SOLUTION OF 2D BOUSSINESQ SYSTEMS WITH FREEFEM++: THE FLAT BOTTOM CASE

Georges Sadaka

Solution of 2D Boussinesq systems with FreeFem++: the flat bottom case

Georges Sadaka*

May 14, 2012

Abstract

FreeFem++ is an open source platform to solve partial differential equations numerically, based on finite element methods. The FreeFem++ platform has been developed to facilitate teaching and basic research through prototyping. For the moment this platform is restricted to the numerical simulations of problems which admit a variational formulation. We will use FreeFem++ in this work to solve a three-parameter family of Boussinesq type systems in two space dimensions which approximate the three-dimensional Euler equations over an horizontal bottom and which was studied in [DouMitSau07, DouMitSau09, ChenGou09].

1 Introduction

It has often been observed that variations of the bottom could influence the damping of the waves including extreme ones as *Tsunamis* : the coral reef or the underwater forests in the first shoreline, mangroves; these underwater reefs are also used to prevent corrosion effects of coastal (see P. Azerad *and al.* [AzeBoucIvoIseMoh08] and [AzeBoucIvoIseMoh08]). In these cases, the underwater relief damped the wave energy, in contrast, in other situations we seek to harness this energy: some companies even offer projects underwater reefs for erectile produce energy from waves (see `http://www.aquamarinepower.com/`).

Chen, Goubet, Dougalis, Mitsotakis and Saut have considered 2D models with flat bottom [ChenGou09, DouMitSau07] then with variable bottom [Chen09, DiaDut07, Mit09], on the other hand Dutykh, Katsaounis and Mitsotakis have developed a code in finite volumes for the Boussinesq system with variable bottom in 1D ([DutKatMit11]) and Mitsotakis *and al.* in Galerkin finite elements (using B-splines [DouMitSau07]).

In this paper, we develop a FreeFem++ code for the simulation of Boussinesq equations with flat bottom :

$$\begin{aligned} \eta_t + \nabla\cdot\mathbf{V} + \nabla\cdot(\eta\mathbf{V}) + a\Delta\nabla\cdot\mathbf{V} - b\Delta\eta_t &= 0; \\ \mathbf{V}_t + \nabla\eta + \tfrac{1}{2}\nabla|\mathbf{V}|^2 + c\Delta\nabla\eta - d\Delta\mathbf{V}_t &= 0, \end{aligned} \tag{1}$$

we first check that the simulations provided by our numerical code are consistent with the results of the recent literature, including the work of Dougalis, Mitsotakis et Saut [DouMitSau07, DouMitSau09, DouMitSau10]. This establishes the adequacy of the chosen finite element discretization.

*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

The article is organized as follows: first we discretize the problem in space by using finite element method and in time by using an explicit second order Runge-Kutta scheme, then we develop all the step of the FreeFem++ code to solve the problem by using the technique of mesh adaptation and at the end we present some numerical result.

2 The Problem

We recall in this section the Boussinesq system in 2D which has been derived in [Sad11] as approximations to the three-dimensional Euler equations and describe irrotational free surface flow of an ideal fluid over an horizontal bottom.

2.1 Problem settings

The 2D Boussineq system for the incompressible fluid flows in $\Omega \subset \mathbb{R}^2$ is :

$$\begin{aligned} \eta_t + \nabla \cdot \mathbf{V} + \nabla \cdot (\eta \mathbf{V}) + a\Delta\nabla \cdot \mathbf{V} - b\Delta\eta_t &= 0; \\ \mathbf{V}_t + \nabla\eta + \tfrac{1}{2}\nabla|\mathbf{V}|^2 + c\Delta\nabla\eta - d\Delta\mathbf{V}_t &= 0, \end{aligned} \tag{2}$$

The variables in (2) are non-dimensional and unscaled : $\mathbf{X} = (x, y) \in \Omega$ and $t > 0$ are proportional to position along the channel and time, respectively, $\eta = \eta(\mathbf{X}, t)$ is proportional to the deviation of the free surface from its rest position, $\mathbf{V} = \mathbf{V}(\mathbf{X}, t) = \begin{pmatrix} u(\mathbf{X}, t) \\ v(\mathbf{X}, t) \end{pmatrix} = (u, v)^T = (u; v)$ is proportional to the horizontal velocity of the fluid at some height, $\nabla = \begin{pmatrix} \partial_x \\ \partial_y \end{pmatrix}$ is the gradient, $\nabla \cdot \begin{pmatrix} \star \end{pmatrix} = \partial_x \star + \partial_y$ is the divergence and $\Delta = \partial_{xx} + \partial_{yy}$ is the laplacian. The coefficients a, b, c and d are given by the following formulas :

$$a = \frac{1}{2}\left(\theta^2 - \frac{1}{3}\right)\nu, b = \frac{1}{2}\left(\theta^2 - \frac{1}{3}\right)(1-\nu), c = \frac{1}{2}\left(1-\theta^2\right)\mu, d = \frac{1}{2}\left(1-\theta^2\right)(1-\mu) \tag{3}$$

where ν, μ are real constants and $0 \leqslant \theta \leqslant 1$.
We note that the dispersive constants a, b, c and d satisfy the physical constraints (see [BonaChenSau04] for detail):

$$a + b + c + d = \frac{1}{3} \text{ and } c + d \geq 0 \tag{4}$$

We now list some of the several family of Boussinesq systems 2D in Table 1 of the form (2) :

In [DouMitSau07], V. Dougalis, D. Mitsotakis and J-C. Saut have studied the Well-Posedness of the Boussinesq system (2) (where $b \neq 0$ and $d \neq 0$) and have shown that this system is at least nonlinearly well-posed locally; and in the case of KdV-KdV system (where $b = d = 0, a = c = 1/6$), F. Linares, D. Pilod and J-C. Saut proved recently in [LinPilSau11] the Well-Posedness of this system.

Following [WalBer02], we can write (2) as :

$$\begin{aligned} \Upsilon - \Delta\mathbf{V} &= 0; \\ \eta_t + \nabla \cdot \mathbf{V} + \nabla \cdot (\eta \mathbf{V}) + a\nabla \cdot \Upsilon - b\Delta\eta_t &= 0; \\ \Theta - \Delta\eta &= 0; \\ \mathbf{V}_t + \nabla\eta + \tfrac{1}{2}\nabla|\mathbf{V}|^2 + c\nabla\Theta - d\Delta\mathbf{V}_t &= 0, \end{aligned} \tag{5}$$

where $\Upsilon = \left(\Upsilon^1, \Upsilon^2\right)^T = \left(\Upsilon^1; \Upsilon^2\right)$.

System	θ^2	ν	μ	References
BBM-BBM	$2/3$	0	0	[Chen09], [ChenGou09], [DouMitSau07], [DouMitSau09], [DouMitSau10].
Bona-Smith	$2/3 \leq \theta^2 \leq 1$	0	$\frac{4-6\theta^2}{3(1-\theta^2)}$	[ChenGou09], [DouMitSau07], [DouMitSau09], [DouMitSau10].
" General " Boussinesq	$0 \leq \theta^2 \leq 1$	any	any	[ChenGou09], [DouMitSau07], [DouMitSau10].
KdV-KdV	$2/3$	1	1	[LinPilSau11]

Table 1: Examples of Boussinesq systems in 2D.

3 Numerical Scheme

In this section, we will show the spatial discretization using finite element method with $\mathbb{P}_1$ continuous piecewise linear functions as shown in [WalBer02] and for the time marching scheme an explicit second order Runge-Kutta [Dem91] scheme as used in [DouMitSau07].
We will use in our code a mesh adaptation technic that we can use solving the problem by using the method based on the declaration of the problem obtained by the weak formulation of the system (5); or by using the second method that consist to build matrices and vectors to solve the direct system $\mathbf{AX} = \mathbf{B}$, where the matrix $\mathbf{A}$ and the vectors $\mathbf{X}, \mathbf{B}$ will be defined in the sequel.

3.1 Spatial discretization

We let Ω be a convex, plane domain, let $\mathbf{T}_h$ denote a regular, quasi uniform triangulation of Ω with triangles of maximum size $h < 1$ [BreSco94], let $V_h = \{v_h \in C^0(\bar{\Omega}); v_h|_T \in \mathbb{P}_1(T), \forall T \in \mathbf{T}_h\}$ denote a finite-dimensional subspace of $H^1(\Omega) = \{u \in L^2(\Omega) \text{ s.t. } \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \in L^2(\Omega)\}$ where $\mathbb{P}_1$ is the set of polynomials of $\mathbb{R}$ of degrees ≤ 1 and let $\langle \cdot;\cdot \rangle$ denote the L^2 inner product on Ω.
Consider the weak formulation of the system (5), find $\eta_h, u_h, v_h \in V_h$ such that $\forall \phi_h \in V_h$ we have :

$$\begin{aligned}
\left\langle \Upsilon_h^1; \phi_h \right\rangle - \left\langle \Delta u_h; \phi_h \right\rangle = 0; \quad \left\langle \Upsilon_h^2; \phi_h \right\rangle - \left\langle \Delta v_h; \phi_h \right\rangle = 0; \quad \left\langle \Theta_h; \phi_h \right\rangle - \left\langle \Delta \eta_h; \phi_h \right\rangle &= 0; \\
\left\langle (Id - b\Delta)\eta_{ht} + \nabla \cdot (u_h; v_h) + \eta_{hx} u_h + \eta_h u_{hx} + \eta_{hy} u_h + \eta_h u_{hy} + a\nabla \cdot \left(\Upsilon_h^1; \Upsilon_h^2\right); \phi_h \right\rangle &= 0; \\
\left\langle (Id - d\Delta)u_{ht} + \eta_{hx} + u_h u_{hx} + v_h v_{hx} + c\Theta_{hx}; \phi_h \right\rangle &= 0; \\
\left\langle (Id - d\Delta)v_{ht} + \eta_{hy} + u_h u_{hy} + v_h v_{hy} + c\Theta_{hy}; \phi_h \right\rangle &= 0.
\end{aligned} \tag{6}$$

To simplify, we denote $\Phi = \phi_h, \mathbf{E} = \eta_h, \mathbf{U} = u_h, \mathbf{V} = v_h, \mathbf{T} = \Theta_h, \mathbf{P} = \Upsilon_h^1$ and $\mathbf{Q} = \Upsilon_h^2$, so the system (6) is equivalent to the following system :

$$\begin{aligned}
\langle \mathbf{P}; \Phi \rangle &= \langle \Delta \mathbf{U}; \Phi \rangle; \quad \langle \mathbf{Q}; \Phi \rangle = \langle \Delta \mathbf{V}; \Phi \rangle; \quad \langle \mathbf{T}; \Phi \rangle = \langle \Delta \mathbf{E}; \Phi \rangle; \\
\left\langle (Id - b\Delta)\partial_t \mathbf{E}; \Phi \right\rangle &= -\langle \nabla \cdot (\mathbf{U}; \mathbf{V}) + \mathbf{E}_x \mathbf{U} + \mathbf{E}\mathbf{U}_x + \mathbf{E}_y \mathbf{V} + \mathbf{E}\mathbf{V}_y + a\nabla \cdot (\mathbf{P}; \mathbf{Q}); \Phi \rangle \\
&= -\mathbf{F}(\mathbf{E}, \mathbf{U}, \mathbf{V}, \mathbf{P}, \mathbf{Q}); \\
\left\langle (Id - d\Delta)\partial_t \mathbf{U}; \Phi \right\rangle &= -\langle \mathbf{E}_x + \mathbf{U}\mathbf{U}_x + \mathbf{V}\mathbf{V}_x + c\mathbf{T}_x; \Phi \rangle = -\mathbf{G}(\mathbf{E}, \mathbf{U}, \mathbf{V}, \mathbf{T}); \\
\left\langle (Id - d\Delta)\partial_t \mathbf{V}; \Phi \right\rangle &= -\langle \mathbf{E}_y + \mathbf{U}\mathbf{U}_y + \mathbf{V}\mathbf{V}_y + c\mathbf{T}_y; \Phi \rangle = -\mathbf{H}(\mathbf{E}, \mathbf{U}, \mathbf{V}, \mathbf{T}).
\end{aligned} \tag{7}$$

3.2 Time marching scheme

Our method is based on an explicit second order Runge-Kutta scheme. To this end, let us denote by $(\mathbf{E}^{n+1}, \mathbf{U}^{n+1}, \mathbf{V}^{n+1})$ and $(\mathbf{E}^{n}, \mathbf{U}^{n}, \mathbf{V}^{n}, \mathbf{P}^{n}, \mathbf{Q}^{n}, \mathbf{T}^{n})$ the approximate value at time $t = t^{n+1}$ and $t = t^{n}$, respectively and by δt the time step size. Then, by using (7), the unknown fields at time $t = t^{n+1}$ are defined as the solution of the system

$$\left\{\begin{array}{l} \langle \mathbf{P}^n; \Phi\rangle = \langle \Delta \mathbf{U}^n; \Phi\rangle\,; \qquad \langle \mathbf{Q}^n; \Phi\rangle = \langle \Delta \mathbf{V}^n; \Phi\rangle\,; \qquad \langle \mathbf{T}^n; \Phi\rangle = \langle \Delta \mathbf{E}^n; \Phi\rangle\,; \\ \langle \mathbf{E}^{n+1}; \Phi\rangle = \langle \mathbf{E}^n + \dfrac{\mathbf{E}^{k1} + \mathbf{E}^{k2}}{2}; \Phi\rangle; \langle \mathbf{U}^{n+1}; \Phi\rangle = \langle \mathbf{U}^n + \dfrac{\mathbf{U}^{k1} + \mathbf{U}^{k2}}{2}; \Phi\rangle; \\ \langle \mathbf{V}^{n+1}; \Phi\rangle = \langle \mathbf{V}^n + \dfrac{\mathbf{V}^{k1} + \mathbf{V}^{k2}}{2}; \Phi\rangle. \end{array}\right. \tag{8}$$

where :

$$\left\{\begin{array}{l} \big\langle (Id - b\Delta)\mathbf{E}^{k1}; \Phi\big\rangle = -\delta t \cdot \mathbf{F}\left(\mathbf{E}^n, \mathbf{U}^n, \mathbf{V}^n, \mathbf{P}^n, \mathbf{Q}^n\right); \\ \big\langle (Id - d\Delta)\mathbf{U}^{k1}; \Phi\big\rangle = -\delta t \cdot \mathbf{G}\left(\mathbf{E}^n, \mathbf{U}^n, \mathbf{V}^n, \mathbf{T}^n\right); \\ \big\langle (Id - d\Delta)\mathbf{V}^{k1}; \Phi\big\rangle = -\delta t \cdot \mathbf{H}\left(\mathbf{E}^n, \mathbf{U}^n, \mathbf{V}^n, \mathbf{T}^n\right); \\ \big\langle \mathbf{P}^{k1}; \Phi\big\rangle = \big\langle \mathbf{U}^{k1}_{xx} + \mathbf{U}^{k1}_{yy}; \Phi\big\rangle; \big\langle \mathbf{Q}^{k1}; \Phi\big\rangle = \big\langle \mathbf{V}^{k1}_{xx} + \mathbf{V}^{k1}_{yy}; \Phi\big\rangle; \big\langle \mathbf{T}^{k1}; \Phi\big\rangle = \big\langle \mathbf{E}^{k1}_{xx} + \mathbf{E}^{k1}_{yy}; \Phi\big\rangle. \end{array}\right. \tag{9}$$

and

$$\left\{\begin{array}{rcl} \big\langle (Id - b\Delta)\mathbf{E}^{k2}; \Phi\big\rangle & = & -\delta t \cdot \mathbf{F}\left(\mathbf{E}^n + \mathbf{E}^{k1}, \mathbf{U}^n + \mathbf{U}^{k1}, \mathbf{V}^n + \mathbf{V}^{k1}, \mathbf{P}^n + \mathbf{P}^{k1}, \mathbf{Q}^n + \mathbf{Q}^{k1}\right); \\ \big\langle (Id - d\Delta)\mathbf{U}^{k2}; \Phi\big\rangle & = & -\delta t \cdot \mathbf{G}\left(\mathbf{E}^n + \mathbf{E}^{k1}, \mathbf{U}^n + \mathbf{U}^{k1}, \mathbf{V}^n + \mathbf{V}^{k1}, \mathbf{T}^n + \mathbf{T}^{k1}\right); \\ \big\langle (Id - d\Delta)\mathbf{V}^{k2}; \Phi\big\rangle & = & -\delta t \cdot \mathbf{H}\left(\mathbf{E}^n + \mathbf{E}^{k1}, \mathbf{U}^n + \mathbf{U}^{k1}, \mathbf{V}^n + \mathbf{V}^{k1}, \mathbf{T}^n + \mathbf{T}^{k1}\right). \end{array}\right. \tag{10}$$

By integrating by parts where we have second order derivative and by developing all the terms of the first order derivative in (8), (9) and (10), we deduce:

$$\begin{array}{c} \langle \mathbf{P}^n; \Phi\rangle = -\langle \nabla \mathbf{U}^n; \nabla\Phi\rangle + \left\langle \dfrac{\partial \mathbf{U}^n}{\partial n}; \Phi \right\rangle_{\partial\Omega}; \langle \mathbf{Q}^n; \Phi\rangle = -\langle \nabla \mathbf{V}^n; \nabla\Phi\rangle + \left\langle \dfrac{\partial \mathbf{V}^n}{\partial n}; \Phi \right\rangle_{\partial\Omega}; \\ \langle \mathbf{T}^n; \Phi\rangle = -\langle \nabla \mathbf{E}^n; \nabla\Phi\rangle + \left\langle \dfrac{\partial \mathbf{E}^n}{\partial n}; \Phi \right\rangle_{\partial\Omega}; \end{array} \tag{11}$$

$$\left\{\begin{array}{rcl} \big\langle \mathbf{E}^{k1}; \Phi\big\rangle + b\big\langle \nabla \mathbf{E}^{k1}; \nabla\Phi\big\rangle - b\left\langle \dfrac{\partial \mathbf{E}^{k1}}{\partial n}; \Phi \right\rangle_{\partial\Omega} & = & -\delta t \cdot \mathbf{F}\left(\mathbf{E}^n, \mathbf{U}^n, \mathbf{V}^n, \mathbf{P}^n, \mathbf{Q}^n\right); \\ \big\langle \mathbf{U}^{k1}; \Phi\big\rangle + d\big\langle \nabla \mathbf{U}^{k1}; \nabla\Phi\big\rangle - d\left\langle \dfrac{\partial \mathbf{U}^{k1}}{\partial n}; \Phi \right\rangle_{\partial\Omega} & = & -\delta t \cdot \mathbf{G}\left(\mathbf{E}^n, \mathbf{U}^n, \mathbf{V}^n, \mathbf{T}^n\right); \\ \big\langle \mathbf{V}^{k1}; \Phi\big\rangle + d\big\langle \nabla \mathbf{V}^{k1}; \nabla\Phi\big\rangle - d\left\langle \dfrac{\partial \mathbf{V}^{k1}}{\partial n}; \Phi \right\rangle_{\partial\Omega} & = & -\delta t \cdot \mathbf{H}\left(\mathbf{E}^n, \mathbf{U}^n, \mathbf{V}^n, \mathbf{T}^n\right); \\ \big\langle \mathbf{P}^{k1}; \Phi\big\rangle = -\big\langle \nabla \mathbf{U}^{k1}; \nabla\Phi\big\rangle + \left\langle \dfrac{\partial \mathbf{U}^{k1}}{\partial n}; \Phi \right\rangle_{\partial\Omega}; & & \\ \big\langle \mathbf{Q}^{k1}; \Phi\big\rangle = -\big\langle \nabla \mathbf{V}^{k1}; \nabla\Phi\big\rangle + \left\langle \dfrac{\partial \mathbf{V}^{k1}}{\partial n}; \Phi \right\rangle_{\partial\Omega}; & & \\ \big\langle \mathbf{T}^{k1}; \Phi\big\rangle = -\big\langle \nabla \mathbf{E}^{k1}; \nabla\Phi\big\rangle + \left\langle \dfrac{\partial \mathbf{E}^{k1}}{\partial n}; \Phi \right\rangle_{\partial\Omega}; & & \end{array}\right. \tag{12}$$

and

$$\begin{cases} \langle \mathbf{E}^{k2}; \Phi \rangle + b \langle \nabla \mathbf{E}^{k2}; \nabla \Phi \rangle - b \left\langle \dfrac{\partial \mathbf{E}^{k2}}{\partial n}; \Phi \right\rangle_{\partial\Omega} = \\ \qquad -\delta t \cdot \mathbf{F}\left(\mathbf{E}^n + \mathbf{E}^{k1}, \mathbf{U}^n + \mathbf{U}^{k1}, \mathbf{V}^n + \mathbf{V}^{k1}, \mathbf{P}^n + \mathbf{P}^{k1}, \mathbf{Q}^n + \mathbf{Q}^{k1}\right); \\ \langle \mathbf{U}^{k2}; \Phi \rangle + d \langle \nabla \mathbf{U}^{k2}; \nabla \Phi \rangle - d \left\langle \dfrac{\partial \mathbf{U}^{k2}}{\partial n}; \Phi \right\rangle_{\partial\Omega} = \\ \qquad -\delta t \cdot \mathbf{G}\left(\mathbf{E}^n + \mathbf{E}^{k1}, \mathbf{U}^n + \mathbf{U}^{k1}, \mathbf{V}^n + \mathbf{V}^{k1}, \mathbf{T}^n + \mathbf{T}^{k1}\right); \\ \langle \mathbf{V}^{k2}; \Phi \rangle + d \langle \nabla \mathbf{V}^{k2}; \nabla \Phi \rangle - d \left\langle \dfrac{\partial \mathbf{V}^{k2}}{\partial n}; \Phi \right\rangle_{\partial\Omega} = \\ \qquad -\delta t \cdot \mathbf{H}\left(\mathbf{E}^n + \mathbf{E}^{k1}, \mathbf{U}^n + \mathbf{U}^{k1}, \mathbf{V}^n + \mathbf{V}^{k1}, \mathbf{T}^n + \mathbf{T}^{k1}\right). \end{cases} \tag{13}$$

Remark : It's easy with FreeFem++ to define boundary condition, in fact if we have the Dirichlet Boundary Conditions on a border $\Gamma_1 \subset \mathbb{R}$ like $\mathbf{U}|_{\Gamma_1} = f$, then it is defined as `on(gamma1,u=f)`, where `u` is the unknown function in the problem. We note that the Neumann Boundary Conditions on $\Gamma_2 \subset \mathbb{R}$, like $\dfrac{\partial \mathbf{U}}{\partial n}|_{\Gamma_2} = g$, appear in the Weak formulation of the problem after integrating by parts for example in the system (11) we have $\left\langle \dfrac{\partial \mathbf{U}}{\partial n}; \Phi \right\rangle_{\Gamma_2} = \langle g; \Phi \rangle_{\Gamma_2} = \displaystyle\int_{\Gamma_2} g \cdot \Phi$ which is defined in FreeFem++ by `int1d(Th,gamma2)(g*phi)` where `Th` is the triangulated domain of Ω. We will see in the next section how it's also easy to define the Bi-Periodic Boundary Conditions.
We remark also that the system (8) can be written on the following matrix form:

$$\underbrace{\begin{pmatrix} \mathbf{M} & 0 & 0 \\ 0 & \mathbf{M} & 0 \\ 0 & 0 & \mathbf{M} \end{pmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{pmatrix} \mathbf{E}^{n+1} \\ \mathbf{U}^{n+1} \\ \mathbf{V}^{n+1} \end{pmatrix}}_{\mathbf{X}} = \underbrace{\begin{pmatrix} \langle \mathbf{E}^n + \dfrac{\mathbf{E}^{k1} + \mathbf{E}^{k2}}{2}; \Phi \rangle \\ \langle \mathbf{U}^n + \dfrac{\mathbf{U}^{k1} + \mathbf{U}^{k2}}{2}; \Phi \rangle \\ \langle \mathbf{V}^n + \dfrac{\mathbf{V}^{k1} + \mathbf{V}^{k2}}{2}; \Phi \rangle \end{pmatrix}}_{\mathbf{B}} \tag{14}$$

where $\mathbf{M}_{ij} = \displaystyle\int_\Omega \phi_i \phi_j \mathrm{d}x\mathrm{d}y$ is the mass matrix.

Algorithm 1:

Finally, to solve the systems (8), (11), (12) and (13), we follow as :

```
Set              E^n = E^0 = η_h0 = η_0, U^n = U^0 = u_h0 = u_0, V^n = V^0 = v_h0 = v_0
Set              P^n = P_h0, Q^n = Q_h0, T^n = T_h0, P^k1 = P_hk1, Q^k1 = Q_hk1, T^k1 = T_hk1
Set              E^{n+1} = η_h, U^{n+1} = u_h, V^{n+1} = v_h
Set              E^k1 = η_hk1, U^k1 = u_hk1, V^k1 = v_hk1, E^k2 = η_hk2, U^k2 = u_hk2, V^k2 = v_hk2
For  t = 0 : δt : T
                 Mesh adaptation, (optional)
                 Compute P_h0, Q_h0, T_h0      Compute η_hk1, u_hk1, v_hk1
                 Compute P_hk1, Q_hk1, T_hk1      Compute η_hk2, u_hk2, v_hk2
                 Compute η_h, u_h, v_h
        Set      η_h0 = η_h, u_h0 = u_h, v_h0 = v_h
End for
```

Algorithm 2:

Another method to solve the systems (11), (12) and (13), taking into account (14) :

```
Set             E^n = E^0 = η_h0 = η_0, U^n = U^0 = u_h0 = u_0, V^n = V^0 = v_h0 = v_0
Set             P^n = P_h0, Q^n = Q_h0, T^n = T_h0, P^k1 = P_hk1, Q^k1 = Q_hk1, T^k1 = T_hk1
Set             E^{n+1} = η_h, U^{n+1} = u_h, V^{n+1} = v_h
Set             E^{k1} = η_hk1, U^{k1} = u_hk1, V^{k1} = v_hk1, E^{k2} = η_hk2, U^{k2} = u_hk2, V^{k2} = v_hk2
Compute         A(if we want to use the mesh adaptation
                we must compute A in the for-loop time)
For t = 0 : δt : T
                Mesh adaptation, (optional)
                Update η_h0 = η_h0; u_h0 = u_h0; v_h0 = v_h0; η_h = η_h; u_h = u_h; v_h = v_h;
                (with mesh adaptation)
                Compute P_h0, Q_h0, T_h0    Compute η_hk1, u_hk1, v_hk1
                Compute P_hk1, Q_hk1, T_hk1    Compute η_hk2, u_hk2, v_hk2
                Compute A(with mesh adaptation)
                Set X = [η_h, u_h, v_h]    Compute B    Solve AX = B
        Set     η_h0 = η_h, u_h0 = u_h, v_h0 = v_h
End for
```

4 Code

In this section we will present by details all the step of the FreeFem++ code to solve (8) to (14).

4.1 Construction of the domain Ω

We note that in FreeFem++ the domain is assumed to described by its boundary that is on the left side of the boundary which is implicitly oriented by the parametrization.
Let Ω be the rectangle defined by its frontier $\partial\Omega = [-5,5]\times[-1,1]$ where his vertices are $A(-5,-1), B(5,-1), C(5,1)$ and $D(-5,1)$, so we must define the border AB, BC, CD and DA of $\partial\Omega$ by using the keyword `border` then the triangulation $\mathbf{T}_h$ of Ω is automatically generated by using the keyword `buildmesh`. Note that in FreeFem++ the automatic mesh generation is based on the Delaunay-Voronoi algorithm, cf. [LucPir98].

```
real Dx=.2; // discretization space parameter
int aa=-5,bb=5,cc=-1,dd=1;
border AB (t = aa, bb){x = t ;y = cc;label = 1;};
border BC (t = cc, dd){x = bb;y = t ;label = 2;};
border CD (t = bb, aa){x = t ;y = dd;label = 3;};
border DA (t = dd, cc){x = aa;y = t ;label = 4;};
mesh Th = buildmesh( AB(floor(abs(bb-aa)/Dx)) + BC(floor(abs(dd-cc)/Dx)) + CD(
    ➥floor(abs(bb-aa)/Dx)) + DA(floor(abs(dd-cc)/Dx)) );
```

The keyword `label` can be added to define a group of boundaries for later use (Boundary Conditions for instance). Boundaries can be referred to either by name (AB for example) or by label (1 here).

4.2 Finite Element Space

A finite element space (F.E.S) is, usually, a space of polynomial functions on elements of $\mathbf{T}_h$, triangles here, with certain matching properties at edges, vertices, ...
In our case, since we have after integrating by parts in the equation (11) a first derivative order in space, then we must use for the F.E.S. at least $\mathbb{P}_1$ of continuous piecewise linear functions. Then the F.E.S. is defined as

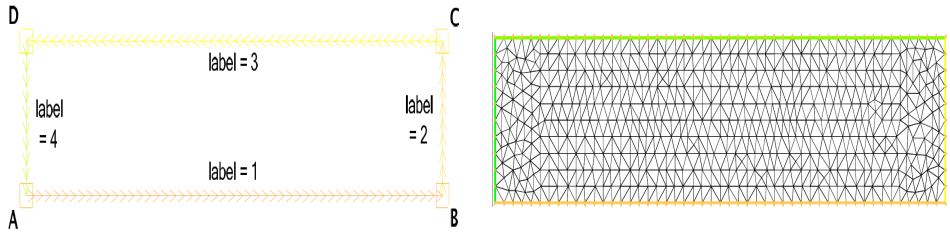

Figure 1: Plot of the border (left) and the mesh (right)

```
fespace Vh( Th, P1 );
```

In the case of Bi-Periodic Boundary Condition, they will be set in the F.E.S as

```
fespace Vh( Th, P1,periodic=[[1,x],[3,x],[2,y],[4,y]] );
```

Since η_h, u_h, v_h and $\phi_h \in V_h$, we define `etah, uh, vh, etah0, uh0, vh0, Ph0, Qh0, Th0, etahk1, etahk2, uhk1, uhk2, vhk1, vhk2, Phk1, Qhk1, Thk1, etah0pk1, uh0pk1, vh0pk1, Th0pk1, Ph0pk1, Qh0pk1, phih` as piecewise-$\mathbb{P}_1$ continuous linear functions, as

```
Vh etah, uh, vh, etah0, uh0, vh0, Ph0, Qh0, Th0, etahk1, etahk2, uhk1, uhk2,
    ➥vhk1, vhk2, Phk1, Qhk1, Thk1, etah0pk1, uh0pk1, vh0pk1, Th0pk1, Ph0pk1,
    ➥Qh0pk1, phih;
```

where

```
etah0pk1=etah0+etahk1;  uh0pk1=uh0+uhk1;    vh0pk1=vh0+vhk1;
Th0pk1=Th0+Thk1;        Ph0pk1=Ph0+Phk1;    Qh0pk1=Qh0+Qhk1;
```

4.3 Scheme parameter

Using (3), we define the parameter $\theta^2, \nu, \mu, \delta t, Tf, a, b, c$ and d where δt is the time step size and Tf is the final time of the program. These parameter will be declared in the next section depending on each numerical case.

```
real theta2, nu, mu, dt, Tf; // to be declared
real a = .5*(theta2 - 1./3.)*nu, b = .5*(theta2 - 1./3.)*(1. - nu), c = .5*(1.
    ➥ - theta2)*mu , d = .5*(1. - theta2)*(1. - mu);
```

4.4 Initial data

We define the initial data as following :

```
etah0=eta0;// to be declared
uh0=u0;// to be declared
vh0=v0;// to be declared
```

4.5 Declaration of the problems

Note that in FreeFem++ the scalar product in L^2 : $\langle ,\phi_h\rangle = \int_\Omega \cdot \phi_h$ = `int2d(Th)( *phih )` ; also we can define a macro for the gradient $\nabla u = (\partial_x(u), \partial_y(u))^T$, divergence $\nabla \cdot \begin{pmatrix} u \\ v \end{pmatrix} = \partial_x(u) + \partial_y(v)$ and for the right hand side function $\mathbf{F}(\mathbf{E},\mathbf{U},\mathbf{V},\mathbf{P},\mathbf{Q})$, $\mathbf{G}(\mathbf{E},\mathbf{U},\mathbf{V},\mathbf{T})$, $\mathbf{H}(\mathbf{E},\mathbf{U},\mathbf{V},\mathbf{T})$ defined in (7) using the keyword `macro`, that will be used in the sequence, as :

```
macro grad(u)[dx(u),dy(u)]//
macro div(u,v)(dx(u)+dy(v))//
macro F(e,u,v,p,q)(div(u,v)+dx(e)*u+e*dx(u)+dy(e)*v+e*dy(v)+a*div(p,q))//
macro G(e,u,v,t)(dx(e)+dx(u)*(u)+dx(v)*(v)+c*dx(t))//
macro H(e,u,v,t)(dy(e)+dy(u)*(u)+dy(v)*(v)+c*dy(t))//
```

We note that all the variable (`e,u,v,p,q,t`) used in the macro are dummies.
We declare the problem for $\mathbf{P}^n, \mathbf{Q}^n, \mathbf{T}^n$ defined is the system (11) as :

```
problem PH0(Ph0,phih) = int2d(Th)(Ph0*phih) + int2d(Th)(grad(uh0)'*grad(phih))
    ➥       + "Boundary Conditions of uh for Ph0";
problem QH0(Qh0,phih) = int2d(Th)(Qh0*phih) + int2d(Th)(grad(vh0)'*grad(phih))
    ➥       + "Boundary Conditions of vh for Qh0";
problem TH0(Th0,phih) = int2d(Th)(Th0*phih) + int2d(Th)(grad(etah0)'*grad(phih
    ➥))     + "Boundary Conditions of etah for Th0";
```

then the problems to find $\mathbf{E}^{k1}, \mathbf{U}^{k1}, \mathbf{V}^{k1}, \mathbf{P}^{k1}, \mathbf{Q}^{k1}, \mathbf{T}^{k1}$ of the system (12) are declared as :

```
problem ETAHK1(etahk1,phih) = int2d(Th)(etahk1*phih) + int2d(Th)(grad(etahk1)
    ➥'*grad(phih)*b) + int2d(Th)( F(etah0,uh0,vh0,Ph0,Qh0)*phih*dt ) + "
    ➥Boundary Conditions of etah for etahk1";
problem UHK1(uhk1,phih) = int2d(Th)(uhk1*phih) + int2d(Th)(grad(uhk1)'*grad(
    ➥phih)*d) + int2d(Th)( G(etah0,uh0,vh0,Th0)*phih*dt)+"Boundary Conditions
    ➥of uh for uhk1";
problem VHK1(vhk1,phih) = int2d(Th)(vhk1*phih) + int2d(Th)(grad(vhk1)'*grad(
    ➥phih)*d) + int2d(Th)( H(etah0,uh0,vh0,Th0)*phih*dt)+"Boundary Conditions
    ➥of vh for vhk1";
problem PHK1(Phk1,phih) = int2d(Th)(Phk1*phih) + int2d(Th)(grad(uhk1)'*grad(
    ➥phih))   + "Boundary Conditions of uh for Phk1" ;
problem QHK1(Qhk1,phih) = int2d(Th)(Qhk1*phih) + int2d(Th)(grad(vhk1)'*grad(
    ➥phih))   + "Boundary Conditions of vh for Qhk1" ;
problem THK1(Thk1,phih) = int2d(Th)(Thk1*phih) + int2d(Th)(grad(etahk1)'*grad(
    ➥phih)) + "Boundary Conditions of etah for Thk1" ;
```

and the problems to find $\mathbf{E}^{k2}, \mathbf{U}^{k2}, \mathbf{V}^{k2}$ of the system (13) are declared as :

```
problem ETAHK2(etahk2,phih) = int2d(Th)(etahk2*phih) + int2d(Th)(grad(etahk2)
    ➥'*grad(phih)*b) + int2d(Th)( F(etah0pk1,uh0pk1,vh0pk1,Ph0pk1,Qh0pk1)*phih
    ➥*dt)  + "Boundary Conditions of etah for etahk2";
problem UHK2(uhk2,phih) = int2d(Th)(uhk2*phih) + int2d(Th)(grad(uhk2)'*grad(
    ➥phih)*d) + int2d(Th)( G(etah0pk1,uh0pk1,vh0pk1,Th0pk1)*phih*dt) + "
    ➥Boundary Conditions of uh for uhk1";
problem VHK2(vhk2,phih) = int2d(Th)(vhk2*phih) + int2d(Th)(grad(vhk2)'*grad(
    ➥phih)*d) + int2d(Th)( H(etah0pk1,uh0pk1,vh0pk1,Th0pk1)*phih*dt) + "
    ➥Boundary Conditions of vh for vhk1";
```

Finally, to find $\mathbf{E}^{n+1}, \mathbf{U}^{n+1}, \mathbf{V}^{n+1}$ defined in the system (8), we declare the corresponding problem as :

```
problem ETAH(etah,phih) = int2d(Th)(etah*phih) - int2d(Th)(etah0*phih) - int2d
    ➥(Th)((etahk1 + etahk2) * phih /2.);
```

```
problem UH(uh,phih) = int2d(Th)(uh*phih) - int2d(Th)(uh0*phih) - int2d(Th)((
    ➥uhk1 + uhk2) * phih /2.);
problem VH(vh,phih) = int2d(Th)(vh*phih) - int2d(Th)(vh0*phih) - int2d(Th)((
    ➥vhk1 + vhk2) * phih /2.);
```

Remark: In order to make our code faster, we can use the keyword `init` in the declaration of the problem, for example :

```
problem ETAH(etah,phih,init=0) = int2d(Th)(...); // if we want to compute the
    ➥mass matrix
problem ETAH(etah,phih,init=1) = int2d(Th)(...); // if we want to use the mass
    ➥ matrix computed before
```

4.6 Solve of the problems

To solve all the problems defined above, we make a for-loop time and we call the problems by their names when we want them to be solved, then we update the data and at the end we plot the solution using the keyword `plot`.
We note that in each iteration of the for-loop a mesh adaptation will be done which depend on the error (`err`) which is the $\mathbb{P}_1$ interpolation error level, where `hmin` is the minimum edge size and `nbvx` is the maximum number of vertices generated by the mesh generator.

```
for (real t=0.;t<=T;t+=dt){
    Th=adaptmesh(Th,etah0,err=1e-4,hmin=Dx,nbvx=1e6); // we can use adaptmesh
        ➥each 10 iterations or more.
    PH0;         QH0;         TH0;
    ETAHK1;      UHK1;        VHK1;
    PHK1;        QHK1;        THK1;
    etah0pk1=etah0+etahk1;  uh0pk1=uh0+uhk1;    vh0pk1=vh0+vhk1;
    Th0pk1=Th0+Thk1;        Ph0pk1=Ph0+Phk1;    Qh0pk1=Qh0+Qhk1;
    ETAHK2;      UHK2;        VHK2;
    ETAH;        UH;          VH;
    etah0=etah; uh0=uh; vh0=vh;     //update of the data
    plot(etah0,cmm="t="+t+"sec",fill=true,value=true,dim=3);
}
```

In the case of Bi-Periodic Boundary Condition, we must build an adapted periodic mesh, so we use :

```
Th=adaptmesh(Th,etah0,err=1e-4,hmin=Dx,nbvx=1e6,periodic=[[1,x],[3,x],[2,y
    ➥],[4,y]]);
```

In order to use the second method, we build the matrix **A** before the for-loop time as:

```
varf Mass(u, phih) = int2d(Th)( u * phih );
matrix A, MASS;
MASS = Mass(Vh,Vh);
A = [[MASS, 0, 0],[0, MASS, 0],[0, 0, MASS]];
set(A,solver=GMRES); // to be set
```

Then we build the vector **B** in the for-loop time as :

```
for (real t=0.;t<=T;t+=dt){
PH0;         QH0;         TH0;
ETAHK1;      UHK1;        VHK1;
PHK1;        QHK1;        THK1;
etah0pk1=etah0+etahk1;  uh0pk1=uh0+uhk1;    vh0pk1=vh0+vhk1;
Th0pk1=Th0+Thk1;        Ph0pk1=Ph0+Phk1;    Qh0pk1=Qh0+Qhk1;
```

```
ETAHK2;     UHK2;       VHK2;
Vh B1, B2, B3, etahk1pk2D2, uhk1pk2D2, vhk1pk2D2;
real[int] B(3*Vh.ndof), X(3*Vh.ndof), X0(3*Vh.ndof), W(3*Vh.ndof);
etahk1pk2D2 = .5*etahk1 + .5*etahk2;
uhk1pk2D2 = .5*uhk1 + .5*uhk2;
vhk1pk2D2 = .5*vhk1 + .5*vhk2;
X0=[etah0[], uh0[], vh0[]];
B1[]=MASS*etahk1pk2D2[];
B2[]=MASS*uhk1pk2D2[];
B3[]=MASS*vhk1pk2D2[];
B=[B1[],B2[],B3[]];
X = A^-1*B;
W = X + X0;
[etah[], uh[], vh[]] = W;
etah0=etah; uh0=uh; vh0=vh;     //update of the data
plot(etah0,cmm="t="+t+"sec",fill=true,value=true,dim=3);
}
```

Finally, if we want to use mesh adaptation in the second method, we must compute the matrix A in the for-loop time as :

```
for (real t=0.;t<=T;t+=dt){
Th=adaptmesh(Th,etah0,err=1e-4,hmin=Dx,nbvx=1e6); // we can use adaptmesh each
    ➥ 10 iterations or more.
etah0=etah0;uh0=uh0;vh0=vh0; etah=etah;uh=uh;vh=vh; // to update our data in
    ➥the new mesh
PH0;        QH0;        TH0;
ETAHK1;     UHK1;       VHK1;
PHK1;       QHK1;       THK1;
etah0pk1=etah0+etahk1;  uh0pk1=uh0+uhk1;    vh0pk1=vh0+vhk1;
Th0pk1=Th0+Thk1;        Ph0pk1=Ph0+Phk1;    Qh0pk1=Qh0+Qhk1;
ETAHK2;     UHK2;       VHK2;
matrix A, MASS;
MASS = Mass(Vh,Vh);
A = [[MASS, 0, 0],[0, MASS, 0],[0, 0, MASS]];
set(A,solver=GMRES); // to be set
Vh B1, B2, B3, etahk1pk2D2, uhk1pk2D2, vhk1pk2D2;
real[int] B(3*Vh.ndof), X(3*Vh.ndof), X0(3*Vh.ndof), W(3*Vh.ndof);
etahk1pk2D2 = .5*etahk1 + .5*etahk2;
uhk1pk2D2 = .5*uhk1 + .5*uhk2;
vhk1pk2D2 = .5*vhk1 + .5*vhk2;
X0=[etah0[], uh0[], vh0[]];
B1[]=MASS*etahk1pk2D2[];
B2[]=MASS*uhk1pk2D2[];
B3[]=MASS*vhk1pk2D2[];
B=[B1[],B2[],B3[]];
X = A^-1*B;
W = X + X0;
[etah[], uh[], vh[]] = W;
etah0=etah; uh0=uh; vh0=vh;     //update of the data
plot(etah0,cmm="t="+t+"sec",fill=true,value=true,dim=3);
}
```

4.7 Numerical simulations

In the sequel, we present the results of numerical simulations of the evolution of initially localized heaps of fluid of initial velocity zero. Unless specified, all computations were performed on the square $\Omega =]-40, 40[\times]-40, 40[$, a $\mathbb{P}_1$ continuous piecewise linear functions was used for the finite element space and for all the numerical simulations, we work with the space discretization $\Delta x = 0.5$ and the time step $\Delta t = 0.1$.
We deduce from the subsection 4.1 that the border $y = -40, x = 40, y = 40$ and $x = -40$ have the label `1,2,3` and `4`, respectively.

4.7.1 Rate of convergence

At the beginning, we prove in the figure below, that the RK2 time scheme considered for the BBM-BBM system is of order 2. In this example, we took zero Dirichlet homogenous Boundary Conditions for η_h, u_h and v_h on the whole boundary and we have consider the following exacts solutions :

$$\eta_{ex} = e^t \cdot \sin(\pi x) \cdot (y-1) \cdot y,$$
$$u_{ex} = e^t \cdot x \cdot \cos(3\pi x/2) \cdot \sin(\pi y),$$
$$v_{ex} = e^t \cdot \sin(\pi x) \cdot \cos(3\pi y/2) \cdot y.$$

Then, we compute the corresponding right hand side in order to obtain the L^2 norm of the error between the exact solution and the numerical one in the table below.

N	$\lvert\eta_h - \eta_{ex}\rvert_{L^2}$	$\lvert u_h - u_{ex}\rvert_{L^2}$	$\lvert v_h - v_{ex}\rvert_{L^2}$
10	0.00871494	0.0233966	0.0230945
20	0.00265707	0.00641675	0.00632314
40	0.000670301	0.00160223	0.00157848
80	0.0001817	0.000419198	0.000412791
160	4.80657e-05	0.000108456	0.000106767

Table 2: L^2 norm of the error for η, u, v.

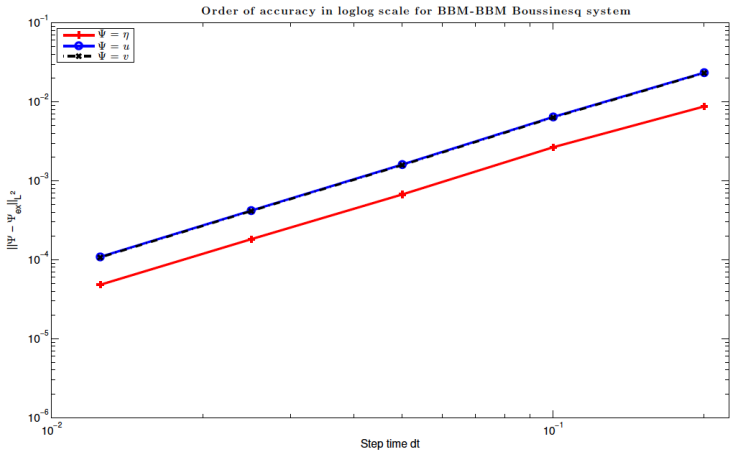

Figure 2: Rate of convergence for BBM-BBM.

4.7.2 Computation time

In this section, we consider the BBM-BBM Boussinesq system on the domain $\Omega =]-40, 40[\times]-40, 40[$ with $\mathbb{P}_1$ continuous piecewise linear functions, the space discretization $\Delta x = 0.5$, the time step $\Delta t = 0.1$ and as initial data $\eta_{h0}(x, y) = 0.2e^{-(x^2+y^2)/5}$, $u_{h0}(x, y) = v_{h0}(x, y) = 0$ with zero Dirichlet homogenous Boundary Conditions for η_h, u_h and v_h on the whole boundary.
In order to solve this system, we will show the time comparison of different method:

- **M1** to solve **Algorithm 1** without using adaptmesh technique and the keyword `init`.
- **M1init** to solve **Algorithm 1** using the keyword `init` and without using adaptmesh technique.
- **M1A-4** to solve **Algorithm 1** using adaptmesh technique with `err=1e-4` and without the keyword `init`.
- **M1A-2** to solve **Algorithm 1** using adaptmesh technique with `err=1e-2` and without the keyword `init`.
- **M2** to solve **Algorithm 2** without using adaptmesh technique and the keyword `init`.
- **M2init** to solve **Algorithm 2** using the keyword `init` and without using adaptmesh technique.
- **M2A-4** to solve **Algorithm 2** using adaptmesh technique with `err=1e-4` and without the keyword `init`.
- **M2A-2** to solve **Algorithm 2** using adaptmesh technique with `err=1e-2` and without the keyword `init`.

We present in Table 3, the time of computation in second at time $T = 10s$ using all the different method cited before. All computation was made on a Macbook OS X, Intel core 2 Duo (CPU), 4Go (Memory), 2 Ghz (Processor).

M1	**M1init**	**M1A-4**	**M1A-2**
1555.48	722.507	115.485	45.3462
M2	**M2init**	**M2A-4**	**M2A-2**
1217.01	619.462	99.5689	39.3365

Table 3: Comparison of computation time for the different method used to solve the BBM-BBM system.

We note that, without using the mesh adaptation technique, we have the same result for all the computed solution, so we can see from Table 3 that the best method to use is the **M2init**.
In other hand, using the mesh adaptation technique, we can remark from Table 3 that the computation time is bester then other method, but unfortunately, we have a little difference between the computed solution, that we plot in Figure 3 the square of L^2 norm of the error between the computed solution using the **M1** method and the one computed with the **M1A-8**, **M1A-6**, **M1A-4**, **M1A-2** methods where we have `err=1e-8`, `err=1e-6`, `err=1e-4`, `err=1e-2`, respectively vs the time till $T = 30s$. We also plot in Figure 4 the mean of all the square of L^2 norm of the error computed for different mesh adaptation method.

We can remark from this result that we have the same result using **M1A-8**, **M1A-6**, **M1A-4** method and the mean error between the solution computed with these method and the computed one using the **M1** is of order 10^{-5} and we can see the large time difference.

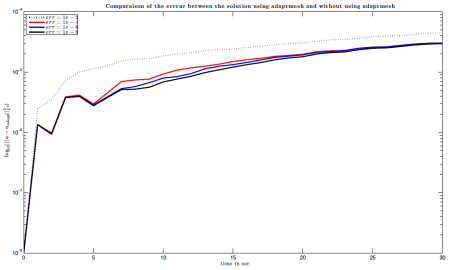

Figure 3: Comparaison of the erreur between the solution.

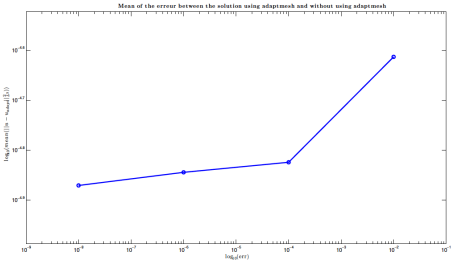

Figure 4: Mean of the erreur between the solution.

4.7.3 Reflection of expanding symmetric waves at two boundaries of the BBM-BBM Boussinesq system

In Figures 5,6 we show the reflection from two parts of the boundary of an expanding symmetric wave of the BBM-BBM Boussinesq system where $a = c = 0$ and $b = d = 1/6$. For this experiment we used as initial data the functions $\eta_{h0}(x, y) = .2e^{-(x^2+y^2)/5}, u_{h0}(x, y) = v_{h0}(x, y) = 0$. We used zero Neumann Boundary Conditions for η_h on the whole boundary, zero Dirichlet data for u_h and v_h on $x = -40$ and $y = 40$ (where we have the wall), and zero Neumann boundary data for u_h and v_h on $x = 40$ and $y = -40$. The expanding waves are reflected from the $x = -40$ and $y = 40$ parts of the boundary.
We note that in Figure 5 we show the effect of the mesh adaptation following the evolution of η_h in time and in Figure 6 we show the propagation of the solution η_h.

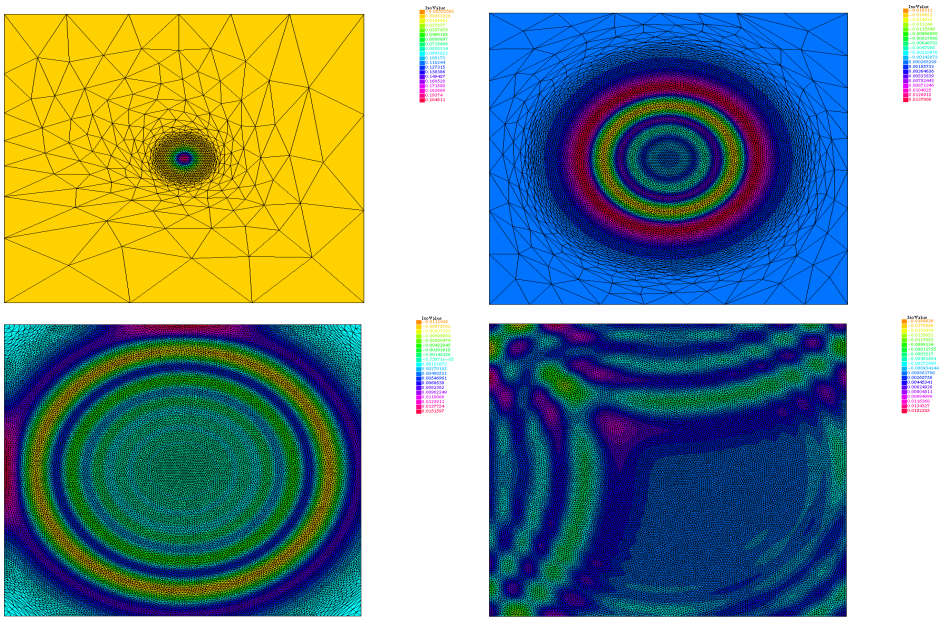

Figure 5: Plot of the solution and the mesh to the BBM-BBM Boussinesq system where $a = c = 0$ and $b = d = 1/6$ for different time $t = \{0.1, 20, 40, 70\}$

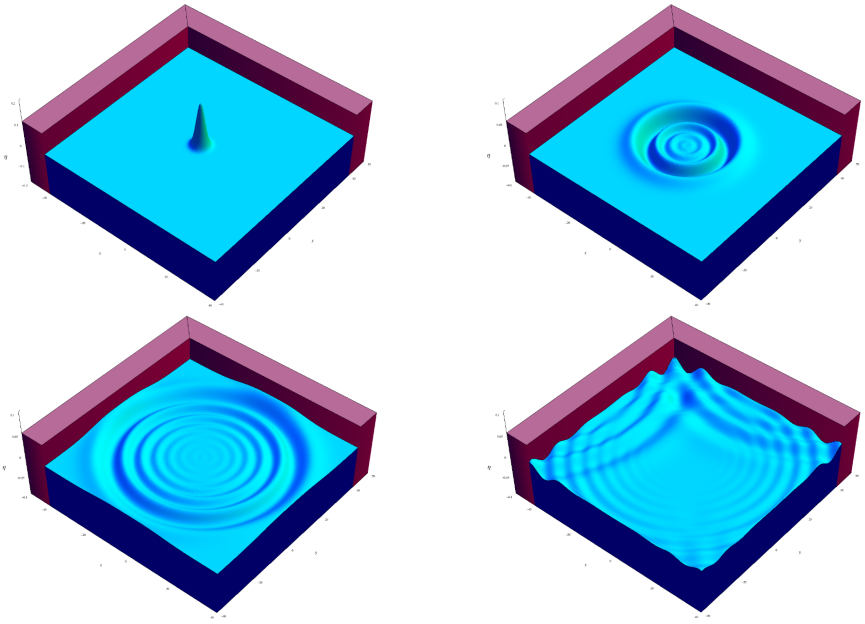

Figure 6: Solution of the BBM-BBM Boussinesq system where $a = c = 0$ and $b = d = 1/6$ for different time $t = \{0, 20, 40, 70\}$

4.7.4 Expanding symmetric waves under the KdV-KdV Boussinesq system

In Figure 7, we present the evolution of the η_h profile emanating from the radially symmetric initial data $\eta_{h0}(x, y) = .5e^{-(x^2+y^2)/5}, u_{h0}(x, y) = v_{h0}(x, y) = 0$, under the KdV-KdV Boussinesq system where $a = c = 1/6$ and $b = d = 0$. We used Bi-Periodic Boundary Conditions for η_h, u_h and v_h and we work with the time step $\Delta t = 0.001$. We remark here that with these Bi-Periodic Boundary Conditions for η, u and v and their derivatives, in addition by integrating the equations in the system (2) on the hole domaine, we deduce the following mass conservation : $(Id - b\Delta)\int_\Omega \eta_t = 0$ and the relations $(Id - d\Delta)\int_\Omega u_t = 0$, $(Id - d\Delta)\int_\Omega v_t = 0$. Hence :

$$\int_\Omega \eta = cte = \int_\Omega \eta_0, \qquad \int_\Omega u = cte = \int_\Omega u_0, \qquad \int_\Omega v = cte = \int_\Omega v_0. \tag{15}$$

In other hand, numerically, we see that these defined quantity are well conserved over time and we have :

$$\int_\Omega \eta = cte = \int_\Omega \eta_0 = 7.84527, \qquad \int_\Omega u = cte = \int_\Omega u_0 = 0, \qquad \int_\Omega v = cte = \int_\Omega v_0 = 0.$$

We can see in Figure 7 a small amplitude periodic profile (ripples) which has been observed in [BDM07] (in the case of 1D KdV-KdV system).
Other simulation of different Boussinesq system can be found in [Sad11].

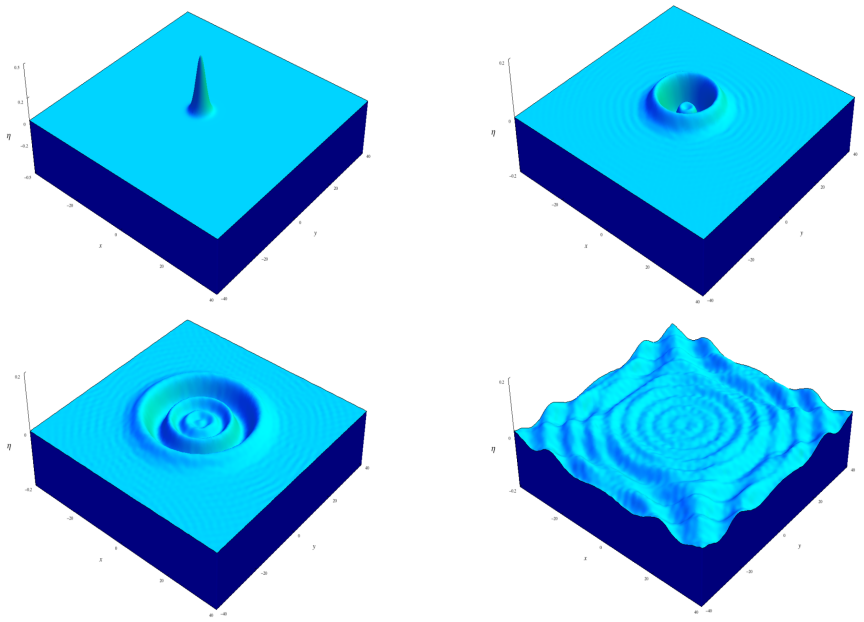

Figure 7: Solution of the KdV-KdV Boussinesq system where $a = c = 1/6$ and $b = d = 0$ for different time $t = \{0, 10, 20, 60\}$

5 Conclusion

We have presented a numerical approach with FreeFem++ to solve the Boussinesq system with a flat bottom, we validated our code and establishes the adequacy of the chosen finite element discretization by comparing the results with those of Mitsotakis and *al.*
We have established also the feasibility of simulating complex equations of hydrodynamics as Boussinesq systems with FreeFem++.
Using this approach, we can consider the case of a variable bottom (in space and/or in time), see [Sad11]. As a feature we address the simulation of *Tsunamis* with our approach, by including realistic data (bathymetry, generation of tsunami waves).

Acknowledgements : This work is supported by the regional program "Appui à lémergence" of the Region Picardie. I would like to thank my PhD advisor Jean-Paul Chehab (LAMFA, Amiens), Denys Dutykh (LAMA, Savoie), Gloria Faccanoni (IMATH, Toulon), Kirill Pichon Gostaf (LJLL, Paris), Frédéric Hecht (LJLL, Paris), Antoine Le Hyaric (LJLL, Paris), Youcef Mammeri (LAMFA, Amiens), Olivier Pantz (CMAP, Paris) and Dimitrios Mitsotakis (IMA, University of Minnesota) for fruitful discussions and remarks.

References

[AzeBoucIvoIseMoh08] Pascal Azerad, Frederic Bouchette, Benjamin Ivorra, Damien Isebe et Mohammadi. Shape optimization of geotextile tubes for sandy beach protection. *Int. J. Numer. Meth. Engng*, Vol. 74, pp. 1262-1277, 2008. PDF. 1

[AzeBoucIvoIseMoh08] Pascal Azerad, Frederic Bouchette, Damien Isebe et Bijan Mohammadi. Optimal shape design of defense structures for minimizing short wave impact. *Coastal Engineering*, Vol. 55, pp. 35-46, 2008. PDF. 1

[BonaChenSau04] Jerry Lioyd Bona, Min Chen, et Jean-Claude Saut. Boussinesq equations and other systems for small amplitude long waves in nonlinear dispersive media: II. the nonlinear theory. *Nonlinearity*, 17, no3, 925-952, 2004. PDF. 2

[BDM07] J. L. Bona, V. A. Dougalis and D. E. Mitsotakis. *Numerical solution of KdV-KdV systems of Boussinesq equations I: The numerical scheme and generalized solitary waves. Math. Comput. Simulation*, 74:214-228, 2007. PDF 14

[BreSco94] Suzanne C. Brenner and L. Ridgway Scott. The Mathematical Theory of Finite Element Methods. *Springer-Verlag*, New York, 1994. PDF. 3

[Chen09] Min Chen. Numerical investigation of a two-dimensional Boussinesq system. *Discrete and Continuous Dynamical Systems*, Vol 23, no4, 1169-1190, April 2009. PDF. 1, 3

[ChenGou09] Min Chen et Olivier Goubet. Long-time asymptotic behavior of two-dimensional dissipative Boussinesq systems. *Discrete and Continuous Dynamical Systems Series S*, Vol 2, no1, 37-53, March 2009. PDF. 1, 1, 3, 3, 3

[Dem91] Jean-Pierre Demailly. Analyse numérique et équations différentielles. *Presses Universitaires de grenoble*, 1991. PDF. 3

[DiaDut07] Frédéric Dias et Denys Dutykh. Dissipative Boussinesq equations. *C. R. Mécanique*, 335 , 559-583, 2007. PDF. 1

[DouMitSau07] Vassilios Dougalis, Dimitrios Mitsotakis et Jean-Claude Saut. On some Boussinesq systems in two space dimensions: theory and numerical analysis. *M2AN*, no. 5, 825-854, Vol 41, 2007. PDF. 1, 1, 1, 1, 3, 3, 3, 2, 3,

[DouMitSau09] Vassilios Dougalis, Dimitrios Mitsotakis et Jean-Claude Saut. On initial-boundary value problems for a Boussinesq system of BBM-BBM type in a plane domain. *AIMS' Journal*, Vol 23, 2009. PDF. 1, 1, 3, 3

[DouMitSau10] Vassilios Dougalis, Dimitrios Mitsotakis et Jean-Claude Saut. Boussinesq systems of Bona-Smith type on plane domain : Theory and Numerical Analysis. *Journal of Scientific Computing*, Vol. 44, no2, pp. 109-135, 2010. PDF. 3, 3, 3

[DutKatMit11] Denys Dutykh, Theodoros Katsaounis et Dimitrios Mitsotakis. Finite volume schemes for dispersive wave propagation and runup. *Computational Physics*, 230 (8), 3035 - 3061, 2011. PDF. 1

[LinPilSau11] Félipe Linares, Didier Pilod et Jean-Claude Saut. Well-posedness of strongly dispersive two-dimensional surface waves Boussinesq. *ArXiv*:1103.4159v2, 11 Apr 2011. PDF. 3

[LucPir98] Brigitte Lucquin and Olivier Pironneau. *Introduction to Scientific Computing*. *Wiley*, 1998. PDF. 6

[Mit09] 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. PDF. 1

[Sad11] Georges Sadaka. *Etude mathématique et numérique d'équations d'ondes aquatiques amorties. Thèse de l'Université de Picardie Jules Verne - Amiens*, 2011. PDF. 2, 14, 15

[WalBer02] Mark Walkley et Martin Berzins. A finite element method for the two-dimensional extended Boussinesq equations. *Int. J. Numer. Meth. Fluids*, 39:865-885, 2002. PDF. 2, 3