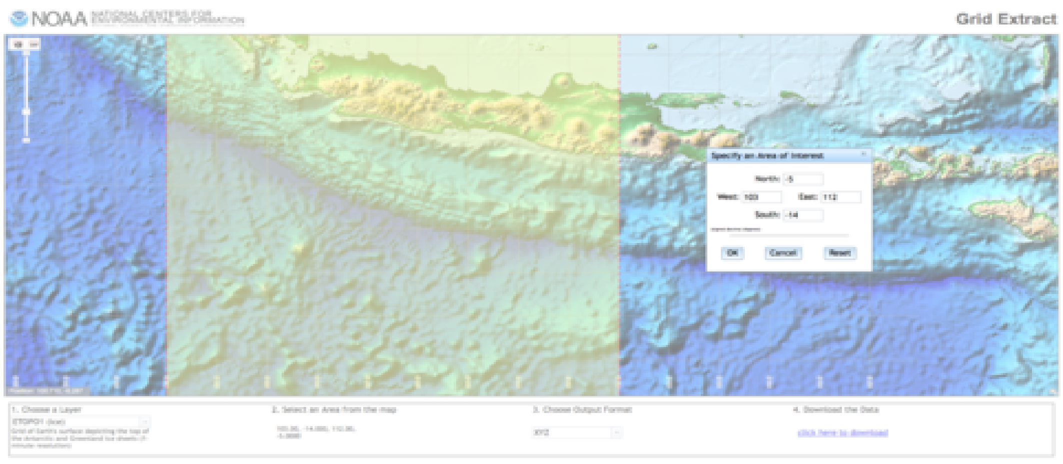
Figure 7 – Creation of the mesh around Crete island in the Mediterranean sea (top left: mesh after first regularization, top right: solution after interpolation, $\diamond$: wave gauge, $\star$: epicenter, down left: isoline of the contour, down right: mesh generated).

3.2 Mesh generated using an imported xyz bathymetry

In order to consider more realistic case, this means that, we now take into account the bathymetry near Java island which can be downloaded from the NOAA [3] website, (cf. Figure 8), we also use FreeFem++ to generate the mesh of the area where the amplitude is zero.

We can read the xyz file (cf. Figure 9, left), using this script :

3. https://maps.ngdc.noaa.gov/viewers/wcs-client/


NOAA National Centers for Environmental Information
Grid Extract
Specify an Area of Interest
OK
Cancel
Reset
1. Choose a Layer
2. Select an Area from the map
3. Choose Output Format
4. Download the Data
click here to download

North: -5
West: 103 East: 112
South: -14

ETOPO1 (ice)

XYZ

Figure 8 – Importation of bathymetry datas through the NOAA website.

```
mesh Sh=triangulate("bathymetry_Java");
fespace Vh(Sh,P1);  Vh fxy;
{ ifstream file(filename);real xx,yy;
    for(int i=0;i<fxy.n;i++)
        file >> xx >>yy >> fxy[][i];
}
```

We can smoothen the bathymetric data by solving :

$$\int_\Omega (\beta u\phi + \nabla(u)\nabla(\phi)) + \int_\Gamma \frac{\partial u}{\partial n}\partial\gamma = \int_\Omega \beta fxy\phi, \qquad \frac{\partial u}{\partial n} = 0. \tag{9}$$

In our code, we take $\beta = 5.e3$, in order to build the mesh to get rid of smallest islands, then, in order to obtain the mesh only around the sea, which is limited by the zero level of the amplitude, we use :

```
fxy=(fxy>=0.5)-(fxy<0.5);// to get value between -1 and 1
```

and then proceeding similarly, as above for the mesh generation using a photo, in order to obtain the mesh of our domain, by taking $\varepsilon = 1.e-2$ and $\alpha = 1.e2$ for the regularization phenomena (8).

Remark 3. We note that our method takes into account the different label for each part of the boundary, which facilitates the use of different types of boundary condition.

Remark 4. For all simulation with bathymetry, we use $\beta = 2.e1$ in (9) to smoothen the initial bathymetry after generation of the mesh (cf. Figure 9, right) in order to ensure the stability of the numerical method, we also note that in order to be in a big deep water wave regime for sBBM system we change the depth close to the shoreline to 100 m.

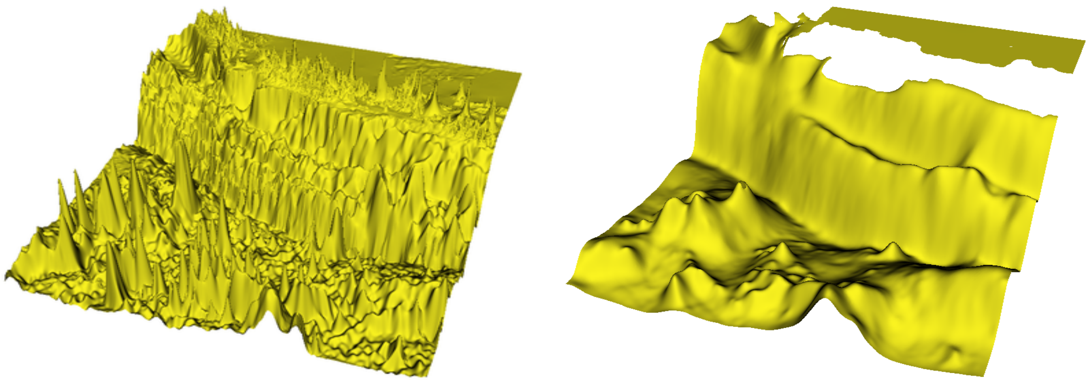


Figure 9 – Left : Bathymetry downloaded from the NOAA website, (min $= -7239m$ and max $= 3002m$). Right : smoothed bathymetry with $\beta = 2.e1$ in (9), (min $= -6207m$ and max $= -100m$).

Remark 5. The bathymetry data downloaded from the NOAA website are in degree coordinate and we need to convert them to meter. So, on the first hand, we must know the degree of Latitude (south and north) and of Longitude (west and east) of our domain where we can deduce the Latitude $lat0 = .5(lat_{south} + lat_{north})$ and the Longitude $long0 = .5(long_{west} + long_{east})$. On the other hand, we must take into account the spherical shape of the Earth, even if it does not play significant role because of the small spatial scale of the experiments. So, we know that the radius of the Earth near the equator is $R_{equator} = 6378,137$ Km, and near to the pole $R_{pole} = 6356,752$ Km, thus the radius of our domain equals to:

$$R = \sqrt{\frac{\left(R_{equator}^2 \cos(lat0 \cdot \pi/180)\right)^2 + \left(R_{pole}^2 \sin(lat0 \cdot \pi/180)\right)^2}{\left(R_{equator} \cos(lat0 \cdot \pi/180)\right)^2 + \left(R_{pole} \sin(lat0 \cdot \pi/180)\right)^2}}.$$

So, we move the mesh of our domain using the following translation (coefl0 $= \pi R/180$):

$$[x; y] \longrightarrow [(x - lon0)\cos(\pi y/180)\text{coefl0}; (y - lat0)\text{coefl0}].$$

Therefore, we need to move the bottom from the original reference downloaded to the new mesh, which is easy to do in FreeFem++ by writing this script :

```
Vh D=fxy;
real coefl0=pi*R/180.;
Sh=movemesh(Sh,[(x-lon0)*cos(y*pi/180.)*coefl0,(y-lat0)*coefl0]);
Vh tmp; // define a temporary function
tmp=D[];     // save the value
D=0;         // to change the FEspace and mesh associated with u
D[]=tmp;     // set the value of u without any mesh update
```

3.3 Mesh adaptation technique

We introduce here a special mesh adaptation technique since some computation domains are huge as in the case of the Mediterranean sea with $1.7e6$ triangles ($8.7e5$ degree of freedom) and our initial solution is concentrated in a small domain, a circle $\mathcal{C}(O, R)$ or a rectangle $[aa, bb] \times [cc, dd]$. So we build the mesh `Th` of the small rectangle or the circle by doing a trunk to the initial full mesh `Thinit` respecting that the function equals to 1 inside the rectangle and 0 outside as in :

```
Th=trunc(Thinit,(1.*(x <= bb & x >= aa) *(y <= dd & y >= cc))>.5,label=0);
```

or

```
Th=trunc(Thinit,(1.*((x -x0)^2+(y-y0)^2<=R^2))>.5,label=0);
```

then we compute the initial solution `uold=uadapt` in this domain. Using the keyword `boundingbox` in FreeFem++, we obtain the limit min max of `Thold=Th` on x and y direction, in which we add `epsadapt` from each side in order to build the new rectangle `Thnew` that contains `Thold`, then using the keyword `interpolate` in FreeFem++, the old FEspace in $\mathbb{P}_1$ `Vhold` and the new FEspace in $\mathbb{P}_1$ `Vhnew`, we interpolate `uold` to `unew` in the new domain. We smooth the function obtained from `abs(unew)>=erradapt` using :

$$\int_\Omega (\texttt{smoothadapt} \cdot \texttt{usmadapt} \cdot \phi + \nabla(\texttt{usmadapt})\nabla(\phi)) = \int_\Omega \texttt{smoothadapt} \cdot (|\texttt{unew}| \geq \texttt{erradapt}) \cdot \phi, \qquad (10)$$

with zero Dirichlet boundary condition on the boundary label 0 of `Thnew` in order to obtain `usmadapt`. And, at the end, we trunk `Thnew` respecting the function `usmadapt>=isoadapt`. We can put all the previous detail for the special mesh adaption in a `macro` such as :

```
macro AdaptGS(Th,uadapt,erradapt,isoadapt,smoothadapt,epsadapt,Thinit,output
   ➥,wv)
     mesh Thold=Th;
     fespace Vhold(Thold,P1);
     Vhold uold=uadapt;
     real[int] bbmM(4);
     boundingbox(Thold,bbmM);
     mesh Thnew=trunc(Thinit,(1.*(x <= (bbmM[1]+epsadapt) & x >= (bbmM[0]-
        ➥epsadapt)) *(y <= (bbmM[3]+epsadapt) & y >= (bbmM[2]-epsadapt)))>
        ➥erradapt,label=0);
```

```
    fespace Vhnew(Thnew,P1);
    Vhnew usmadapt,vsmadapt,unew;
    matrix BSinterp=interpolate(Vhnew,Vhold,inside=true);
    unew[]=BSinterp*uold[];
    solve smoother(usmadapt,vsmadapt)=int2d(Thnew)(smoothadapt*usmadapt*
       ➥vsmadapt+grad(usmadapt)'*grad(vsmadapt))-int2d(Thnew)(smoothadapt*(
       ➥abs(unew)>=erradapt)*vsmadapt)+on(0,usmadapt=0.);
    if(output) plot(usmadapt,fill=true,dim=2,value=true,wait=wv,cmm="
       ➥usmadapt");
    if(output) {Vhnew unewp=(usmadapt>=isoadapt); plot(unewp,fill=true,dim
       ➥=2,value=true,wait=wv,cmm="zone to cut");}
    Th=trunc(Thnew,usmadapt>=isoadapt,label=0);
//
```

Remark 6. We trunk always from the initial full mesh, in this case, we keep the original vertices of the mesh throughout the simulation, and also we keep the original label of the boundary and we put 0 for the label of the rest of the boundary domain. We also note that we need to interpolate all the variables of any kind of FEspace from the old mesh to the newest one using the keyword `interpolate` in FreeFem++. We remark also that we must choose the parameters for the `AdaptGS` in order to obtain the value of `erradapt` on the boundary of `Th` for the function `uadapt`. In addition, we use a reflective boundary condition (BC) on label 0, *i.e.* zero Neumann BC for η and zero Dirichlet BC for $\mathcal{V}$, cause our sBBM system gives artificial numerical explosion on the boundary if we do not use any BC or if we use only Neumann BC for η and $\mathcal{V}$.

3.4 Initial data

Inspiring from [4, 9], *Tsunami* waves are generated by a deformation of the bottom due to an Earthquake, which may be approximated by Okada's formula's [10, 11] in the case of the so called dip-slip dislocation, where the vertical component of displacement vector $\mathcal{O}(x, y)$, is given by the following formulas in Chinnery's notation, cf [2, 10]

$$f(\xi,\eta)\,||= f(\xi,p) - f(\xi,p-W) - f(\xi-L,p) + f(\xi-L,p-W)\ ,$$

$$\mathcal{O}(x,y) = -\frac{U}{2\pi}\left(\frac{\tilde{d}q}{R(R+\xi)} + \sin\delta\arctan\frac{\xi\eta}{qR} - I\sin\delta\cos\delta\right)\bigg|\bigg|.$$

where

$$\xi = (x-x_0)\cos\phi + (y-y_0)\sin\phi,\ \ Y = -(x-x_0)\sin\phi + (y-y_0)\cos\phi,$$
$$p = Y\cos\delta + d\sin\delta,\ \ q = Y\sin\delta - d\cos\delta,$$
$$\tilde{y} = \eta\cos\delta + q\sin\delta,\ \ \tilde{d} = \eta\sin\delta - q\cos\delta,$$
$$R^2 = \xi^2 + \eta^2 + q^2 = \xi^2 + \tilde{y}^2 + \tilde{d}^2,\ \ X^2 = \xi^2 + q^2$$

and

$$I = \begin{cases} \dfrac{\mu}{\lambda+\mu}\dfrac{2}{\cos\delta}\arctan\dfrac{\eta(X+q\cos\delta)+X(R+X)\sin\delta}{\xi(R+X)\cos\delta} & \text{if } \cos\delta \neq 0,\\ \dfrac{\mu}{\lambda+\mu}\dfrac{\xi\sin\delta}{R+\tilde{d}} & \text{if } \cos\delta = 0. \end{cases}$$

Here, W and L are the width and the length of the rectangular fault, (x, y) are the points where we computes displacements, (x_0, y_0) is the epicenter, d = fault depth$(x_0, y_0) + W\sin\delta$, δ is the dip angle, θ is the rake angle, D is the Burger's vector, $U = |D|\sin\theta$ is the slip on the fault, ϕ is the strike angle which is measured conventionally in the counter-clockwise direction from the North (cf. Figure 10 (left)), μ, λ are the Lamé constants derived from elastic-wave velocities : $\lambda = \rho_c\left(V_P^2 - V_S^2\right)$ and $\mu = \rho_c V_S^2$, where ρ_c is the crust density, V_P is the compressional-wave (P-wave) velocity, V_S is the shear-wave (S-wave) velocity.

Remark 7. We can download the script which computes co-seismic displacements according to the classical Okada solution $\mathcal{O}(x, y)$ from the following link `http://www.denys-dutykh.com/downloads.php`.

We will distinguish here the two cases for the mechanisms of the dynamics of *Tsunami* wave generation as in [9] : the *passive* generation and the *active* generation.

Passive **generation :**

In order to compute the initial data for $\eta(x, y, 0) = \mathcal{O}(x, y)$ in meters (cf. Figure 10 (right)), $V(x, y, 0) = 0$ which is referred to as a *passive* generation of a *Tsunami* wave near Java island, using our mesh adaptive technique, we will use the fact that the solution is concentrated in the small rectangle $[x_0 - 3.2W; x_0 + 1.2W] \times [y_0 - L; y_0 + L]$ where $L = 100$ Km, $W = 50$ Km, $\delta = 10.35°$, $\phi = 288.94°$, $\theta = 95°$, $U = 2$ m, $\rho_c = 2700$ Kg/m^3, $V_P = 6000$ m/s, $V_S = 3400$ m/s, $(x_0; y_0) = (107.345°, -9.295°)$ and the fault depth$(x_0; y_0) = 10$ Km. All these geophysical parameters can be downloaded from this website https://Earthquake.usgs.gov.

Therefore, we build the mesh of the small rectangle by doing a `trunc` to the initial full mesh respecting that the function equals to 1 inside the rectangle and 0 outside as in :

```
real aa=x0-3.2*W,bb=x0+1.2*W,cc=y0-L,dd=y0+L;
Th=trunc(Thinit,(1.*(x <= bb & x >= aa) *(y <= dd & y >= cc))>.5,label=0);
```

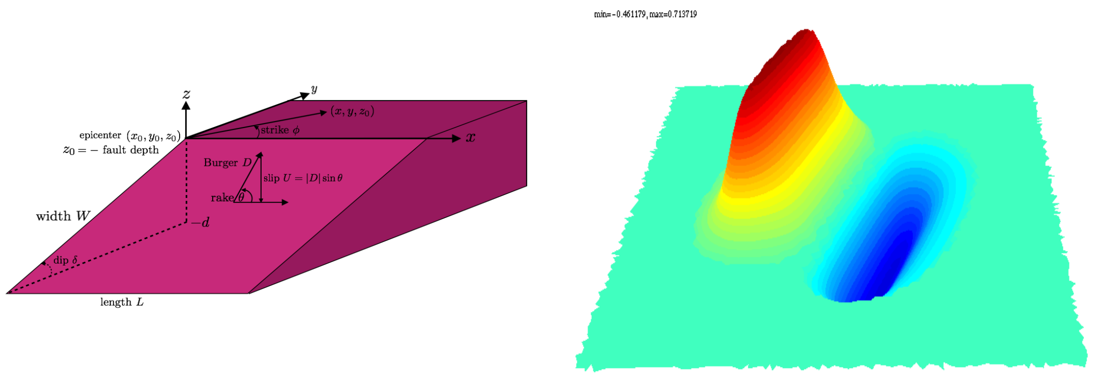


Figure 10 – Geometry of the source model (left) and the initial solution for η (right, min= -0.46 m, max= 0.73 m).

Active **generation :**

For a more realistic case as in the Java 2006 event, we use the *active* generation in order to model the generation of a *Tsunami* wave as in [4, 5]. In this case we consider zero initial conditions for both the free surface elevation and the velocity field, and assume that the bottom is moving in time. This case may be described by considering the bottom motion formula : $-h(x, y, t) = -D(x, y) - \zeta(x, y, t)$ with

$$\zeta(x, y, t) = \sum_{i=1}^{Nx \cdot Ny} \mathcal{H}(t - t_i) \cdot \left(1 - e^{-\alpha(t - t_i)}\right) \cdot \mathcal{O}_i(x, y),$$

where Nx sub-faults along strike and Ny sub-faults down the dip angle, $\mathcal{H}(t)$ is the Heaviside step function and $\alpha = \frac{\log(3)}{t_r}$, where $t_r = 8$ s is the rise time. We choose here an exponential scenario, but in practice, various scenarios could be used (instantaneous, linear, trigonometric, etc) and could be found in [4, 5, 6].

Remark 8. Parameters such as sub-fault location (x_i, y_i), depth d_i, slip U and rake angle θ for each segment, given in Table 3 of the paper [4], can be downloaded from this website https://Earthquake.usgs.gov.
In this file, we remark that the fault's plane is conventionally divided into $Nx = 21$ sub-faults along strike and $Ny = 7$ sub-faults down the dip angle, leading to a total number of $Nx \times Ny = 147$ equal segments.

For our special adapt mesh technique, since the fault plane is considered to be the rectangle with vertices located at (109.20508° (Lon),−10.37387° (Lat)), (106.50434° (Lon),−9.45925° (Lat)), (106.72382° (Lon),−8.82807° (Lat)) and (109.42455° (Lon),−9.74269° (Lat)), we will consider that our bottom displacement is concentrated on the big rectangle which is equidistant of 1° from each side of the initial fault plane as in Figure 11 (left), then we compute each Okada solution $\mathcal{O}_i$ on a circle of center $(x_i - 10m, y_i - 10m)$ and of radius $6 \max(L, W)$ and at then end all the Okada solution will be interpolated on the big rectangle before starting to compute the vertical displacement of the bottom $\zeta(x, y, t)$, in Figure 11 (right) we plot $\mathcal{O}_{14}$. For the computation of $\zeta(x, y, t)$, we start the mesh by a circle of center $(x_c - 5m, y_c - 5m)$ and of radius $4 \max(L, W)$ and we adapt the mesh each 3 iterations *i.e.* each 6 s by using the following value for the adapt mesh `uadapt`= ζ, `isoadapt`=5e-2, `erradapt`=1e-4, `smoothadapt`=5e-9, `epsadapt`=50e3.

We show in the Figure 12, the bottom displacement $\zeta(x, y, t)$ at time $t = 100$s and $t = 270$s using our adapt mesh technique.

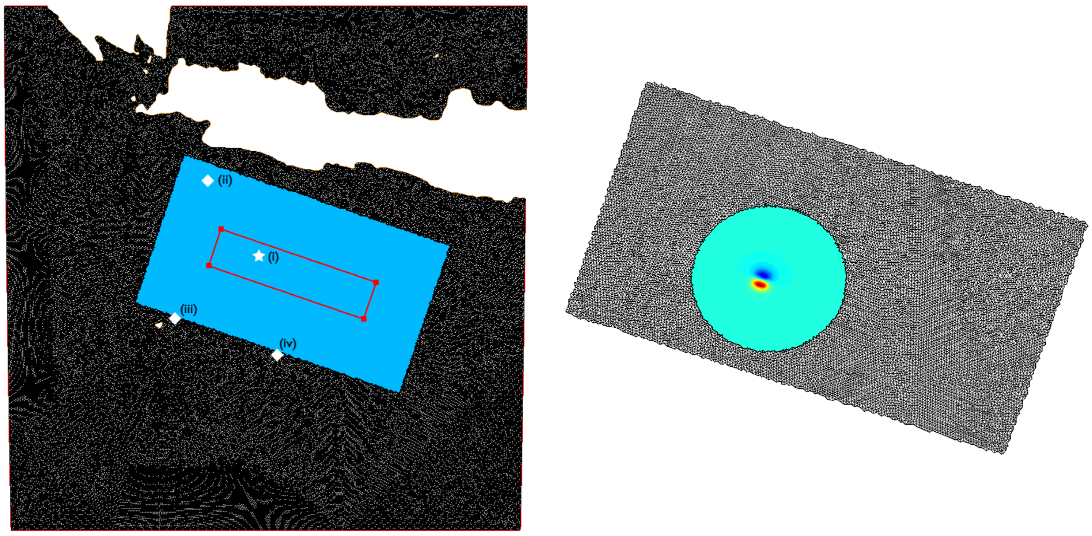


Figure 11 – Left : Surface projection of the fault's plane and the mesh around, $\diamond$: wave gauge, $\star$: epicenter. Right : the 14-th Okada solution (min= -0.09 m, max= 0.17 m).

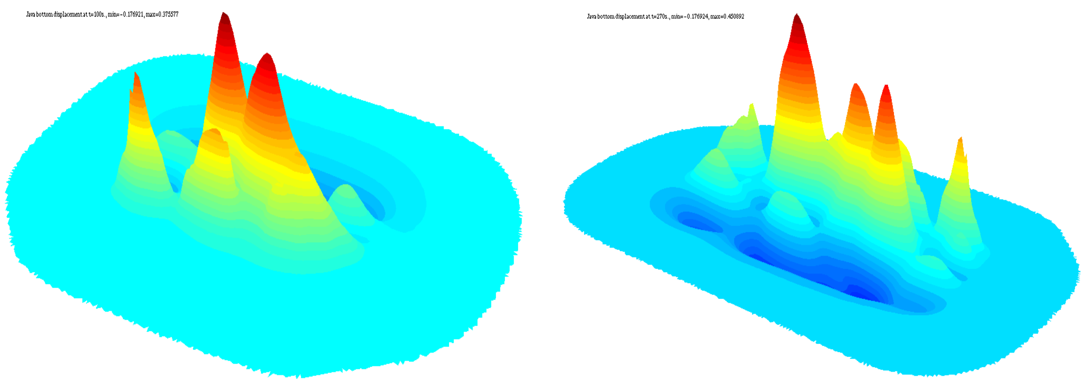

Java bottom displacement at t=100s., min=-0.176921, max=0.375577
Java bottom displacement at t=270s., min=-0.176924, max=0.450892

Figure 12 – Bottom displacement at $t = 100$ s (left, min= -0.18 m, max= 0.38 m) and at $t = 270$ s (right, min= -0.18 m, max= 0.45 m).

Remark 9. We note that after building the Okada solution $\mathcal{O}(x, y)$ in the *passive* generation or $\mathcal{O}_i(x, y)$ in the *Active* generation, we can remark that this solution is non-local and decays slowly to zero, that why in our adapt mesh technique we put 0 where the absolute value of the solution is less then $\min(|\min(\mathcal{O}_i(x, y))|, |\max(\mathcal{O}_i(x, y))|)/9.2$ meter, we make the same thing without adapt mesh in order to compare the solution using the same initial data.

4 Numerical simulations

In this section, we study first the rate of convergence of our codes for the sBBM (4) with non-dimensional and unscaled variable *i.e.*, with $g = 1$ over a variable bottom in space, which establishes the adequacy of the chosen finite element discretization and the used time marching scheme, for the flat bottom case, we refer to [13], where we use the same technique as in this paper. Then we simulate the propagation of a wave, that looks like a *Tsunami* wave generated by an Earthquake, in the Mediterranean sea over the sBBM with a flat bottom, near Java island over a variable bottom in space and at the end near Java island over a variable bottom in space and in time.

In all numerical simulations we used $\mathbb{P}_1$ continuous piecewise linear functions for η, u, v, D and ζ.

4.1 Rate of convergence

We prove in the figure below, that the RK2 time scheme considered for the sBBM variable bottom in space system is of order 2, we note that $\zeta(x, y, t)$ is only used for the generation of the *Tsunami* wave and will not be taken into account in the convergence rate test. In this example, we took Bi-Periodic Boundary Conditions for

η_h, u_h and v_h on the whole boundary of the square $[0, 2L] \times [0, 2L]$, where $L = 50$ and we consider the following exact solutions:

$$\eta_{ex} = .2\cos(2\pi x/L - t)\cos(2\pi y/L - t), u_{ex} = .5\sin(2\pi x/L - t)\cos(2\pi y/L - t),$$

$$v_{ex} = .5\cos(2\pi x/L - t)\sin(2\pi y/L - t), D(x, y) = 1 - .5\cos(2\pi x/L)\cos(2\pi y/L),$$

adding an appropriate right-hand side function.

We measure at time $T = 1$ and for $\theta^2 = \dfrac{2}{3}$, $\delta t = \dfrac{.01}{2^n}$ and $\delta x = \dfrac{2L}{N} = \dfrac{2L}{2^{n+5}}$ $\forall n \in \{0, 1, 2, 3, 4\}$, the following errors

$$N_{L^2}(\eta) = \|\eta_h - \eta_{ex}\|_{L^2}, N_{H^1}(\eta) = \|\eta_h - \eta_{ex}\|_{H^1}, N_{L^2}(\mathcal{V}) = \|u_h - u_{ex}\|_{L^2} + \|v_h - v_{ex}\|_{L^2}$$
$$N_{H^1}(\mathcal{V}) = \|u_h - u_{ex}\|_{H^1} + \|v_h - v_{ex}\|_{H^1}$$

and we end up with the following results:

N	δt	$N_{L^2}(\eta)$	rate	$N_{L^2}(\mathcal{V})$	rate	$N_{H^1}(\eta)$	rate	$N_{H^1}(\mathcal{V})$	rate
2^5	$.01/2^0$	0.241446	-	1.10773	-	0.603174	-	1.62575	-
2^6	$.01/2^1$	0.0607759	1.99013	0.280157	1.98329	0.301957	0.998228	0.812757	1.00021
2^7	$.01/2^2$	0.01524	1.99564	0.0703759	1.99308	0.151186	0.998017	0.406962	0.99793
2^8	$.01/2^3$	0.0038124	1.9987	0.017602	1.99909	0.075782	0.9975	0.203552	0.99965

Table 1 – L^2 norm of the error for η and $\mathcal{V}$.

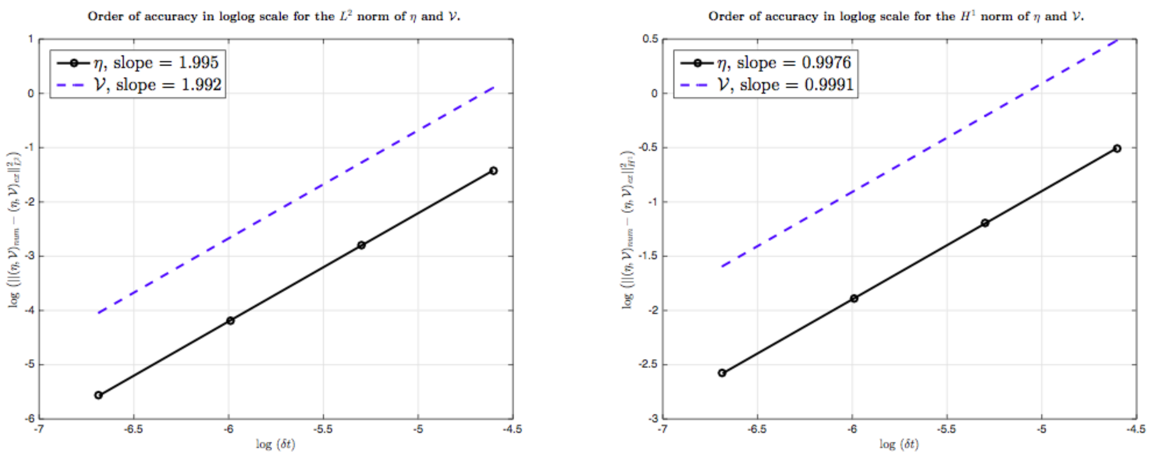


Figure 13 – Rate of convergence for sBBM system with variable bottom in space.

So, the $L^2(\Omega\times]0, T[)^2$ norm slope for η and $\mathcal{V}$ is of order ~ 2 and the $L^2(0, T; H^1(\Omega)^2)$ slope for η and $\mathcal{V}$ is of order ~ 1 as shown in the Figure 13 and which confirms the convergence of the second-order Runge-Kutta scheme in time for the sBBM system with variable bottom in space.

4.2 Propagation of a *Tsunami* wave in the Mediterranean sea with a flat bottom.

In this section, the non-dimensional and unscaled variables in (4) *i.e.* $g = 1$. We simulate here, the propagation of a wave that looks like a *Tsunami* wave generated by an Earthquake in the Mediterranean sea over the sBBM (4) with a flat bottom $-D(x, y) = -1,5$ Km which is the average depth of the Mediterranean sea. This wave was defined above in the *passive* generation part of the Section 3 where, in this case, the initial solution is concentrated in the small rectangle $[x_0 - 5W; x_0 + 4W] \times [y_0 - 1.5L; y_0 + 2.5L]$ and we take these following values : $L = 20$ Km, $W = 10$ Km, $\delta = 7°$, $\phi = 0°$, $\theta = 90°$, $E = 9,5$ GPA is the Young's modulus, $\nu = 0,27$ is the Poisson's ratio, $U = 2,5$ m, $(x_0; y_0) = (2390. * scale, 590. * scale)$ and the fault depth$(x_0; y_0) = 10$ Km. In this example, we will take the fact that the Lamé constants μ and λ are given by the formulas $\mu = E/2(1 + \nu)$ and $\lambda = E\nu/(1 + \nu)(1 - 2\nu)$.
We also use the following settings : for the step time $\delta t = 0.1 = 1$ s, a reflective BC for all the boundary, for the `adaptmesh` of FreeFem++:

```
fespace Vhinit(Thinit,P0);
Vhinit hT=hTriangle;
real Dx=hT[].min;
uadapt=eta0+u0+v0;
Th=adaptmesh(Th,uadapt,err=1.e-6,errg=1.e-2,hmin=Dx,iso=true,nbvx=1e8);
[eta0,u0,v0]=[eta0,u0,v0];  MAX=MAX;    D=D;
```

and for our new adapt mesh technique :

```
fespace Wh(Th,P1);
mesh Thp=Th;
uadapt=eta0+u0+v0;
real isoadapt=5.e-2, erradapt=1.e-7,smoothadapt=5.e-3,epsadapt=2e-2;
bool output =true;
real wv=0.;
{ AdaptGS(Thp,uadapt,erradapt,isoadapt,smoothadapt,epsadapt,Thinit,output,wv
    ➥); }
fespace Whp(Thp,P1);
Wh peta0=eta0,pu0=u0,pv0=v0;    Whp Dp,MAXp,eta0p,u0p,v0p;
matrix BWinterp=interpolate(Whp,Wh,inside=true);
eta0p[]=BWinterp*peta0[];u0p[]=BWinterp*pu0[];v0p[]=BWinterp*pv0[];MAXp[]=
    ➥BWinterp*MAX[];
Th=Thp;
[eta0,u0,v0]=[eta0p,u0p,v0p];   MAX=MAXp;   D=D;
```

We note that, we adapt the mesh around the solution each 100 iterations *i.e.* each 10 s by using the following value for the adapt mesh `uadapt`= $\eta + u + v$, `isoadapt`=5e-2, `erradapt`=1e-7, `smoothadapt`=5e-3, `epsadapt`=2e-2.

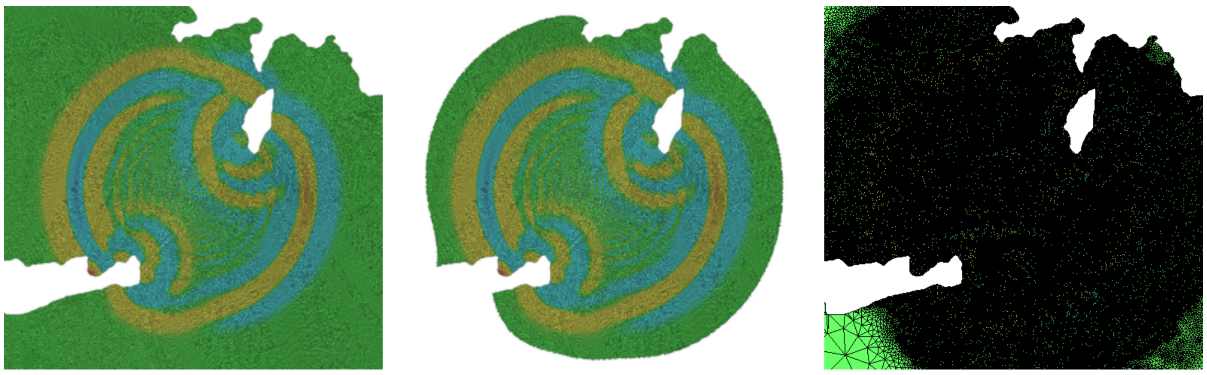


Figure 14 – The mesh and the solution at $t = 1000$ s, with the Full method at left, the Adapt GS at the center and the Adapt FF with `err=1.e-7` at the right.

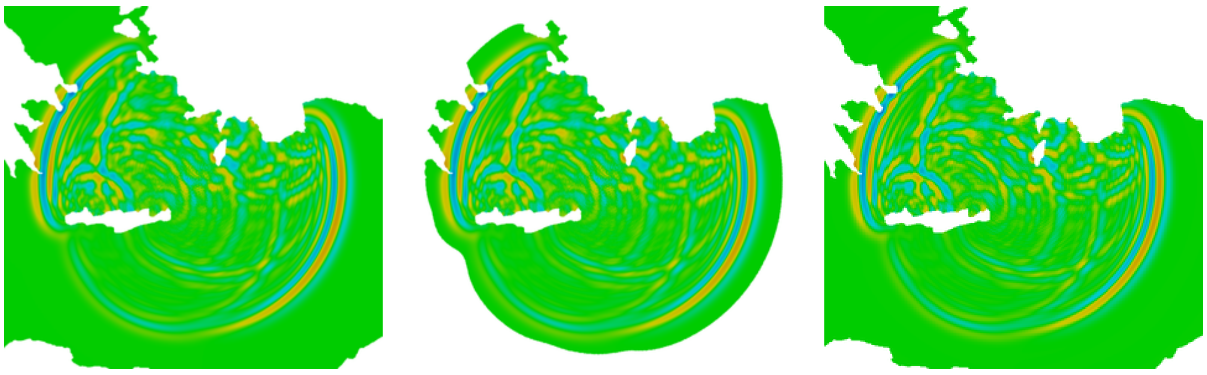


Figure 15 – The solution at $t = 3000$ s, with the Full method at left, the Adapt GS at the center and the Adapt FF with `err=1.e-7` at the right.

In order to compare the results between adaptmesh of FreeFem++, our new adapt mesh technique and without using mesh adaptation, we plot in addition to the free surface elevation η in the Figures 14 $\rightarrow$ 15, the variation of η vs time in Figure 16 at two wave 'gauges' placed at the positions represented by $\diamond$ in Figure 7 (top, right) and the mass of the water $\int \eta$. Specifically, gauges were placed at the points $(i) : (2350. * scale, 550. * scale)$, $(ii) : (2104. * scale, 665. * scale)$.
In the Figure 18, we represent the comparison between the three methods : Full, Adapt FF and Adapt GS of the maximum of the propagation of the solution at time $t = 6800sec$.
We also plot the computation time for each adapt mesh, the computation time of the simulation, the number of degree of freedom in Figure 17.
We can see in Figures 16 and 17 that the adaptmesh of FreeFem++ with `err=1.e-2` is the fastest method but unfortunately it does not preserve the mass invariant $\displaystyle\int \eta$. On the other hand, our new adapt mesh technique preserves the mass invariant throughout the simulation with an error of order $2.1e-3$ and an important time computation difference with the one without mesh adaptation which is very good method for the *Tsunami* wave propagation.
For the adaptmesh of FreeFem++ with `err=1.e-7`, we also get an almost a mass conservation with an error of order $9.5e-4$, but we obtain some difference in wave gauge with the Full method which is due to the refinement mesh adaptation and the interpolation of the solution, although the computation time is almost the double of the new adapt mesh technique.
We note also that we can go faster with our new mesh adaptation technique if we can also trunk the mass matrix after trunking the mesh, of course if the mass matrix is a constant along the simulation of the Full mesh, this is an outgoing project.

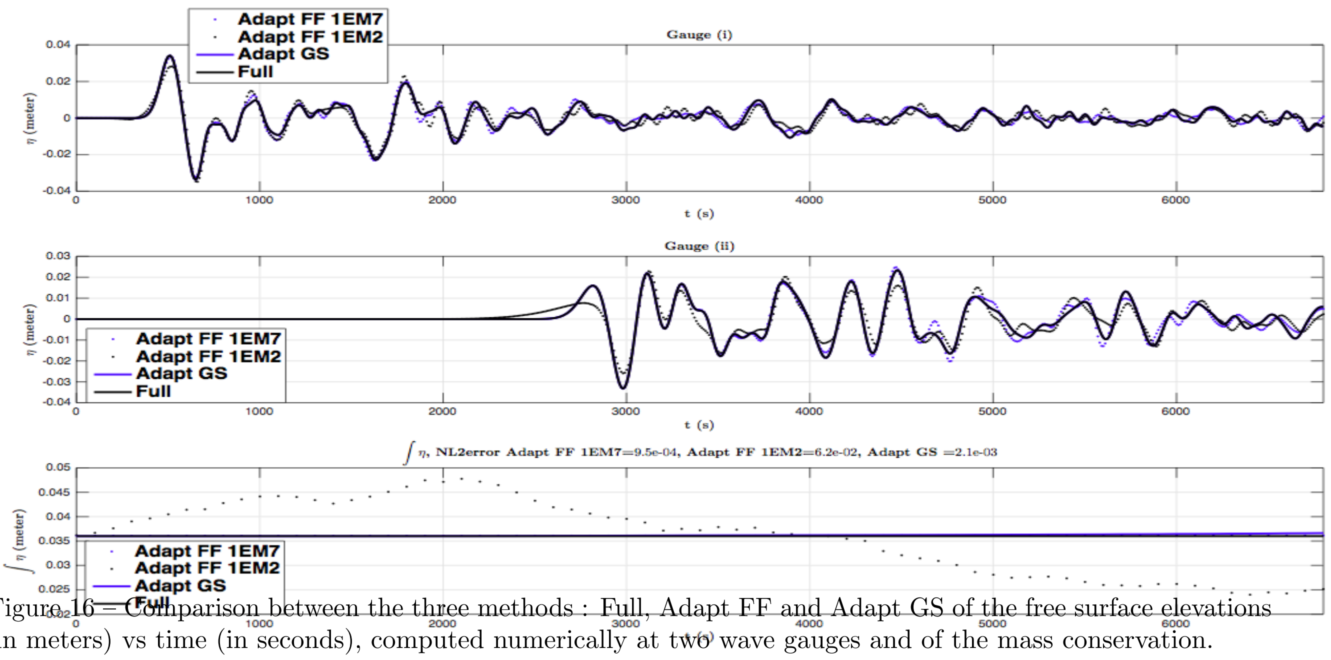


Figure 16 – Comparison between the three methods : Full, Adapt FF and Adapt GS of the free surface elevations (in meters) vs time (in seconds), computed numerically at two wave gauges and of the mass conservation.

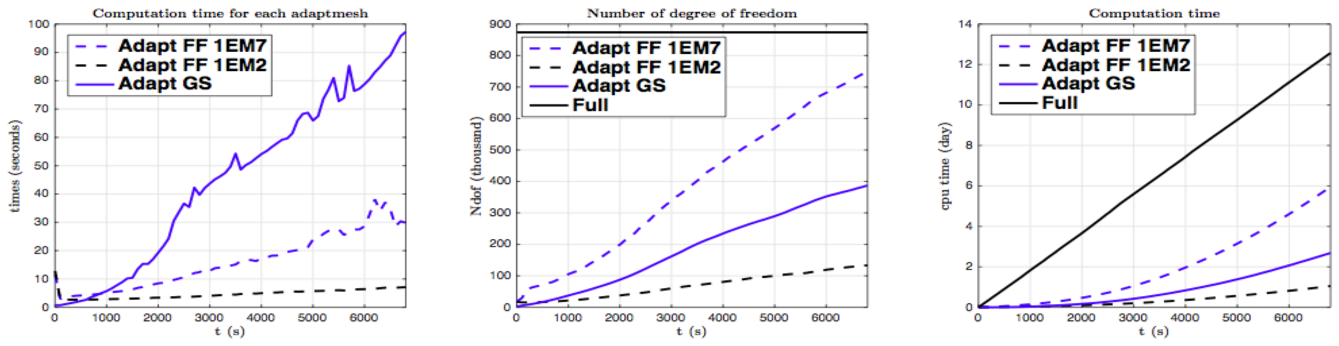


Figure 17 – Comparison between the three methods : Full, Adapt FF and Adapt GS of the computation time of each adaptemsh, the number of degree of freedom and the computation time of the simulation.

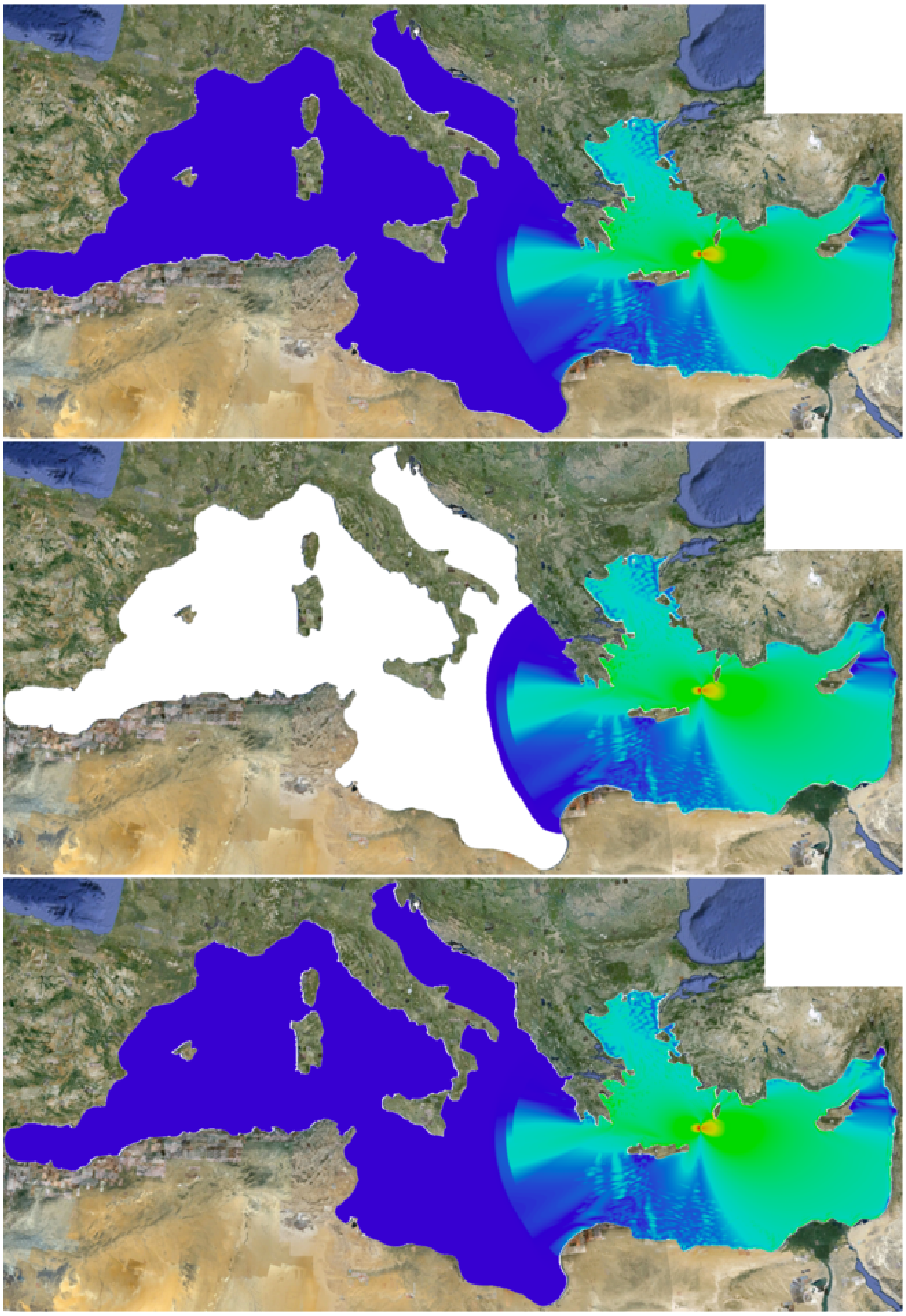


Figure 18 – Comparison between the three method Full (up), Adapt GS (middle) and Adapt FF (down) of the maximum of the propagation of the solution of a *Tsunami* wave in the Mediterranean sea for $t = 6800$ s.

4.3 Propagation of a *Tsunami* wave near Java island : *passive* generation .

In this section, we will take the same initial data as defined above in the *passive* generation part of Section 3, we take $\delta t = 1$ s as the time step size and we note that, we adapt the mesh after computing the initial data for η and then every 50 s by using the following value for the adapt mesh `uadapt`$= \eta + u + v$, `isoadapt`=3e-2, `erradapt`=1e-4, `smoothadapt`=5e-9, `epsadapt`=30e3.
We compare here the results between our new adapt mesh technique and without using mesh adaptation. To this end, we plot the free surface elevation η in the Figures 19 and 20.

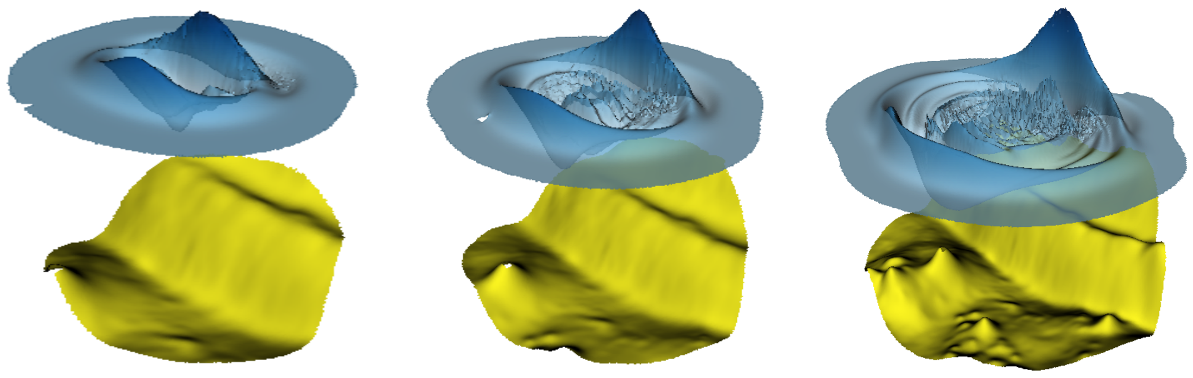


Figure 19 – *Passive* generation : The bottom and the solution at $t = 250$ s (left, solution (min$= -0.36$ m, max$= 0.38$ m), bottom (min$= -6207$ m, max$= -2096$ m)), $t = 500$ s (center, solution (min$= -0.26$ m, max$= 0.35$ m), bottom (min$= -6207$ m, max$= -243$ m)) and $t = 1000$ s (right, solution (min$= -0.21$ m, max$= 0.29$ m), bottom (min$= -6207$ m, max$= -100$ m)), with the Adapt GS method.

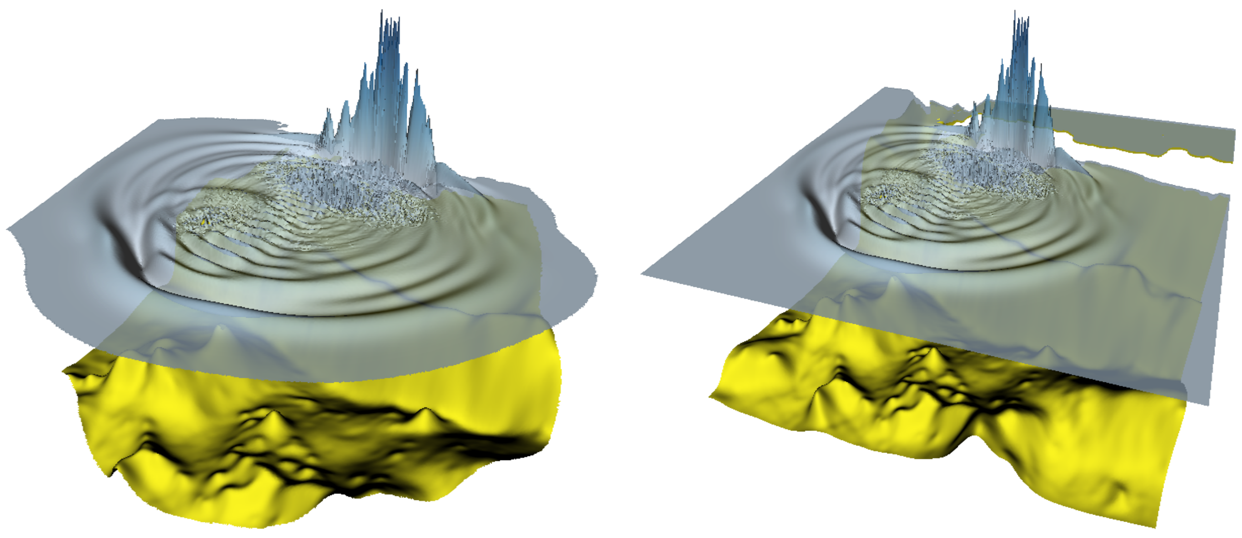


Figure 20 – *Passive* generation : Comparison between the bottom and the solution at $t = 1500$ s, with the Adapt GS method (left, solution (min$= -0.38$ m, max$= 1.07$ m), bottom (min$= -6207$ m, max$= -100$ m)) and with the Full one (right, solution (min$= -0.38$ m, max$= 1.07$ m), bottom (min$= -6207$ m, max$= -100$ m)).

We also plot, the variation of η vs time (in Figure 21) at four numerical wave gauges placed at the following locations: (i) $(107.345°, -9.295°)$, (ii) $(106.5°, -8°)$, (iii) $(105.9°, -10.35°)$ and (iv) $(107.7°, -11°)$ (see Figure 11 (left)) where (i) is the position of the epicenter. However, because of the large variations of the bottom, shorter waves were generated, especially around Christmas Island (southwest of Java) and around the undersea canyon near the Earthquake's epicenter.

Finally, we present a comparison of the Kinetic, Potential and Total energy with the Full mesh (in Figure 22, top left) and with the Adapt GS method (in Figure 22, top right) defined in [3] as:

$$E_c = \frac{1}{2}\rho_w \int_\Omega \left(\int_{-D(x,y)}^{\eta} |V|^2 \mathrm{d}z \right) \mathrm{d}x\mathrm{d}y, \qquad E_p = \frac{1}{2}\rho_w \cdot g \int_\Omega \eta^2 \mathrm{d}x\mathrm{d}y,$$

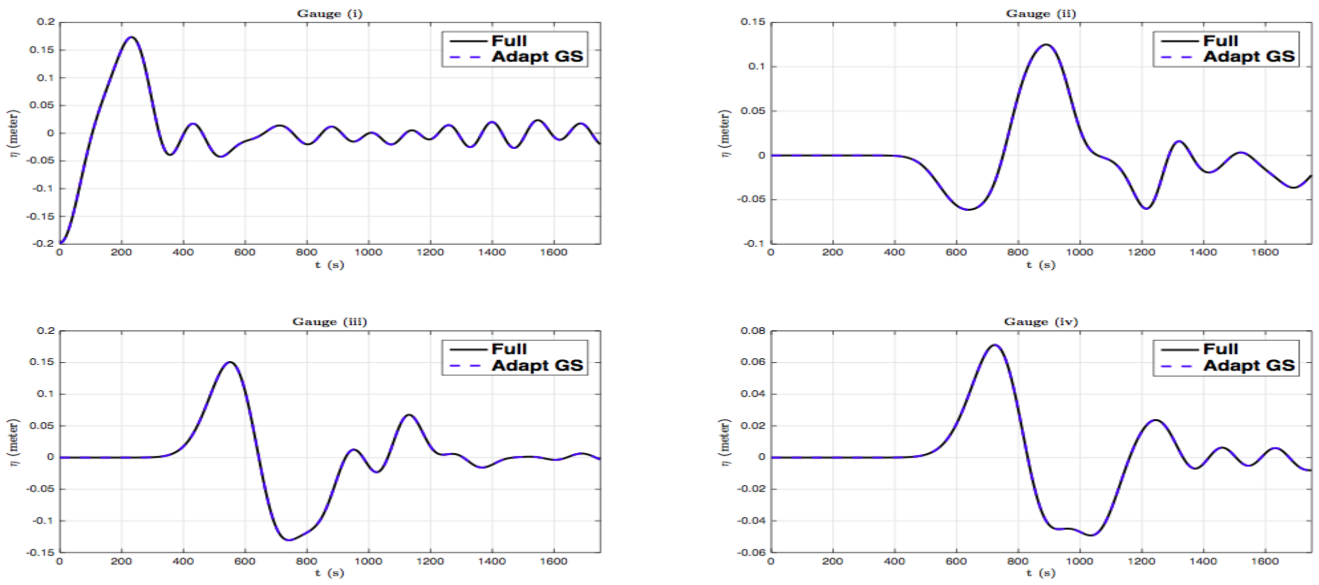


Figure 21 – *passive* generation : Comparison between the two methods Full and Adapt GS of the free surface elevations (in meters) vs time (in seconds), computed numerically at four wave gauges where the gauge (i) correspond to the epicenter.

where $\rho_w = 1027$ Kg/m^3 is the ocean water density, the number of degree of freedom (in Figure 22, down left) and the computation time of the simulation (in Figure 22, down right). We obtain here an error of order $2.6e-4$ between the Total Energy with Adapt GS and without adaptation.

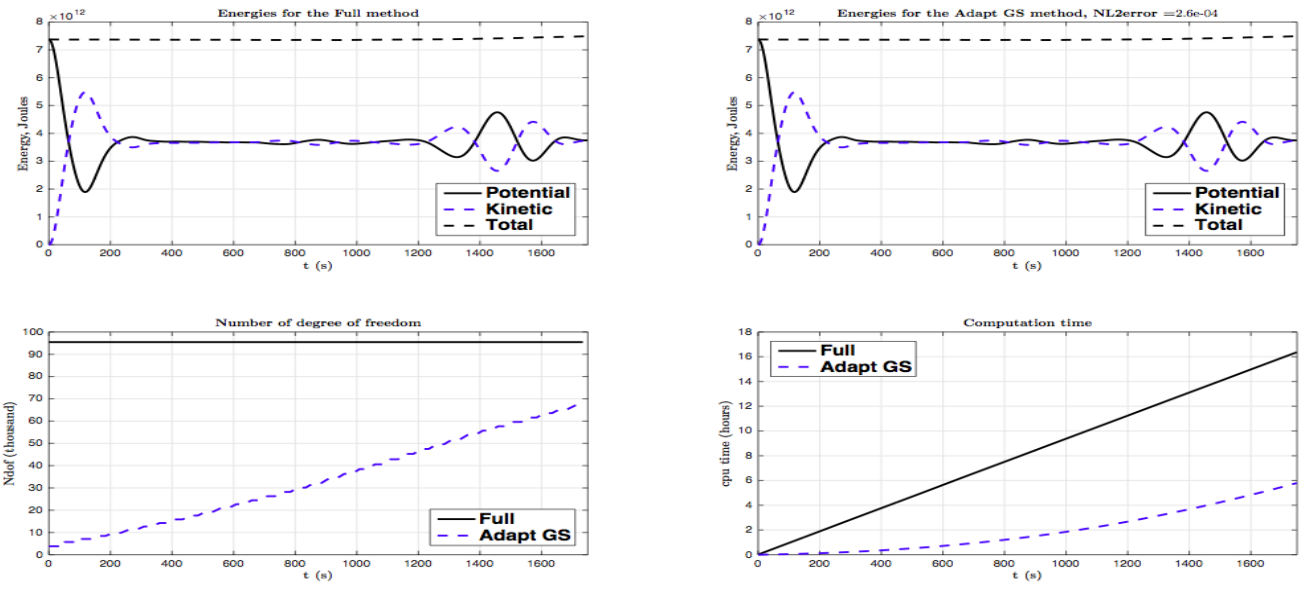


Figure 22 – *passive* generation : Comparison between the two methods Full and Adapt GS of the Kinetic, Potential and Total energy, the number of degree of freedom and the computation time of the simulation.

We present in the Figure 23 the comparison of the maximum of the propagation of the solution between the Full and the Adapt GS method at $t = 1750$ s.

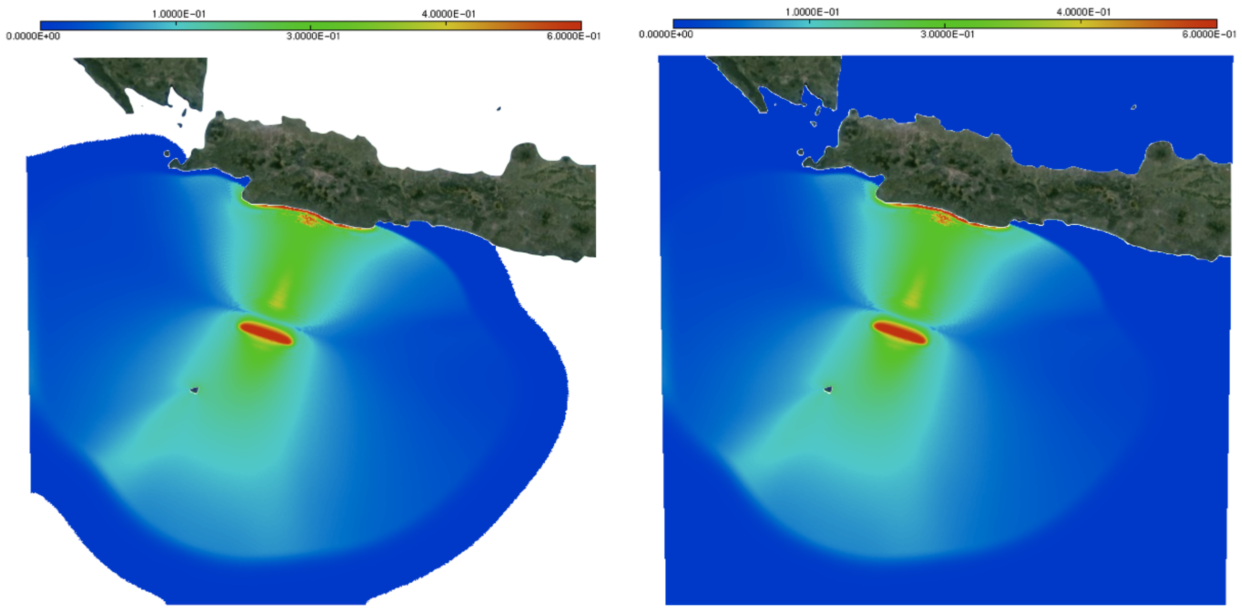


Figure 23 – *Passive* generation : Comparison between the maximum of the solution at $t = 1750$ s, with the Adapt GS method (left) and with the Full one (right).

4.4 Propagation of a *Tsunami* wave near Java island : *active* generation .

For a more realistic case as in the Java 2006 event, we use the *active* generation in order to model the generation of a *Tsunami* wave as in [4, 5]. In this case we consider zero initial conditions for both the surface elevation and the velocity field, we take $\delta t = 2$ s as the time step size, we assume that the bottom described in the Section 3 is moving in time and we note that we adapt the mesh, before the end of the generation time $t = 270$ s, each 3 iterations *i.e.* each 6 s by using the following value for the adapt mesh `uadapt`$= \eta + u + v$, `isoadapt`=5e-2, `erradapt`=1e-4, `smoothadapt`=5e-9, `epsadapt`=50e3 and then for $t > 270$ s each 25 iterations *i.e.* each 50 s.
We compare here only the results between our new adapt mesh technique and without using mesh adaptation. To this end, we plot the free surface elevation η in the Figures 24 $\rightarrow$ 26. However, because of the large variations of the bottom, shorter waves were generated, especially around Christmas Island (southwest of Java) and around the undersea canyon near the Earthquake's epicenter.
We also plot, the variation of η vs time (in Figure 27) at four numerical wave gauges placed at the following locations: (i) $(107.345°, -9.295°)$, (ii) $(106.5°, -8°)$, (iii) $(105.9°, -10.35°)$ and (iv) $(107.7°, -11°)$ (see Figure 11 (left)) where (i) is the position of the epicenter.

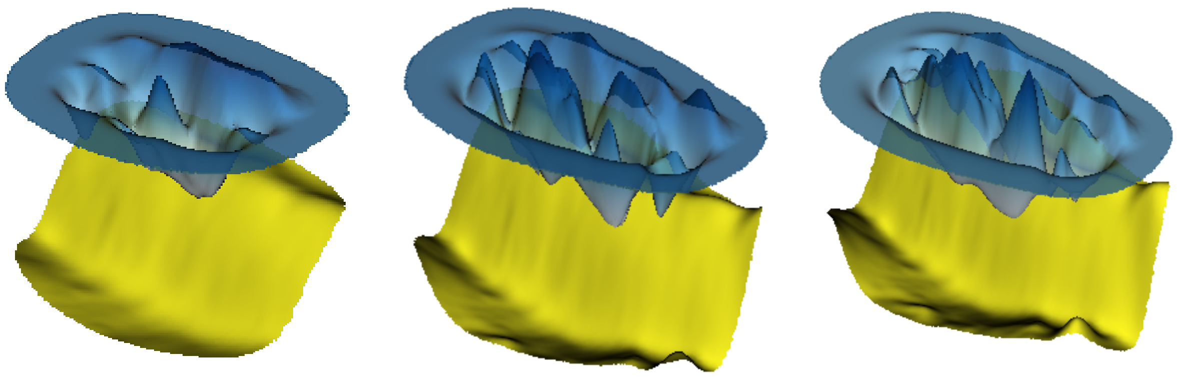


Figure 24 – *Active* generation : The bottom and the solution at $t = 100$ s (left, solution (min$= -0.17$ m, max$= 0.07$ m), bottom (min$= -6207$ m, max$= -2589$ m)), $t = 200$ s (center, solution (min$= -0.19$ m, max$= 0.08$ m), bottom (min$= -6207$ m, max$= -2285$ m)) and $t = 270$ s (right, solution (min$= -0.14$ m, max$= 0.10$ m), bottom (min$= -6207$ m, max$= -2084$ m)), with the Adapt GS method.

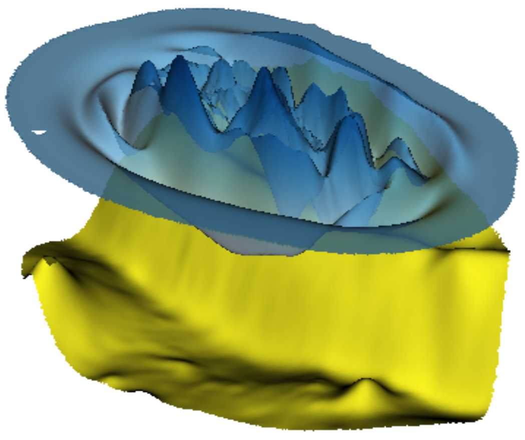
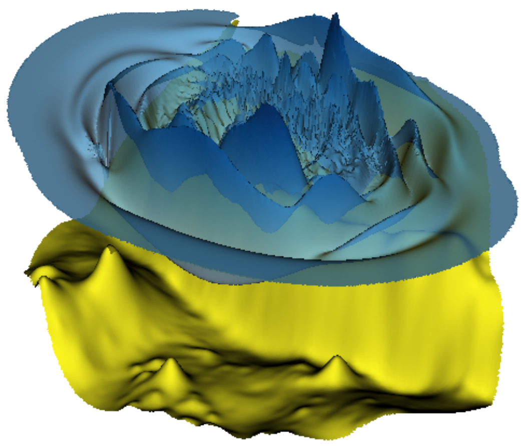



Figure 25 – *Active* generation : The bottom and the solution at $t = 500$ s (left, solution (min$= -0.15$ m, max$= 0.10$ m), bottom (min$= -6207$ m, max$= -260$ m)) and $t = 1000$ s (right, solution (min$= -0.14$ m, max$= 0.09$ m), bottom (min$= -6207$ m, max$= -100$ m)), with the Adapt GS method.

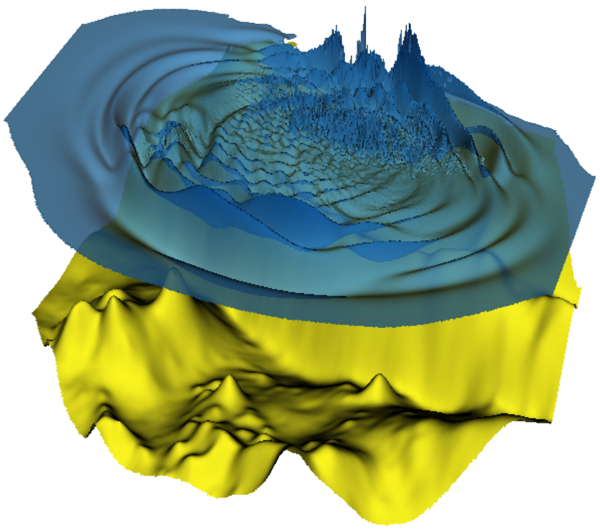
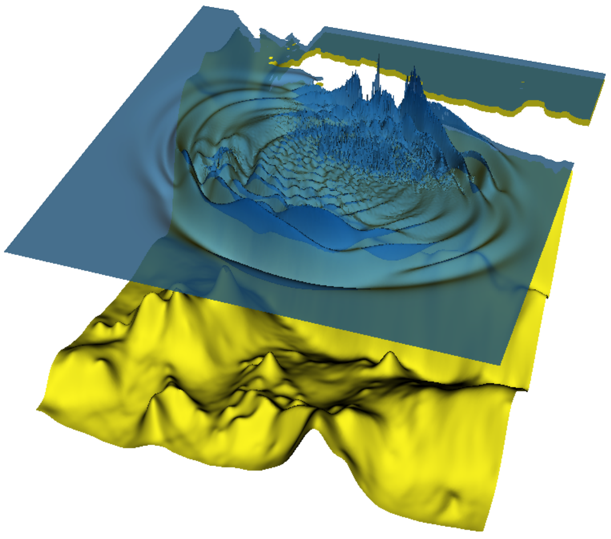



Figure 26 – *Active* generation : Comparison between the bottom and the solution at $t = 1500$ s, with the Adapt GS method (left, solution (min$= -0.29$ m, max$= 0.13$ m), bottom (min$= -6207$ m, max$= -100$ m)) and with the Full one (right, solution (min$= -0.29$ m, max$= 0.13$ m), bottom (min$= -6207$ m, max$= -100$ m)).

At the end, we present a comparison of the Kinetic, Potential and Total energy with the Full mesh (in Figure 28, top left) and with the Adapt GS method (in Figure 28, top right) defined in [3] as:

$$E_c = \frac{1}{2}\rho_w \int_\Omega \left(\int_{-D(x,y)}^{\eta} |V|^2 \mathrm{d}z \right) \mathrm{d}x\mathrm{d}y, \qquad E_p = \frac{1}{2}\rho_w \cdot g \int_\Omega \eta^2 \mathrm{d}x\mathrm{d}y,$$

the number of degree of freedom (in Figure 28, down left) and the computation time of the simulation (in Figure 28, down right). We obtain here an error of order $2e-5$ between the Total Energy with Adapt GS and without adaptation.
We present in the Figure 29 the comparison of the maximum of the propagation of the solution between the Full and the Adapt GS method at $t = 1750$ s.

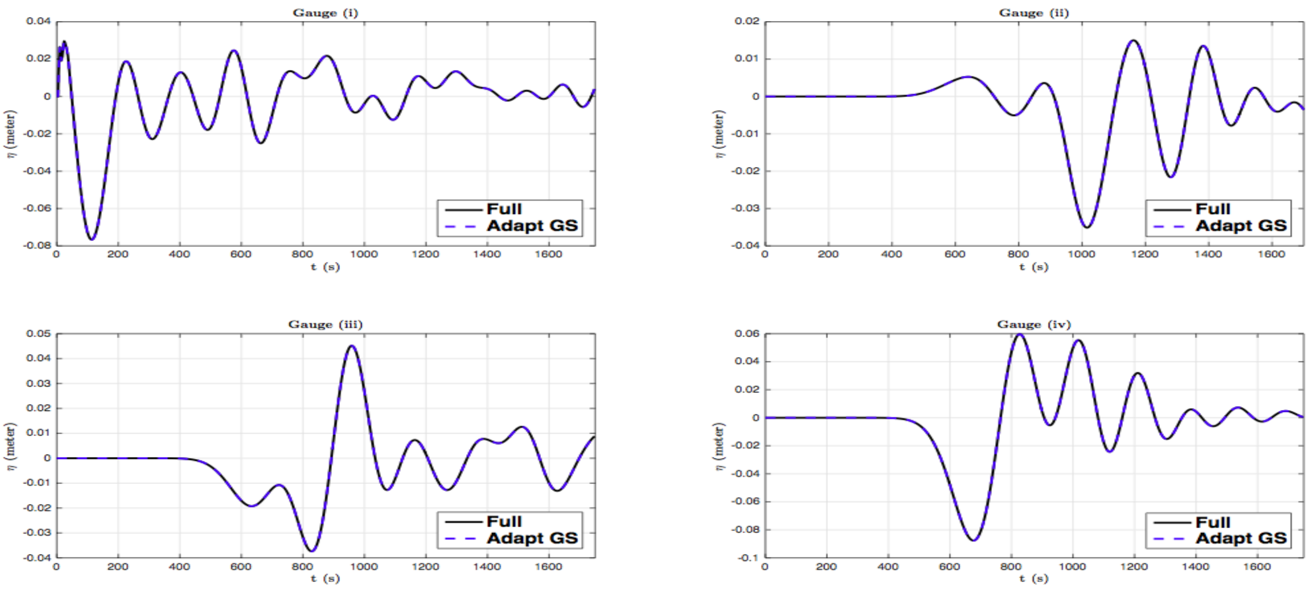


Figure 27 – *Active* generation : Comparison between the two methods Full and Adapt GS of the free surface elevations (in meters) vs time (in seconds), computed numerically at four wave gauges where the gauge (i) correspond to the epicenter.

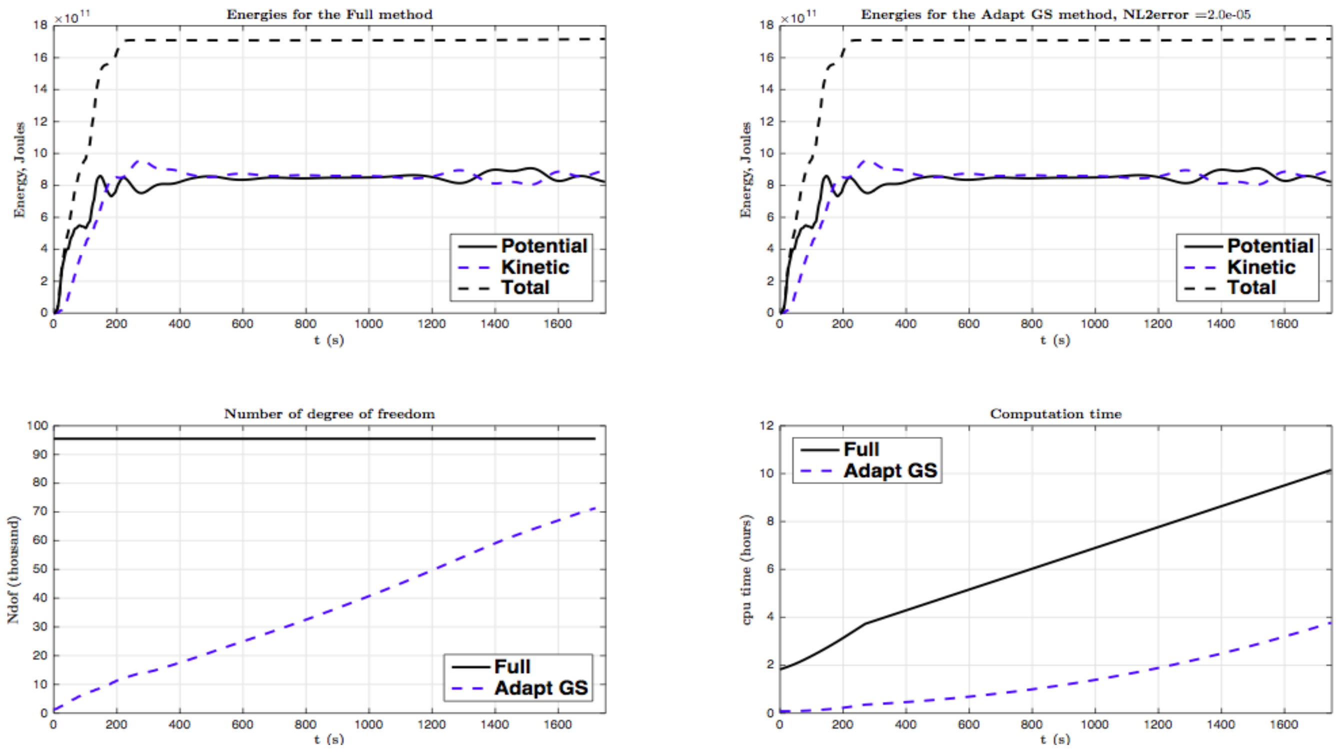


Figure 28 – *Active* generation : Comparison between the two methods Full and Adapt GS of the Kinetic, Potential and Total energy, the number of degree of freedom and the computation time of the simulation.

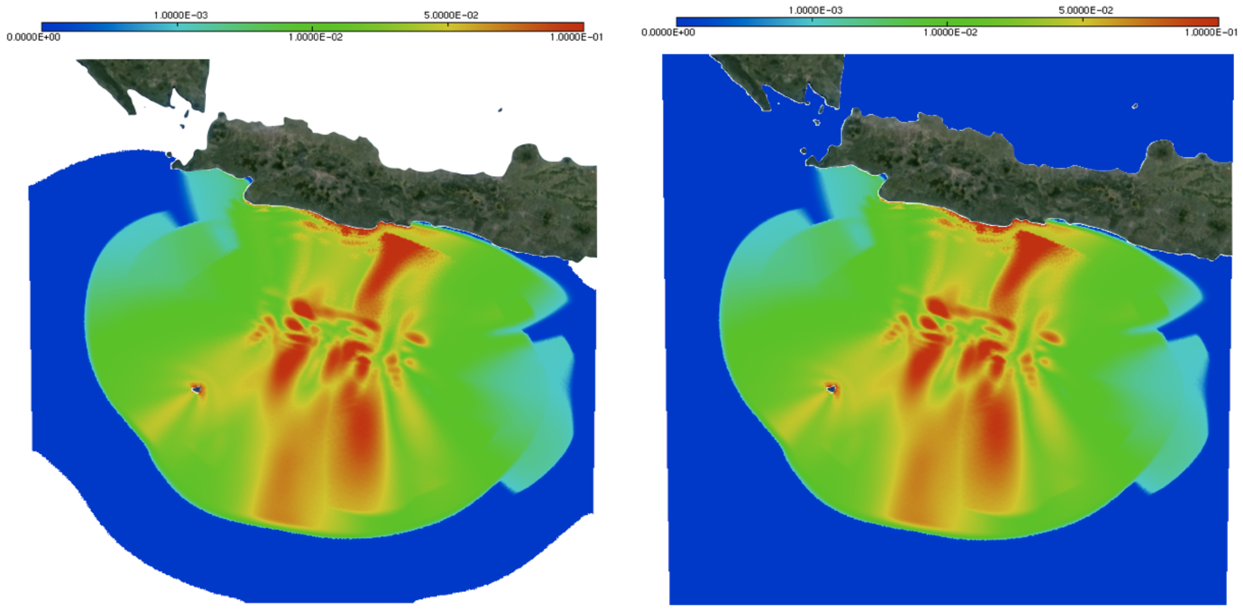


Figure 29 – *Active* generation : Comparison between the maximum of the solution at $t = 1750$ s, with the Adapt GS method (left) and with the Full one (right).

5 Conclusion and Outlook

We show in this paper, the usefulness of FreeFem++ for the *simplified* Boussinesq system of BBM type by building the domain, on the one hand using a photo taken from Google Earth® and on the other hand through an xyz bathymetry downloaded from the NOAA website. For the simulation of a *Tsunami* wave near Java island, the digital computing environment that we developed allows the integration of realistic data (bathymetry and geography) in a relatively simple framework. Another work concerning the generation, propagation and inundation of a *Tsunami* wave will be discussed in the case of the Shallow Water equations in [14], where in this case, we are not constraint by the smoothness of the bathymetry to avoid blow-up and where the same special adapt technique with a parallel version of the code will be introduced.

Acknowledgements : This work would not be done without a lot of advice and special care from Denys Dutykh, Frédéric Hecht, Dimitrios Mitsotakis, and Olivier Pantz, to whom I am grateful for all their help, their fruitful discussions and their remarks.

All the videos of the simulations of a *Tsunami* wave for the results presents in this paper are given in the following links :
`http://www.lamfa.u-picardie.fr/sadaka/movies/Tsu_Medit_Full.mov`
`http://www.lamfa.u-picardie.fr/sadaka/movies/Tsu_Medit_Adapt_GS.mov`
`http://www.lamfa.u-picardie.fr/sadaka/movies/Tsu_Medit_Adapt_FF_1EM2_sol.mov`
`http://www.lamfa.u-picardie.fr/sadaka/movies/Tsu_Medit_Adapt_FF_1EM2_mesh.mov`
`http://www.lamfa.u-picardie.fr/sadaka/movies/Tsu_Medit_Adapt_FF_1EM7_sol.mov`
`http://www.lamfa.u-picardie.fr/sadaka/movies/Tsu_Medit_Adapt_FF_1EM7_mesh.mov`
`http://www.lamfa.u-picardie.fr/sadaka/movies/Okada_Java_Dynamic.gif`
`http://www.lamfa.u-picardie.fr/sadaka/movies/Java_Bottom_Displacement.gif`
`http://www.lamfa.u-picardie.fr/sadaka/movies/Tsu_Java_sBBM_Pas_Adapt_GS_Full.mov`
`http://www.lamfa.u-picardie.fr/sadaka/movies/Tsu_Java_sBBM_Act_Adapt_GS_Full.mov`

References

[1] Joseph Valentin Boussinesq. Théorie générale des mouvements, qui sont propagés dans un canal rectangulaire horizontal. *C. R. Acad. Sci. Paris,* T73: 256-260, 1871. 1

[2] Denys Dutykh, Frédéric Dias, Water waves generated by a moving bottom, Springer Verlag 2007, Approx. 325 p., 170 illus., Hardcover, ISBN: 978-3-540-71255-8, 2007. 11

[3] Denys Dutykh, Frédéric Dias, Energy of Tsunami waves generated by bottom motion, Proc. R. Soc. A, 465, 725 - 744, 2009. 21

[4] Denys Dutykh, Dimitrios Mitsotakis, Xavier Gardeil, Frédéric Dias, On the use of finite fault solution for tsunami generation problems, Theor. Comput. Fluid Dyn., 27, 177-199, 2013. 11, 12, 12, 12, 20

[5] Denys Dutykh, Dimitrios Mitsotakis, Leonid B. Chubarov, Yuri I. Shokin, On the contribution of the horizontal sea-bed displacements into the tsunami generation process. Ocean Modelling, 56, 43-56, 2012. 20

[6] Joseph L. Hammack, Tsunamis - A Model of Their Generation and Propagation. PhD thesis, California Institute of Technology, 1972. 12

[7] Frédéric Hecht. Personal communication, 2011. 5

[8] Frédéric Hecht, Olivier Pironneau, Antoine Le Hyaric and Kohji Ohtsuka. Freefem++ Manual, 2012. 1

[9] Dimitrios Mitsotakis, Boussinesq systems in two space dimensions over a variable bottom for the generation and propagation of *Tsunami* waves. Mat. Comp. Simul., 80:860-873, 2009. 1,4,11,11

[10] Yoshimitsu Okada, Surface deformation due to shear and tensile faults in a half space. *Bull. Seism. Soc. Am.*, 75, 1135-1154, 1985. 11, 11

[11] Yoshimitsu Okada, Internal deformation due to shear and tensile faults in a half-space. *Bull. Seism. Soc. Am.*, 82, 1018-1040, 1992. 11

[12] Olivier Pantz. Personal communication, 2011. 5, 7

[13] Georges Sadaka. *Solution of 2D Boussinesq systems with FreeFem++ : the flat bottom case. JNM, Vol. 20, 303-324*, March 2013. 13

[14] Georges Sadaka. *Solving Shallow Water flows in 2D with FreeFem++ on structured mesh. hal-00715301*, 2012. 23

[15] Georges Sadaka. *FreeFem++, a tool to solve PDEs numerically. This work was start during the CIMPA School - Caracas 16-27 of April 2012, hal-00694787*, 2012. 7