HAL
open science

Solving Shallow Water flows in 2D with FreeFem++ on structured mesh

Georges Sadaka

To cite this version:

Georges Sadaka. Solving Shallow Water flows in 2D with FreeFem++ on structured mesh. 2012. hal-00715301

HAL Id: hal-00715301

https://hal.science/hal-00715301

Submitted on 6 Jul 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Solving Shallow Water flows in 2D with FreeFem++ on structured mesh

Georges Sadaka

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

Abstract - FreeFem++ is an open source platform to solve partial differential equations numerically, based on finite element methods. What we will present in this work is inspired from the finite volume method for hyperbolic problem in order to solve it with finite element method in FreeFem++. In particular we will see how it can be possible, with an approach of finite element and finite volume method, to solve the Shallow Water equations (Saint - Venant system) in 2D with topographic source term. More precisely, we will define the quantity types of finite volume such as up-wind scheme, HLL flux, well balanced scheme with hydrostatic reconstruction that are used in the numerical resolution of the variational problem.

To this end we will consider a particular rectangular structured isotropic domain using $\mathbb{P}_1$ finite element space. We note that in other situations, for example with unstructured anisotropic mesh or with other type of finite element space, an approach of Discontinuous Galerkin method could be used as in [11]. We note also that, there is a FreeVol project in order to introduce Finite Volume technic in FreeFem++ for hyperbolic PDEs by Frédéric Hecht *et al.* which is an ongoing work.

Keywords: Shallow Water equations, finite volume method, finite element method, well-balanced scheme, hydrostatic reconstruction, FreeFem++.

1 Introduction

The Shallow Water equations introduced in [12] is very commonly used for the numerical simulation of various geophysical shallow-water flows [2], such as rivers [4], lakes or coastal areas [6], rainfall runoff on agricultural fields [9], or even atmosphere or avalanches [1, 8] when completed with appropriate source terms.

The numerical study of damped hydrodynamic surface wave propagation is a very challenging problem through the phenomena that represent (giant waves, *Tsunamis*, ...). The shallow water equations are routinely used to predict a tsunami wave Run-up and, subsequently, constitute inundation maps for tsunami hazard areas.

Our motivation in this paper comes from the oscillation that we see during our simulations, using the Boussinesq system in 2D, with the propagation of *Tsunamis* near the coast where we are not in the big deep water wave regime (cf. [17]).

To solve this problem, we must consider another regime for small deep water wave for example the Shallow Water equations in order also to see the inundation of the *Tsunamis*. Many works have consider this problem from a finite volume point of view to construct approximate solutions for the

hyperbolic conservation laws where they build a well balanced scheme which preserve the positivity of the solution during the Run-up (cf. [3, 7, 13, 16]).

The two-dimensional Saint-Venant system for Shallow Water writes as follows :

$$\partial_t \mathbf{U} + \partial_x F(\mathbf{U}) + \partial_y G(\mathbf{U}) = S(\mathbf{U}), \tag{1}$$

where

$$\mathbf{U} = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}, F(\mathbf{U}) = \begin{pmatrix} hu \\ hu^2 + \frac{g}{2}h^2 \\ huv \end{pmatrix}, G(\mathbf{U}) = \begin{pmatrix} hv \\ huv \\ hv^2 + \frac{g}{2}h^2 \end{pmatrix} \text{ and } S(\mathbf{U}) = \begin{pmatrix} 0 \\ -gh\partial_x z_b \\ -gh\partial_y z_b \end{pmatrix}. \tag{2}$$

Here u and v are the scalar components in the horizontal x, y directions of the depth-averaged velocity, h is the local water depth and $g > 0$ denotes the gravity constant. $\mathbf{U}$ is the vector for the conservative variables, $F(\mathbf{U})$ and $G(\mathbf{U})$ stand for the flux functions respectively along the x and y directions and $S(\mathbf{U})$ represents the bed slope source term with the bed slope $\nabla z_b = \begin{pmatrix} \partial_x z_b \\ \partial_y z_b \end{pmatrix}$.

This model is very robust, being hyperbolic and admitting an entropy inequality (related to the physical energy $\mathbf{E}$, see [18] for more details)

$$\partial_t \tilde{\mathbf{E}}(\mathbf{U}, z_b) + \partial_x \left[u \cdot \left(\tilde{\mathbf{E}}(\mathbf{U}, z_b) + \frac{gh^2}{2} \right) \right] + \partial_y \left[v \cdot \left(\tilde{\mathbf{E}}(\mathbf{U}, z_b) + \frac{gh^2}{2} \right) \right] \leq 0 \tag{3}$$

where

$$\mathbf{E}(\mathbf{U}) = (hu^2 + hv^2)/2 + \frac{g}{2}h^2 \text{ and } \tilde{\mathbf{E}}(\mathbf{U}, z_b) = \mathbf{E}(\mathbf{U}) + hgz_b. \tag{4}$$

Another nice property is that it preserves the steady state of a lake at rest

$$h + z_b = Cst, \qquad u = 0, \qquad v = 0. \tag{5}$$

When solving numerically (1), it is very important to be able to preserve these steady states at the discrete level and to accurately compute the evolution of small deviations from them, because the majority of real-life applications resides in this flow regime. It is a difficult problem identified for the first time in [5] and the scheme which preserves this type of equilibrium are called well balanced since [14].

The paper is organized as follows. In Section 2 we remind the construction of the well-balanced scheme with hydrostatic reconstruction. In Section 3 the HLL approximate Riemann solver is presented for the wet/dry transition problem. In Section 4, we present the corresponding CFL condition in order to have the stability of the up-wind scheme. We present in Section 5 the details of the finite volume approach in FreeFem++ by giving the corresponding code for each step. Finally, in Section 6 we present the numerical simulations starting by testing the correctness and precision of the numerical scheme using an analytical solution, then by showing the simulation of a solitary wave crossing an empty pond.

2 Well-balanced scheme with hydrostatic reconstruction

The system (1) can be written as :

$$\partial_t \mathbf{U} + \mathbf{A}(\mathbf{U})\partial_x \mathbf{U} + \mathbf{B}(\mathbf{U})\partial_y \mathbf{U} = S(\mathbf{U}), \tag{6}$$

where

$$\mathbf{A}(\mathbf{U}) = \begin{pmatrix} 0 & 1 & 0 \\ -u^2 + gh & 2u & 0 \\ -uv & v & u \end{pmatrix} \text{ and } \mathbf{B}(\mathbf{U}) = \begin{pmatrix} 0 & 0 & 1 \\ -uv & v & u \\ -v^2 + gh & 0 & 2v \end{pmatrix},$$

then

$$\det(\mathbf{A}(\mathbf{U}) - \lambda I) = (u - \lambda) \cdot (\lambda - u - \sqrt{gh}) \cdot (\lambda - u + \sqrt{gh})$$

and

$$\det(\mathbf{B}(\mathbf{U}) - \lambda I) = (v - \lambda) \cdot (\lambda - v - \sqrt{gh}) \cdot (\lambda - v + \sqrt{gh}).$$

Thus, the eigenvalues of the linearized convection matrices $\mathbf{A}(\mathbf{U})$ and $\mathbf{B}(\mathbf{U})$ are, respectively :

$$\lambda_1 = u, \quad \lambda_2 = u - \sqrt{gh}, \quad \lambda_3 = u + \sqrt{gh} \text{ and } \lambda_1^* = v, \quad \lambda_2^* = v - \sqrt{gh}, \quad \lambda_3^* = v + \sqrt{gh} \tag{7}$$

We remark that for $h > 0$, all the eigenvalues are distinct and the system is strictly hyperbolic. Finite volume schemes for hyperbolic systems consist in using an up-winding of the fluxes. The two-dimensional semi-discrete finite volume formulation of system (1) with hydrostatic reconstruction, is given by :

$$\mathbf{U}_{i,j}^{n+1} - \mathbf{U}_{i,j}^{n} + \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i+1/2,j}^{n} - \mathbf{F}_{i-1/2,j}^{n}\right) + \frac{\Delta t}{\Delta y}\left(\mathbf{G}_{i,j+1/2}^{n} - \mathbf{G}_{i,j-1/2}^{n}\right) = \Delta t \cdot \mathbf{S}_{i,j}^{n} \tag{8}$$

where

$$\begin{aligned} \mathbf{F}_{i+1/2,j}^{n} &= \mathcal{F}\left(\mathbf{U}_{i+1/2,j,L}, \mathbf{U}_{i+1/2,j,R}\right), & \mathbf{F}_{i-1/2,j}^{n} &= \mathcal{F}\left(\mathbf{U}_{i-1/2,j,L}, \mathbf{U}_{i-1/2,j,R}\right), \\ \mathbf{G}_{i,j+1/2}^{n} &= \mathcal{G}\left(\mathbf{U}_{i,j+1/2,D}, \mathbf{U}_{i,j+1/2,U}\right), & \mathbf{G}_{i,j-1/2}^{n} &= \mathcal{G}\left(\mathbf{U}_{i,j-1/2,D}, \mathbf{U}_{i,j-1/2,U}\right), \end{aligned} \tag{9}$$

and

$$\mathbf{S}_{i,j}^{n} = \begin{pmatrix} 0 \\ \frac{1}{\Delta x}\left(\frac{g}{2}h_{i+1/2,j,L}^2 - \frac{g}{2}h_{i-1/2,j,R}^2\right) \\ \frac{1}{\Delta y}\left(\frac{g}{2}h_{i,j+1/2,D}^2 - \frac{g}{2}h_{i,j-1/2,U}^2\right) \end{pmatrix}. \tag{10}$$

Here $\mathcal{F}$ and $\mathcal{G}$ are the numerical flux to be define in the sequel and the evaluation of the cell interface of the conservative variables are defined as :

$$\mathbf{U}_{i-1/2,j,L} = \begin{pmatrix} h_{i-1/2,j,L} \\ h_{i-1/2,j,L} \cdot u_{i-1,j} \\ h_{i-1/2,j,L} \cdot v_{i-1,j} \end{pmatrix}, \mathbf{U}_{i-1/2,j,R} = \begin{pmatrix} h_{i-1/2,j,R} \\ h_{i-1/2,j,R} \cdot u_{i,j} \\ h_{i-1/2,j,R} \cdot v_{i,j} \end{pmatrix},$$

$$\mathbf{U}_{i+1/2,j,L} = \begin{pmatrix} h_{i+1/2,j,L} \\ h_{i+1/2,j,L} \cdot u_{i,j} \\ h_{i+1/2,j,L} \cdot v_{i,j} \end{pmatrix}, \mathbf{U}_{i+1/2,j,R} = \begin{pmatrix} h_{i+1/2,j,R} \\ h_{i+1/2,j,R} \cdot u_{i+1,j} \\ h_{i+1/2,j,R} \cdot v_{i+1,j} \end{pmatrix},$$

$$\mathbf{U}_{i,j-1/2,D} = \begin{pmatrix} h_{i,j-1/2,D} \\ h_{i,j-1/2,D} \cdot u_{i,j-1} \\ h_{i,j-1/2,D} \cdot v_{i,j-1} \end{pmatrix}, \mathbf{U}_{i,j-1/2,U} = \begin{pmatrix} h_{i,j-1/2,U} \\ h_{i,j-1/2,U} \cdot u_{i,j} \\ h_{i,j-1/2,U} \cdot v_{i,j} \end{pmatrix},$$

and

$$\mathbf{U}_{i,j+1/2,D} = \begin{pmatrix} h_{i,j+1/2,D} \\ h_{i,j+1/2,D} \cdot u_{i,j} \\ h_{i,j+1/2,D} \cdot v_{i,j} \end{pmatrix}, \mathbf{U}_{i,j+1/2,U} = \begin{pmatrix} h_{i,j+1/2,U} \\ h_{i,j+1/2,U} \cdot u_{i,j+1} \\ h_{i,j+1/2,U} \cdot v_{i,j+1} \end{pmatrix}.$$

We use the notation U for Up, L for Left, R for Right and D for Down (cf. Figure 1).
This ansatz is motivated by a balancing requirement, as follows. For nearly hydrostatic flows one has $u \ll \sqrt{gh}$ and $v \ll \sqrt{gh}$. In the associated asymptotic limit the leading order water height $\underline{h}$ adjusts so as to satisfy the balance of momentum flux and momentum source terms, *i.e.*

$$\partial_x\left(\frac{g\underline{h}^2}{2}\right) = -\underline{h}g\partial_x z_b \text{ and } \partial_y\left(\frac{g\underline{h}^2}{2}\right) = -\underline{h}g\partial_y z_b. \tag{11}$$

Integrating over the (i,j)-th grid cell, we obtain an approximation to the net source term as

$$-\int_{x_{i-1/2,j}}^{x_{i+1/2,j}} \underline{h}g\partial_x z_b \mathrm{dx} = \frac{g}{2}\underline{h}_{i+1/2,j,L}^2 - \frac{g}{2}\underline{h}_{i-1/2,j,R}^2$$

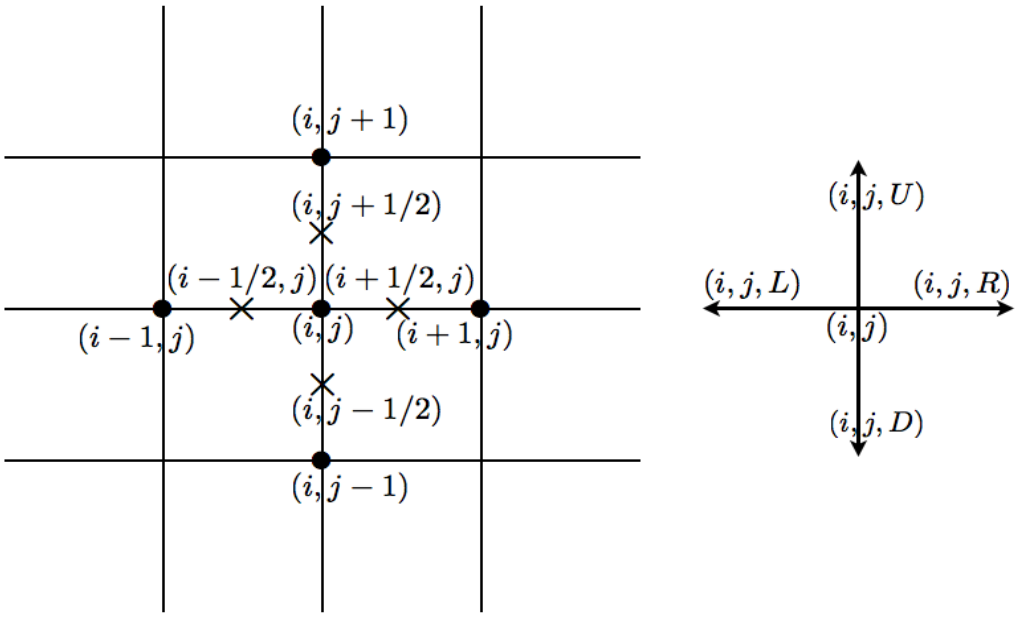

Figure 1: Notations

and

$$-\int_{y_{i,j-1/2}}^{y_{i,j+1/2}} \underline{h} g \partial_y z_b \mathrm{dy} = \frac{g}{2}\underline{h}^2_{i,j+1/2,D} - \frac{g}{2}\underline{h}^2_{i,j-1/2,U}.$$

Thus we are able to represent locally the cell-averaged source term as the discrete gradient of the hydrostatic momentum flux, and this motivates the source term discretization in (10).
It is obvious now that any hydrostatic state is maintained exactly if, for such a state, the momentum fluxes in (8) and the locally reconstructed heights satisfy

$$\mathbf{F}^{n,hu}_{i+1/2,j} = \frac{g}{2}h^2_{i+1/2,j,L} = \frac{g}{2}h^2_{i+1/2,j,R} \text{ and } \mathbf{G}^{n,hv}_{i,j+1/2} = \frac{g}{2}h^2_{i,j+1/2,D} = \frac{g}{2}h^2_{i,j+1/2,U}.$$

This is the motivation for (9), which gives this property if for hydrostatic states we have

$$\mathbf{U}_{i+1/2,j,L} = \mathbf{U}_{i+1/2,j,R} = \begin{pmatrix} h_{i+1/2,j,L} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} h_{i+1/2,j,R} \\ 0 \\ 0 \end{pmatrix}$$

and

$$\mathbf{U}_{i,j+1/2,D} = \mathbf{U}_{i,j+1/2,U} = \begin{pmatrix} h_{i,j+1/2,D} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} h_{i,j+1/2,U} \\ 0 \\ 0 \end{pmatrix}.$$

The hydrostatic balance in (11) is equivalent to the "lake at rest" equation (5), so that the reconstruction of the leading order heights is straightforward,

$$\underline{h}_{i+1/2,j,L} = h_{i,j} + z_{b_{i,j}} - z_{b_{i+1/2,j}}, \qquad \underline{h}_{i+1/2,j,R} = h_{i+1,j} + z_{b_{i+1,j}} - z_{b_{i+1/2,j}} \tag{12}$$

and

$$\underline{h}_{i,j+1/2,D} = h_{i,j} + z_{b_{i,j}} - z_{b_{i,j+1/2}}, \qquad \underline{h}_{i,j+1/2,U} = h_{i,j+1} + z_{b_{i,j+1}} - z_{b_{i,j+1/2}}. \tag{13}$$

An important challenge is to design a scheme that robustly captures dry regions where $h \equiv 0$. In order to ensure non negativity of the water height even when cells begin to "dry out", we need first to perform a truncation of the leading order heights in (12,13),

$$h_{i+1/2,j,L/R} = \max\left(0, \underline{h}_{i+1/2,j,L/R}\right) \text{ and } h_{i,j+1/2,D/U} = \max\left(0, \underline{h}_{i,j+1/2,D/U}\right). \tag{14}$$

Next, the evaluation of the cell interface height $z_{b_{i+1/2,j}}$ and $z_{b_{i,j+1/2}}$ has to be done in a quite subtle way. Our construction, combined with a centered value of $z_{b_{i+1/2,j}}$ and $z_{b_{i,j+1/2}}$, is not stable. We rather take an *upwind* evaluation of the form

$$z_{b_{i+1/2,j}} = \max\left(z_{b_{i,j}}, z_{b_{i+1,j}}\right) \text{ and } z_{b_{i,j+1/2}} = \max\left(z_{b_{i,j}}, z_{b_{i,j+1}}\right). \tag{15}$$

With these choices, we ensure that $0 \leq h_{i+1/2,j,L} \leq h_{i,j}$ and $0 \leq h_{i+1/2,j,R} \leq h_{i+1,j}$ also that $0 \leq h_{i,j+1/2,D} \leq h_{i,j}$ and $0 \leq h_{i,j+1/2,U} \leq h_{i,j+1}$, and we prove below that this property ensures the non negativity requirement.

3 HLL approximate Riemann solver

We implement here the HLL (Harten, Lax and van Leer) approximate Riemann solver proposed in [15]. When working on dry bed problems the HLL approach highlights a better behavior, avoiding unidimensionalisation effects on the flow field. Such a reason leads to the choice of the HLL Riemann solver for the development of the presented code.

The application of this approach to the two-dimensional scheme gives the following expression for the numerical flux:

$$\mathcal{F}(\mathbf{a},\mathbf{b}) = \begin{cases} F(\mathbf{a}) & \text{if } 0 < s_L, \\ F(\mathbf{b}) & \text{if } s_R < 0, \\ \dfrac{s_R \cdot F(\mathbf{a}) - s_L \cdot F(\mathbf{b}) + s_L \cdot s_R \cdot (\mathbf{b}-\mathbf{a})}{s_R - s_L} & \text{if } s_L \leq 0 \leq s_R, \end{cases} \tag{16}$$

where

$$s_L = \inf_{\mathbf{c}=\mathbf{a},\mathbf{b}} \left(\inf_{i\in\{1,2,3\}} (\lambda_i(\mathbf{c})) \right) \text{ and } s_R = \sup_{\mathbf{c}=\mathbf{a},\mathbf{b}} \left(\sup_{i\in\{1,2,3\}} (\lambda_i(\mathbf{c})) \right), \tag{17}$$

and

$$\mathcal{G}(\mathbf{a},\mathbf{b}) = \begin{cases} G(\mathbf{a}) & \text{if } 0 < s_D, \\ G(\mathbf{b}) & \text{if } s_U < 0, \\ \dfrac{s_U \cdot G(\mathbf{a}) - s_D \cdot G(\mathbf{b}) + s_D \cdot s_U \cdot (\mathbf{b}-\mathbf{a})}{s_U - s_D} & \text{if } s_D \leq 0 \leq s_U, \end{cases} \tag{18}$$

where

$$s_D = \inf_{\mathbf{c}=\mathbf{a},\mathbf{b}} \left(\inf_{i\in\{1,2,3\}} (\lambda_i^*(\mathbf{c})) \right) \text{ and } s_U = \sup_{\mathbf{c}=\mathbf{a},\mathbf{b}} \left(\sup_{i\in\{1,2,3\}} (\lambda_i^*(\mathbf{c})) \right). \tag{19}$$

4 Numerical stability

Explicit schemes require a careful selection of the time step to fulfill stability requirements. Classically, the time step needs to be restricted in such a way that no interaction is possible between waves from different cells during each time step. Courant, Friedrichs and Lewy defined a stability criterion for fully explicit schemes given by CFL < 1 where CFL is known as the Courant number. Various definitions of this number have been proposed, leading to different time step restrictions. As in [7] , we choose here to define the time step as follows:

$$\Delta_t \leq CFL \frac{\min(\Delta_x, \Delta_y)}{\max\left(|u_L| + c_L; |u_R| + c_R; |v_D| + c_D; |v_U| + c_U \right)}; \tag{20}$$

where

$$u_L = u_{i+1/2,j,L}; h_L = h_{i+1/2,j,L}; c_L = \sqrt{g \cdot h_L}; u_R = u_{i+1/2,j,R}; h_R = h_{i+1/2,j,R}; c_R = \sqrt{g \cdot h_R}$$

and

$$v_D = v_{i,j+1/2,D}; h_D = h_{i,j+1/2,D}; c_D = \sqrt{g \cdot h_D}; v_U = v_{i,j+1/2,U}; h_U = h_{i,j+1/2,U}; c_U = \sqrt{g \cdot h_U}.$$

5 Finite volume approach in FreeFem++

In this section, we present an approach in order to use finite volume method in FreeFem++. To this end, we must restrict our study to a rectangular domain ABCD with a structured isotropic mesh (cf. Figure 2). We must also use a $\mathbb{P}_1$ finite element space because in this case the number of degree of freedom is the same that the number of vertex and then, using these hypothesis, we can have access to each value of the vertex by using in FreeFem++ : `X[](i)` where `i` is the number of degree of freedom.

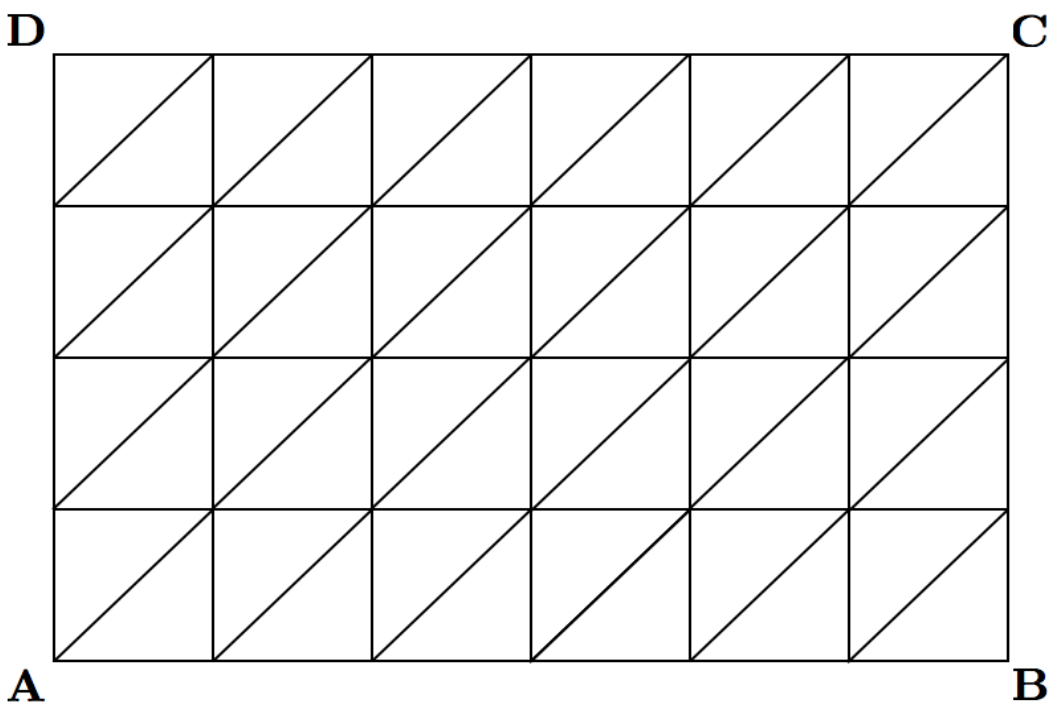

Figure 2: Structured isotropic mesh.

5.1 Access to the nodes

We remind that in FreeFem++, to build a rectangular domain with isotropic mesh we use :

```
mesh Th=square(M,N,[x,y]); // build a square with M point on x
   ➥direction and N point on y direction
mesh Th1=movemesh(Th,[x+1,y*2]); // translate the square
   ➥]0,1[*]0,1[ to a rectangle ]1,2[*]0,2[
```

To have access to the nodes, we will use the numbering defined in Figure 3 and we remind that in FreeFem++, the numbering of degree of freedom to a rectangular isotropic domain starts from the down left vertex (by 0) to the upper right one (by `Vh.ndof-1`).

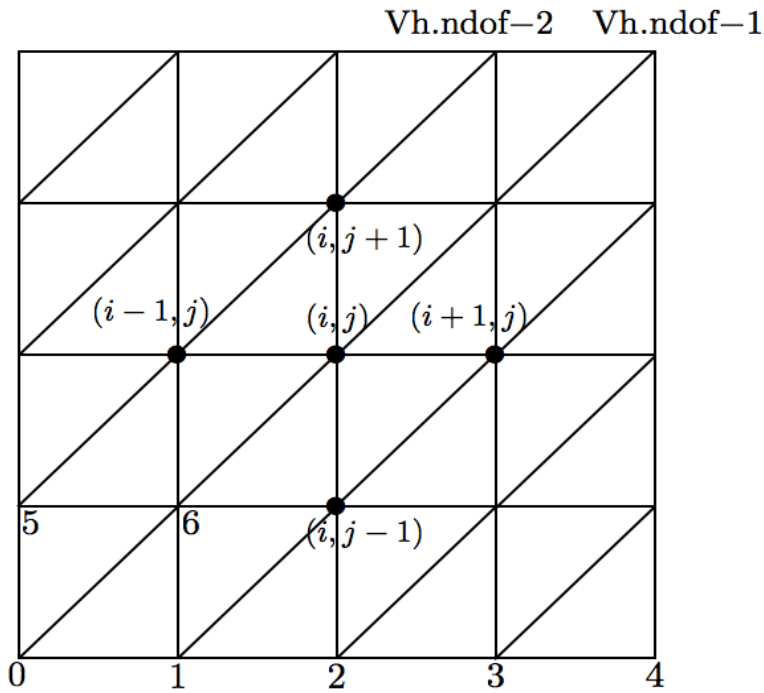

Figure 3: Numbering of the nodes (degree of freedom).

Since we discretize the x-direction with M point and the y-direction with N points, we have `Vh.ndof=(N+1)*(M+1)` and we can remark that the position of the vertex A, B, C and D have the coordinates `A(Th(0).x,Th(0).y)`, `B(Th(M).x,Th(M).y)`, `C(Th(Vh.ndof-M-1).x,Th(Vh.ndof-M-1).y)` and `D(Th(Vh.ndof-1).x,Th(Vh.ndof-1).y)`; and the value of the finite element space at these points can be found using `A[](0), B[](M), C[](Vh.ndof-M-1)` and `D[](Vh.ndof-1)`.

5.2 Boundary condition

To put boundary condition on h, we can have access to the boundary of our rectangular domain ABCD using :

```
int bc=0,da=0; // counter
for (int i=0;i<Vh.ndof;i+=1){
    // =============
    // for border AB
    // =============
    if (i <= M)
        H[](i) = 1; // for example we take 1
    // =============
    // for border BC
    // =============
    if ( i == bc*(M+1)+M ){
        H[](i) = 1;
        bc+=1;
    }
    // =============
    // for border CD
    // =============
```

```
    if(i >= Vh.ndof-M-1)
        H[](i) =    1;
    // =============
    // for border DA
    // =============
    if ( i == da*(M+1) ){
        H[](i) = 1;
        da+=1;
    }
}
```

5.3 Cell interface

In order to compute $z_{b_{i+1,j}}$, $h_{i+1,j}$, $u_{i+1,j}$ and $v_{i+1,j}$, we can define a macro function `XIP1J(X)` where X will take the place of z_b, h, u and v, such as :

```
int j;
Vh Xip1j;
macro XIP1J(X)
    j=0;
    for (int i=0;i<Vh.ndof;i+=1){
        if ( i == j*(M+1)+M ){
            Xip1j[](i) = X[](i);
            j+=1;
        }
        else
            Xip1j[](i) = X[](i+1);
    }
    #X#ip1j=Xip1j;//
    XIP1J(zb); // give zbip1j
```

In the same way, we can compute $z_{b_{i-1,j}}$, $h_{i-1,j}$, $u_{i-1,j}$ and $v_{i-1,j}$, we can define a macro function `XIM1(X)` where X will take the place of z_b, h, u and v, such as :

```
Vh Xim1j;
macro XIM1J(X)
    j=0;
    for (int i=0;i<Vh.ndof;i+=1){
        if ( i == j*(M+1) ){
            Xim1j[](i) = X[](i);
            j+=1;
        }
        else
            Xim1j[](i) = X[](i-1);
    }
    #X#im1j=Xim1j;//
    XIM1J(zb); // give zbim1j
```

On the other hand, to compute $z_{b_{i,j+1}}$, $h_{i,j+1}$, $u_{i,j+1}$ and $v_{i,j+1}$, we can define a macro function `XIJP1(X)` where X will take the place of z_b, h, u and v, such as :

```
macro XIJP1(X)
    for (int i=0;i<Vh.ndof;i+=1){
        if(i<Vh.ndof-M-1)
            Xijp1[](i) = X[](i+M+1);
        else
            Xijp1[](i) = X[](i);
```

```
    }
    #X#ijp1=Xijp1;//
    XIJP1(zb); // give zbijp1
```

```
macro XIJM1(X)
    for (int i=0;i<Vh.ndof;i+=1){
        if (i<=M)
            Xijm1[](i) = X[](i);
        else if(i>M)
            Xijm1[](i) = X[](i-M-1);
    }
    #X#ijm1=Xijm1;//
    XIJM1(zb); // give zbijm1
```

Since we have $u_{i+1/2,j,L} = u_{i,j}, u_{i,j+1/2,D} = u_{i,j}, u_{i-1/2,j,R} = u_{i,j}, u_{i,j-1/2,U} = u_{i,j}, v_{i+1/2,j,L} = v_{i,j}, v_{i,j+1/2,D} = v_{i,j}, v_{i-1/2,j,R} = v_{i,j}, v_{i,j-1/2,U} = v_{i,j}, u_{i+1/2,j,R} = u_{i+1,j}, u_{i,j+1/2,U} = u_{i,j+1}$, $u_{i-1/2,j,L} = u_{i-1,j}, u_{i,j-1/2,D} = u_{i,j-1}, v_{i+1/2,j,R} = v_{i+1,j}, v_{i,j+1/2,U} = v_{i,j+1}, v_{i-1/2,j,L} = v_{i-1,j}$ and $v_{i,j-1/2,D} = v_{i,j-1}$, we can define them in FreeFem++ as :

```
Uip12jL=U; Uijp12D=U; Uim12jR=U; Uijm12U=U; Vip12jL=V; Vijp12D=V;
    ➥Vim12jR=V; Vijm12U=V;
Uim12jL=Uim1j; Uijm12D=Uijm1; Uip12jR=Uip1j; Uijp12U=Uijp1; Vim12jL
    ➥=Vim1j; Vijm12D=Vijm1; Vip12jR=Vip1j; Vijp12U=Vijp1;
```

5.4 Hydrostatic reconstruction

For the hydrostatic reconstruction terms $h_{i+1/2,j,L/R}$ and $h_{i,j+1/2,D/U}$ defined in the equation (12) to (15) we can write them as :

```
Hip12jL=H+zb-max(zb,zbip1j); Hip12jL=max(Hip12jL,0);
Hijp12D=H+zb-max(zb,zbijp1); Hijp12D=max(Hijp12D,0);

Hip12jR=Hip1j+zbip1j-max(zb,zbip1j); Hip12jR=max(Hip12jR,0);
Hijp12U=Hijp1+zbijp1-max(zb,zbijp1); Hijp12U=max(Hijp12U,0);

Him12jL=Him1j+zbim1j-max(zbim1j,zb); Him12jL=max(Him12jL,0);
Hijm12D=Hijm1+zbijm1-max(zbijm1,zb); Hijm12D=max(Hijm12D,0);

Him12jR=H+zb-max(zbim1j,zb); Him12jR=max(Him12jR,0);
Hijm12U=H+zb-max(zbijm1,zb); Hijm12U=max(Hijm12U,0);
```

5.5 CFL condition

We may write in FreeFem++ the CFL condition defined in (20) such as :

```
Vh cflL, cflR, cflD, cflU;
cflL=abs(Uip12jL)+sqrt(gravity*Hip12jL);
cflR=abs(Uip12jR)+sqrt(gravity*Hip12jR);
cflD=abs(Vijp12D)+sqrt(gravity*Hijp12D);
cflU=abs(Vijp12U)+sqrt(gravity*Hijp12U);
Vh maxLR, maxDU;
maxLR=max(cflL,cflR);    maxDU=max(cflD,cflU);
real c = max(maxLR[].max,maxDU[].max);
real Dx=Th(1).x-Th(0).x, Dy=Th(M+1).y-Th(0).y;
real dt=CFL*min(Dx,Dy)/c;
```

5.6 Flux functions and source term

The flux functions $F(\mathbf{U})$ and $G(\mathbf{U})$ defined in (2) and the source term $\mathbf{S}_{i,j}^n$ defined in (10) can be written in FreeFem++ such as :

```
macro Source(hp,hm)[0.,.5*gravity*(hp^2 - hm^2),.5*gravity*(hp^2 -
    ➥hm^2)]//
Vh SU,SHU,SHV;
SU  = Source(Hip12jL,Him12jR)[0];
SHU = Source(Hip12jL,Him12jR)[1];
SHV = Source(Hijp12D,Hijm12U)[2];

macro F(h,u,v)[h*u,h*u^2+.5*gravity*h^2,h*u*v]//

macro G(h,u,v)[h*v,h*u*v,h*v^2+.5*gravity*h^2]//
```

5.7 HLL flux

The numerical HLL flux $\mathcal{F}(\mathbf{a},\mathbf{b})$ defined in (16) and (17) can be written in FreeFem++ as :

```
macro lambda1(u)(u)//
macro lambda2(h,u)(u-sqrt(gravity*h))//
macro lambda3(h,u)(u+sqrt(gravity*h))//

Vh S1L, S2L, S3L, S1R, S2R, S3R;
real S12Lm, S3L1m, S23Lm, sL1, sL, S12LM, S3L1M, S23LM, sR1, sR;
Vh FHp12, FHUp12, FHVp12, FHm12, FHUm12, FHVm12;
Vh HFL, HUFL, HVFL, HFR, HUFR, HVFR;
Vh HGD, HUGD, HVGD, HGU, HUGU, HVGU;

macro HLLH(hL,uL,vL,hR,uR,vR,s)
  S1L = lambda1(uL);                    S1R = lambda1(uR);
  S2L = lambda2(hL,uL);                 S2R = lambda2(hR,uR);
  S3L = lambda3(hL,uL);                 S3R = lambda3(hR,uR);
  S12Lm  = min(S1L[].min,S2L[].min);    S12LM  = max(S1L[].max,S2L
     ➥[].max);
  S3L1m = min(S3L[].min,S1R[].min);     S3L1M = max(S3L[].max,S1R
     ➥[].max);
  S23Lm  = min(S2R[].min,S3R[].min);    S23LM  = max(S2R[].max,S3R
     ➥[].max);
  sL1  = min(S12Lm,S3L1m);              sR1  = max(S12LM,S3L1M);
  sL   = min(sL1,S23Lm);                sR   = max(sR1,S23LM);
  HFL=F(hL,uL,vL)[0];HUFL=F(hL,uL,vL)[1];HVFL=F(hL,uL,vL)[2];
  HFR=F(hR,uR,vR)[0];HUFR=F(hR,uR,vR)[1];HVFR=F(hR,uR,vR)[2];
  if (sL>0) {
    FH#s#12 = HFL;
    FHU#s#12 = HUFL;
    FHV#s#12 = HVFL;
  }else if (sR<0){
    FH#s#12 = HFR;
    FHU#s#12 = HUFR;
    FHV#s#12 = HVFR;
  }else if (sL<=0 & sR>=0){
    FH#s#12 = (sR*HFL-sL*HFR+sL*sR*(hR-hL))/(sR-sL);
    FHU#s#12 = (sR*HUFL-sL*HUFR+sL*sR*(hR*uR-hL*uL))/(sR-sL);
    FHV#s#12 = (sR*HVFL-sL*HVFR+sL*sR*(hR*vR-hL*vL))/(sR-sL);
  }
```

```
  //   end of the macro
HLLH(Hip12jL, Uip12jL, Vip12jL, Hip12jR, Uip12jR, Vip12jR,p);
HLLH(Him12jL, Uim12jL, Vim12jL, Him12jR, Uim12jR, Vim12jR,m);
```

Similarly, the numerical HLL flux $\mathcal{G}(\mathbf{a},\mathbf{b})$ defined in (18) and (19) can be written in FreeFem++ as :

```
macro lambda1s(v)(v)//
macro lambda2s(h,v)(v-sqrt(gravity*h))//
macro lambda3s(h,v)(v+sqrt(gravity*h))//

Vh S1D, S2D, S3D, S1U, S2U, S3U;
real S12Dm, S3D1m, S23Dm, sD1, sD, S12DM, S3D1M, S23DM, sU1, sU;
Vh GHp12,GHUp12,GHVp12,GHm12,GHUm12,GHVm12;

macro HLLG(hL,uL,vL,hR,uR,vR,s)
  S1D = lambda1s(uL);                          S1U = lambda1s(uR);
  S2D = lambda2s(hL,vL);                       S2U = lambda2s(hR,vR);
  S3D = lambda3s(hL,vL);                       S3U = lambda3s(hR,vR);
  S12Dm  = min(S1D[].min,S2D[].min);           S12DM  = max(S1D[].max,S2D
     ➥[].max);
  S3D1m = min(S3D[].min,S1U[].min);            S3D1M = max(S3D[].max,S1U
     ➥[].max);
  S23Dm  = min(S2U[].min,S3U[].min);           S23DM  = max(S2U[].max,S3U
     ➥[].max);
  sD1   = min(S12Dm,S3D1m);                    sU1   = max(S12DM,S3D1M);
  sD    = min(sD1,S23Dm);                      sU    = max(sU1,S23DM);
  HGD=G(hL,uL,vL)[0];HUGD=G(hL,uL,vL)[1];HVGD=G(hL,uL,vL)[2];
  HGU=G(hR,uR,vR)[0];HUGU=G(hR,uR,vR)[1];HVGU=G(hR,uR,vR)[2];
  if (sD>0) {
    GH#s#12  = HGD;
    GHU#s#12 = HUGD;
    GHV#s#12 = HVGD;
  }else if (sU<0){
    GH#s#12  = HGU;
    GHU#s#12 = HUGU;
    GHV#s#12 = HVGU;
  }else if (sD<=0 & sU>=0){
    GH#s#12  = (sU*HGD-sD*HGU+sD*sU*(hR-hL))/(sU-sD);
    GHU#s#12 = (sU*HUGD-sD*HUGU+sD*sU*(hR*uR-hL*uL))/(sU-sD);
    GHV#s#12 = (sU*HVGD-sD*HVGU+sD*sU*(hR*vR-hL*vL))/(sU-sD);
  }
  //   end of the macro
HLLG(Hijp12D, Uijp12D, Vijp12D, Hijp12U, Uijp12U, Vijp12U,p);
HLLG(Hijm12D, Uijm12D, Vijm12D, Hijm12U, Uijm12U, Vijm12U,m);
```

5.8 Solve the problem

Finally, to solve the problem defined by it semi-discrete finite volume formulation in (8), we define the corresponding variational problem as :
Find $\mathbf{U}_{i,j} \in V_h = \{v_h \in C^0(\bar{\Omega}); v_h|_T \in \mathbb{P}_1(T), \forall T \in \mathcal{T}_h\}$, such that for all $\varphi \in V_h$,

$$\int_\Omega \mathbf{U}_{i,j}^{n+1}\varphi - \int_\Omega \mathbf{U}_{i,j}^{n}\varphi + \int_\Omega \left[\frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i+1/2,j}^n - \mathbf{F}_{i-1/2,j}^n\right) + \frac{\Delta t}{\Delta y}\left(\mathbf{G}_{i,j+1/2}^n - \mathbf{G}_{i,j-1/2}^n\right) - \Delta t \cdot \mathbf{S}_{i,j}^n\right]\varphi = 0 \tag{21}$$

To this end, we can proceed in FreeFem++ as :

```
Vh u,v;
solve PH(u,v,init=op) = int2d(Th)( u*v ) - int2d(Th)( ( H - (FHp12-
    ➥FHm12)*dt/Dx - (GHp12-GHm12)*dt/Dy + SU*dt )*v );
H=u;
solve PHU(u,v,init=op) = int2d(Th)( u*v ) - int2d(Th)( ( HU - (
    ➥FHUp12-FHUm12)*dt/Dx - (GHUp12-GHUm12)*dt/Dy + SHU*dt/Dx )*v );
HU=u;
solve PHV(u,v,init=op) = int2d(Th)( u*v ) - int2d(Th)( ( HV - (
    ➥FHVp12-FHVm12)*dt/Dx - (GHVp12-GHVm12)*dt/Dy + SHV*dt/Dy )*v );
HV=u;
```

We note that in order to make our code faster, we can use the keyword `init` in the declaration of the problem, thus if `init=0` we compute the mass matrix and if `init=1` we use the mass matrix computed before.

6 Numerical simulations

We present in this section some numerical simulations, we start by the rate of convergence of the up-wind scheme in order to verify the accuracy of this scheme. Then, we show the simulation of a solitary wave which will cross an empty pond.

6.1 Rate of convergence

Many exact solutions for the Shallow Water equation (called Thacker's solutions) are given in [10, 19] in order to compute the rate of convergence of the scheme. We will choose the one with the planar surface in a paraboloid where the moving shoreline is a circle and the topography is given by :

$$z_b(x,y) = -h_0\left(1 - \frac{(x-L/2)^2 + (y-L/2)^2}{a^2}\right)$$

where $(x,y) \in \Omega = [0;L] \times [0;L]$, h_0 is the water depth at the central point of the domain for a zero elevation and a is the distance from this central point to the zero elevation of the shoreline. The free surface, which has a periodic motion and remains planar in time, are given by :

$$\begin{aligned}
h(x,y,t) &= \frac{\eta h_0}{a^2}\left[2\left(x-\frac{L}{2}\right)\cos(\omega t) + 2\left(y-\frac{L}{2}\right)\sin(\omega t)\right] - z_b(x,y),\\
u(x,y,t) &= -\eta\omega\sin(\omega t),\\
v(x,y,t) &= \eta\omega\cos(\omega t),
\end{aligned}$$

where the frequency ω is defined as $\omega = \dfrac{\sqrt{2gh_0}}{a}$ and η is a parameter.

In this simulation we consider that the analytical solution at $t = 0$ s is taken as initial condition and the parameter are defining such as $a = 1$ m, $h_0 = 0.1$ m, $\eta = 0.5$, $L = 4$ m, $g = 9.8$ and CFL$= 0.5$.

The L^2-norm of the error between the exact solution and the numerical one for the conservative variable h, hu and hv are defined as :

$$E(h,N_i) = |h_{num}(N_i) - h_{ex}(N_i)|_{L^2}, E(hu,N_i) = |(hu)_{num}(N_i) - (hu)_{ex}(N_i)|_{L^2}$$

$$\text{and } E(hv,N_i) = |(hv)_{num}(N_i) - (hv)_{ex}(N_i)|_{L^2} \text{ with } N_i = N_x = N_y = \{2^{i+4}, i = 0:1:5\};$$

and, for a fixed time $t = 3$ s, these error are given in the Table 1.

We show in Figure 4 the L^2-convergence curves for the water height h and for the flows hu and hv and we see that the rate of convergence for these three conservative variable converge to the first order which is conform with the used up-wind scheme.

We remind that the rate of convergence in space for u is :

$$r(\bullet, N_i) = \frac{\log\left(E(\bullet, N_{i-1})/E(\bullet, N_i)\right)}{\log\left(N_i/N_{i-1}\right)}, \forall i = 0:1:5$$

N_i	$E(h, N_i)$	$r(h, N_i)$	$E(hu, N_i)$	$r(hu, N_i)$	$E(hv, N_i)$	$r(hv, N_i)$
16	0.0278598	-	0.0212665	-	0.0253648	-
32	0.0187917	0.5681	0.0145818	0.5444	0.0164372	0.6259
64	0.0116788	0.6862	0.00914136	0.6737	0.0101747	0.6920
128	0.00686053	0.7675	0.00536215	0.7696	0.00599727	0.7626
256	0.00382327	0.8435	0.00297793	0.8485	0.00334075	0.8441
512	0.00204941	0.8996	0.00159311	0.9025	0.00177952	0.9087

Table 1: L^2 norm of the error and the rate of convergence for h, hu and hv.

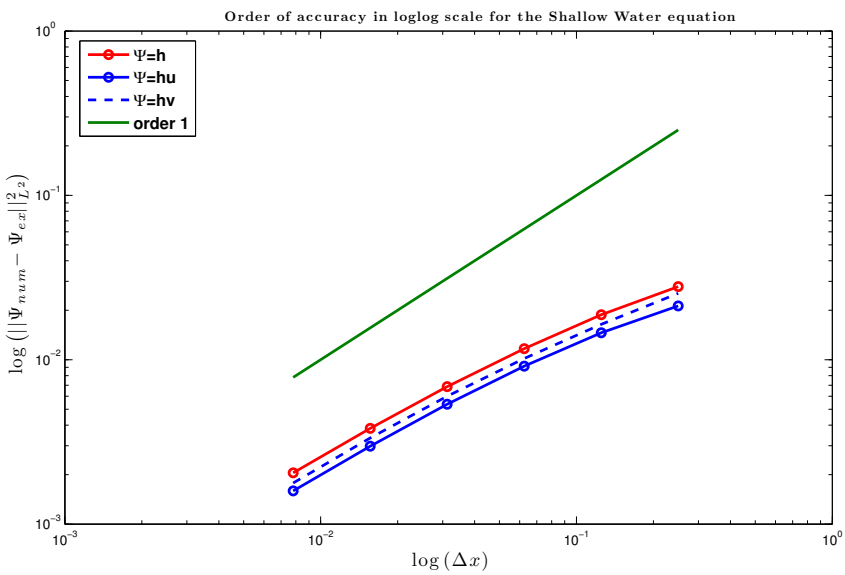

Figure 4: Rate of convergence for the Shallow Water equations.

6.2 Filling an empty pond with a solitary wave

We present in Figure 5, the simulation of the propagation of a solitary wave crossing an empty pond.

In this experience, we work in the domain $[-10; 10] \times [-5; 5]$ with $M = 150$ and $N = 75$ and we use as initial condition and parameter :

$$h(x, y, 0) = \frac{4}{\cosh(3(x+2))^2}, \; u(x, y, 0) = v(x, y, 0) = 0, \text{CFL} = 0.5, g = 9.8,$$

$$\text{and the bottom as : } z_b(x, y) = -\frac{16}{5}\exp\left(-\frac{(x-5)^2}{8} - \frac{5y^2}{8}\right) + 4\exp\left(-\frac{(x-5)^2}{2} - \frac{y^2}{4}\right) - 1.$$

We remark that, after crossing the empty pond by the solitary wave, this region is filled of water and we converge to a steady state solution.

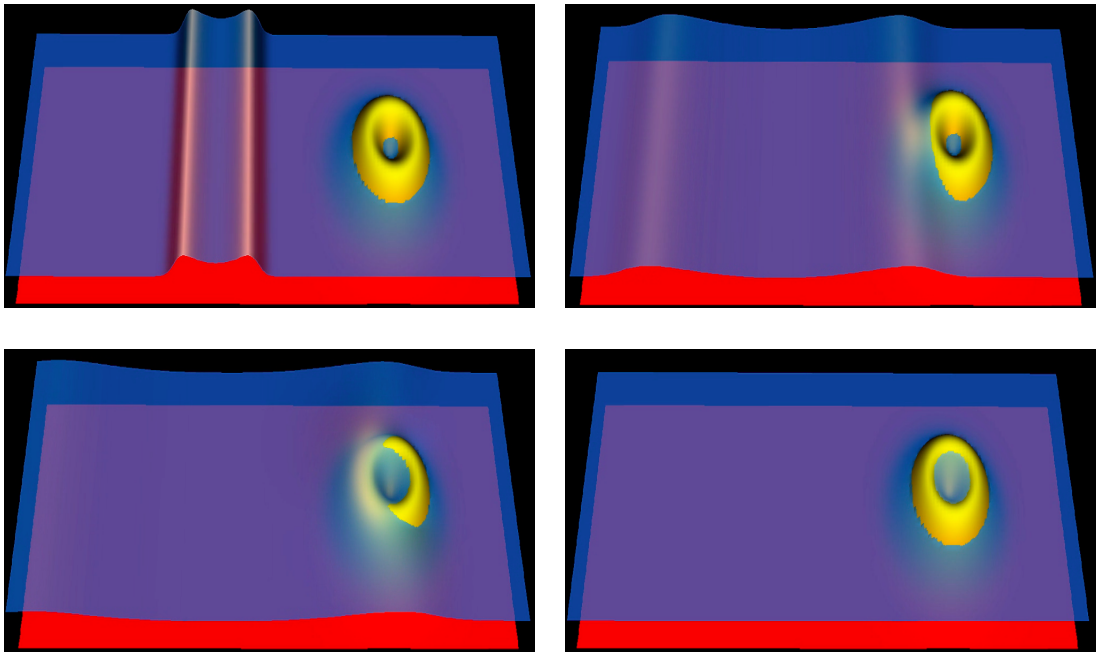

Figure 5: Propagation of the solitary wave crossing an empty pond for different time $t = \{0.2, 1, 1.5, 10\}$ s. Video available at : `http://lamfa.u-picardie.fr/sadaka/Runup_SW2D_Gaussian.mpeg`.

7 Conclusion and perspectives

In the present article we provided a detailed description of the FreeFem++ code, designed for solving Shallow Water equation using an approach of finite element - finite volume method in order to simulate the Run-up for the inundation of *Tsunamis* waves. Special attention was payed to the finite volume method used for the Shallow Water equations especially for the wet/dry transition problem and described in Sections 2, 3 and 4. In Section 5, we present the details of the code for the finite volume approach in FreeFem++. The overall performance test and validation were done in Section 6.

Finally, in order to be able to simulate the whole life-cycle of a tsunami from generation to inundation with FreeFem++, a proper way to deal with this problem is to use a domain decomposition method where in the first domain the Boussinesq systems is solved as in [17] and in the second domain, the Shallow Water equation is solved as we present here in this paper.

On the other hand, solving Shallow Water flows in 2D with FreeFem++ on unstructured mesh is still an open problem, where an approach of Discontinuous Galerkin method could be used as in [11].

Acknowledgements : I would like to thank Emmanuel Audusse (LAGA-Paris 13), Olivier Delestre (LJAD, Nice), Denys Dutykh (LAMA, Savoie), Jean-Paul Chehab (LAMFA, Amiens), Youcef Mammeri (LAMFA, Amiens) and Jacques Sainte-Marie (LJLL, Paris 6) for very helpful discussions.

References

[1] Céline Acary-Robert, Didier Bresch and Denys Dutykh. *Mathematical modeling of powder-snow avalanche flows. Studies in Applied Mathematics, 127(1), 38 - 66*, 2011.

[2] Emmanuel Audusse. *Modésitation hyperbolique et analyse numérique pour les écoulements en eaux peu profondes. Thèse de l'Université de Pierre et Marie Curie - Paris 6*, 2004.

[3] Emmanuel Audusse and Marie-Odile Bristeau. *A well-balanced positivity preserving "second-order" scheme for shallow water flows on unstructured meshes. JCP, Volume 206, Issue 1, Pages 311-333*, 2005.

[4] Emmanuel Audusse, Christophe Chalons, Olivier Delestre, Nicole Goutal, Jacques Sainte-Marie, Jan Giesselmann and Georges Sadaka. *Sediment transport modeling relaxation schemes for Saint-Venant - Exner and three layer models. Proceedings CEMRACS'11 (submit- ted),* 2011.

[5] Alfredo Bermudez and Maria Elena Vazquez. *Upwind methods for hyperbolic conservation laws with source terms. Computers & Fluids, 23(8) :1049 - 1071*, 1994.

[6] Philippe Bonneton and Fabien Marche. *A simple and efficient well-balanced model for 2DH bore propagation and run-up over a sloping beach. Coastal Engineering, proceedings of the 30th International Conference, page 998-1010*, 2006.

[7] Philippe Bonneton, Pierre Fabrie, Fabien Marche and Nicolas Seguin.*Evaluation of well-balanced bore-capturing schemes for 2D wetting and drying processes. Int. J. Numer. Meth. Fluids*, 53:867-894, 2007.

[8] François Bouchut, Anne Mangeney-Castelnau, Benoît Perthame and Jean-Pierre Vilotte. *A new model of Saint-Venant and Savage-Hutter type for gravity driven shallow water flows. C. R. Math. Acad. Sci. Paris, 336, no.6, 531-536*, 2003.

[9] Olivier Delestre. *Simulation du ruissellement d'eau de pluie sur des surfaces agricoles. Thèse de l'Université d'Orléans*, 2010.

[10] Olivier Delestre, Carine Lucas, Pierre-Antoine Ksinant, Frédéric Darboux, Christian Laguerre, Thi Ngoc Tuoi Vo, Francois James and Stephane Cordier. *SWASHES: a compilation of Shallow Water Analytic Solutions for Hydraulic and Environmental Studies. HAL arXiv:1110.0288v3*, 2012.

[11] Karim DjaDel, Alexandre Ern and Serge Piperno. A well-balanced Runge-Kutta Discontinuous Galerkin method for the Shallow Water Equations with flooding and drying. *Int. J. Numer. Meth. Fluids*, 58(1), 1-25, 2008.

[12] Barré de Saint-Venant. *Théorie du mouvement non permanent des eaux, avec application aux crues des rivières et à l'introduction des marées dans leur lit. Comptes Rendus des Séances de l'Académie des Sciences. Paris. 73, 147-154, 237-240*, 1871.

[13] Frédéric Dias, Denys Dutykh and Raphaël Poncet. *The VOLNA code for the numerical modeling of tsunami waves: generation, propagation and inundation. European Journal of Mechanics B/Fluids, 30(6), 598 - 615*, 2011.

[14] Joshua M. Greenberg and A. Y. LeRoux. *A well-balanced scheme for the numerical processing of source terms in hyperbolic equation. SIAM Journal on Numerical Analysis, 33 :1-16*, 1996.

[15] Ami Harten. *High resolution schemes for hyperbolic conservation laws. J. Comp. Physics, 49, 357–393*, 1983.

[16] Mario Ricchiuto. *Contributions to the development of residual discretization for hyperbolic conservation laws with application to shallow water flows. HDR dissertation*, September 2011.

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

[18] Denis Serre. *Systèmes hyperboliques de lois de conservation. Parties I et II., Diderot, Paris*, 1996.

[19] William Carlisle Thacker. *Some exact solutions to the nonlinear shallow water wave equations. Journal of Fluid Mechanics*, 107:499-508, 1981.