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

To cite this version:
<hal-00715301>

HAL Id: hal-00715301
https://hal.archives-ouvertes.fr/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/

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 $P_1$ 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:

$$\partial_t U + \partial_x F(U) + \partial_y G(U) = S(U),$$  \hfill (1)

where

$$U = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}, \quad F(U) = \begin{pmatrix} hu \\ huv + g h^2/2 \\ huv \\ hv^2 + g h^2/2 \end{pmatrix}, \quad G(U) = \begin{pmatrix} hv \\ huv \\ hv^2 + g h^2/2 \end{pmatrix}, \quad S(U) = \begin{pmatrix} 0 \\ -gh \partial_x z_b \\ -gh \partial_y z_b \end{pmatrix}. \hfill (2)$$

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. $U$ is the vector for the conservative variables, $F(U)$ and $G(U)$ stand for the flux functions respectively along the $x$ and $y$ directions and $S(U)$ represents the bed slope source term with the bed slope $\nabla z_b = (\partial_x z_b, \partial_y z_b)$.

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

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

where

$$\tilde{E}(U) = \left( hu^2 + hv^2 \right)/2 + \frac{gh^2}{2} \quad \text{and} \quad \tilde{E}(U, z_b) = E(U) + hgz_b. \hfill (4)$$

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

$$h + z_b = Cst, \quad u = 0, \quad v = 0. \hfill (5)$$

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:

$$\partial_t U + A(U)\partial_x U + B(U)\partial_y U = S(U),$$  \hfill (6)

where

$$A(U) = \begin{pmatrix} 0 & 1 & 0 \\ -u^2 + gh & 2u & 0 \\ -uv & v & u \end{pmatrix} \quad \text{and} \quad B(U) = \begin{pmatrix} 0 & 0 & 1 \\ -uv & v & u \\ -v^2 + gh & 0 & 2v \end{pmatrix}.$$
We use the notation $U$. We remark that for $h > 0$.

This ansatz is motivated by a balancing requirement, as follows. For nearly hydrostatic flows one has $u \ll \sqrt{gh}$ and $v \ll \sqrt{gh}$.

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

$$\lambda_1 = u, \quad \lambda_2 = u - \sqrt{gh}, \quad \lambda_3 = u + \sqrt{gh} \quad \lambda_4 = v, \quad \lambda_5 = v - \sqrt{gh}, \quad \lambda_6 = v + \sqrt{gh}$$

(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:

$$U_{i,j}^{n+1} - U_{i,j}^n + \frac{\Delta t}{\Delta x} \left( F_{i+1/2,j}^n - F_{i-1/2,j}^n \right) + \frac{\Delta t}{\Delta y} \left( G_{i,j+1/2}^n - G_{i,j-1/2}^n \right) = \Delta t \cdot S_{i,j}^n$$

(8)

where

$$F_{i+1/2,j}^n = F \left( U_{i+1/2,j,L}, U_{i+1/2,j,R} \right), \quad F_{i-1/2,j}^n = F \left( U_{i-1/2,j,L}, U_{i-1/2,j,R} \right),$$

$$G_{i,j+1/2}^n = G \left( U_{i,j+1/2,D}, U_{i,j+1/2,U} \right), \quad G_{i,j-1/2}^n = G \left( U_{i,j-1/2,D}, U_{i,j-1/2,U} \right),$$

(9)

and

$$S_{i,j}^n = \left( \begin{array}{c} 0 \\ \frac{1}{\Delta x} \left( \frac{2h^2}{2h_{i+1/2,j,L}} - \frac{2h^2}{2h_{i-1/2,j,R}} \right) \end{array} \right).$$

(10)

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

$$U_{i-1/2,j,L} = \left( \begin{array}{c} h_{i-1/2,j,L} \\ h_{i-1/2,j,L} \cdot u_{i-1,j} \\ h_{i-1/2,j,L} \cdot v_{i-1,j} \end{array} \right), \quad U_{i-1/2,j,R} = \left( \begin{array}{c} h_{i-1/2,j,R} \\ h_{i-1/2,j,R} \cdot u_{i-1,j} \\ h_{i-1/2,j,R} \cdot v_{i-1,j} \end{array} \right),$$

$$U_{i+1/2,j,L} = \left( \begin{array}{c} h_{i+1/2,j,L} \\ h_{i+1/2,j,L} \cdot u_{i+1,j} \\ h_{i+1/2,j,L} \cdot v_{i+1,j} \end{array} \right), \quad U_{i+1/2,j,R} = \left( \begin{array}{c} h_{i+1/2,j,R} \\ h_{i+1/2,j,R} \cdot u_{i+1,j} \\ h_{i+1/2,j,R} \cdot v_{i+1,j} \end{array} \right),$$

$$U_{i,j-1/2,D} = \left( \begin{array}{c} h_{i,j-1/2,D} \\ h_{i,j-1/2,D} \cdot u_{i,j-1} \\ h_{i,j-1/2,D} \cdot v_{i,j-1} \end{array} \right), \quad U_{i,j-1/2,U} = \left( \begin{array}{c} h_{i,j-1/2,U} \\ h_{i,j-1/2,U} \cdot u_{i,j-1} \\ h_{i,j-1/2,U} \cdot v_{i,j-1} \end{array} \right),$$

$$U_{i,j+1/2,D} = \left( \begin{array}{c} h_{i,j+1/2,D} \\ h_{i,j+1/2,D} \cdot u_{i,j+1} \\ h_{i,j+1/2,D} \cdot v_{i,j+1} \end{array} \right), \quad U_{i,j+1/2,U} = \left( \begin{array}{c} h_{i,j+1/2,U} \\ h_{i,j+1/2,U} \cdot u_{i,j+1} \\ h_{i,j+1/2,U} \cdot v_{i,j+1} \end{array} \right).$$

We use the notation $U$ for $U_p$, $L$ 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 $h$ adjusts so as to satisfy the balance of momentum flux and momentum source terms, i.e.

$$\partial_x \left( \frac{gh^2}{2} \right) = -hg\partial_x z_b \quad \text{and} \quad \partial_y \left( \frac{gh^2}{2} \right) = -hg\partial_y z_b.$$  

(11)

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

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

\[ F_{i+1/2,j,L} = \frac{g}{2} h_{i+1,j}^2 - \frac{g}{2} h_{i,j+1/2,D} \]  

and

\[ G_{i+1/2,j,R} = \frac{g}{2} h_{i+1,j}^2 - \frac{g}{2} h_{i,j+1/2,U}. \]

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

\[ U_{i+1/2,j,L} = U_{i+1/2,j,R} = \begin{pmatrix} h_{i+1/2,j,L} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} h_{i+1/2,j,R} \\ 0 \\ 0 \end{pmatrix} \]

and

\[ U_{i,j+1/2,D} = U_{i,j+1/2,U} = \begin{pmatrix} h_{i,j+1/2,D} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} h_{i,j+1/2,U} \\ 0 \\ 0 \end{pmatrix}. \]

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,

\[ h_{i+1/2,j,L} = h_{i,j} + z_{b_{i,j}} - z_{b_{i+1/2,j}}, \quad h_{i+1/2,j,R} = h_{i+1,j} + z_{b_{i+1,j}} - z_{b_{i+1/2,j}} \]  

and

\[ h_{i,j+1/2,D} = h_{i,j} + z_{b_{i,j}} - z_{b_{i,j+1/2}}, \quad h_{i,j+1/2,U} = h_{i+1,j} + z_{b_{i,j+1}} - z_{b_{i,j+1/2}}. \]
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),

\[
h_{i+1/2,j,L/R} = \max \left( 0, \frac{h_{i+1/2,j,L/R}}{\Delta t} \right) \quad \text{and} \quad h_{i,j+1/2,D/U} = \max \left( 0, \frac{h_{i,j+1/2,D/U}}{\Delta t} \right). \tag{14}
\]

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

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

With these choices, we ensure that \( 0 \leq h_{i+1/2,j,L} \leq h_{i,j} \) and \( 0 \leq h_{i,j+1/2,D} \leq h_{i,j} \) also that \( 0 \leq h_{i,j+1/2,U} \leq h_{i,j+1} \), 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:

\[
F(a,b) = \begin{cases} 
F(a) & \text{if } 0 < s_L, \\
F(b) - s_L \cdot F(a) + s_L \cdot s_R \cdot (b - a) \bigg/ (s_R - s_L) & \text{if } s_R < 0, \\
0 & \text{if } s_L \leq 0 \leq s_R,
\end{cases}
\tag{16}
\]

where

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

and

\[
G(a,b) = \begin{cases} 
G(a) & \text{if } 0 < s_D, \\
G(b) - s_D \cdot G(a) + s_D \cdot s_U \cdot (b - a) \bigg/ (s_U - s_D) & \text{if } s_U < 0, \\
0 & \text{if } s_D \leq 0 \leq s_U,
\end{cases}
\tag{18}
\]

where

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

### 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:

\[
\Delta t \leq CFL \frac{\min(\Delta x, \Delta y)}{\max(|u_L| + c_L, |u_R| + c_R, |v_D| + c_D, |v_U| + c_U)};
\tag{20}
\]
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
\[ \bar{v}_D = \bar{v}_{i,j+1/2,D}; h_D = h_{i,j+1/2,D}; c_D = \sqrt{g \cdot h_D}; \bar{v}_U = \bar{v}_{i,j+1/2,U}; h_U = h_{i,j+1/2,U}; c_U = \sqrt{g \cdot h_U}. \]

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 P1 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++ : \( X[i](i) \) where \( i \) is the number of degree of freedom.

![Figure 2: Structured isotropic mesh.](image)

5.1 Access to the nodes

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

```csharp
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 to a rectangle ]0,1[0,1[ to ]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 \( Vh.ndof-1 \)).
Since we discretize the x-direction with $M$ points 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:

```c
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
    // =============
```
```c
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 \( \text{XIP1J}(X) \) where \( X \) will take the place of \( z_b, h, u \) and \( v \), such as:

```c
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 \( \text{XIM1}(X) \) where \( X \) will take the place of \( z_b, h, u \) and \( v \), such as:

```c
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 \( \text{XIJP1}(X) \) where \( X \) will take the place of \( z_b, h, u \) and \( v \), such as:

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

\[
\begin{align*}
Uip_{12jL} &= U; Uij_{p12D} = U; Uim_{12jR} = U; Uij_{m12U} = U; Vip_{12jL} = V; Vij_{p12D} = V; \\
\bar{V}_{im_{12jL}} &= V; Vij_{m12D} = V; Vij_{p12U} = V; V_{ip12jR} = V; V_{ijp12U} = V;
\end{align*}
\]

### 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:

\[
\begin{align*}
Hp_{12jL} &= H+zb - \max(zb, z_{bip1j}); Hp_{12jL} = \max(Hp_{12jL}, 0); \\
Hp_{12jD} &= H+zb - \max(zb, z_{bijp1j}); Hp_{12jD} = \max(Hp_{12jD}, 0); \\
Hp_{12jR} &= Hp_{1j} + z_{bip1j} - \max(zb, z_{bip1j}); Hp_{12jR} = \max(Hp_{12jR}, 0); \\
Hp_{12jU} &= Hp_{1j} + z_{bijp1} - \max(zb, z_{bijp1}); Hp_{12jU} = \max(Hp_{12jU}, 0);
\end{align*}
\]

\[
\begin{align*}
Hp_{12jL} &= H+zb - \max(zb, z_{bip1j}); Hp_{12jL} = \max(Hp_{12jL}, 0); \\
Hp_{12jD} &= H+zb - \max(zb, z_{bijp1j}); Hp_{12jD} = \max(Hp_{12jD}, 0); \\
Hp_{12jR} &= H+zb - \max(zbim_{1j}, zb); Hp_{12jL} = \max(Hp_{12jL}, 0); \\
Hp_{12jD} &= H+zb - \max(zbim_{1j}, zb); Hp_{12jD} = \max(Hp_{12jD}, 0);
\end{align*}
\]

### 5.5 CFL condition

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

\[
\begin{align*}
V_{h} &= \text{abs}(U_{ip12jL}) + \sqrt{\text{gravity} \cdot Hp_{12jL}}; \\
&\text{abs}(U_{ip12jR}) + \sqrt{\text{gravity} \cdot Hp_{12jR}}; \\
&\text{abs}(V_{ip12jD}) + \sqrt{\text{gravity} \cdot Hj_{p12d}}; \\
&\text{abs}(V_{ip12jU}) + \sqrt{\text{gravity} \cdot Hj_{p12u}}; \\
V_{h} \text{ maxLR}, V_{h} \text{ maxDU}; \\
\text{maxLR} &= \max(\text{maxLR}[.], \text{maxDU}[.].\text{max}); \\
\text{real} &\ c = \max(CFL \cdot \min(Dx, Dy)/c);
\end{align*}
\]
5.6 Flux functions and source term

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

```plaintext
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 $F(a,b)$ defined in (16) and (17) can be written in FreeFem++ as:

```plaintext
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, FHU12, FHVp12, FHm12, FHUm12, FHVm12;
Vh HFL, HUFL, HVFL, HFR, HUFR, HVFR;
Vh HGD, HUGD, HVGD, HGU, HUGU, HGVU;

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

```cpp
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);
GH#s#12 = GHG;
GHU#s#12 = GHU;
GHV#s#12 = HVG;
if (sD>0) {
GH#s#12 = HGD;
GHU#s#12 = HUGD;
GHV#s#12 = HVG;
} else if (sU<0) {
GH#s#12 = HGU;
GHU#s#12 = HUGU;
GHV#s#12 = HVG;
} 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);
}
```

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 $U_{i,j} \in V_h = \{v_h \in C^0(\Omega); v_h|_T \in P_1(T), \forall T \in \mathcal{T}_h \}$, such that for all $\varphi \in V_h$,

$$
\int_{\Omega} U_{i,j}^{n+1} \varphi - \int_{\Omega} U_{i,j}^n \varphi + \int_{\Omega} \left[ \frac{\Delta t}{\Delta x} \left( F_{i+1/2,j}^n - F_{i-1/2,j}^n \right) \right] + \int_{\Omega} \left[ \frac{\Delta t}{\Delta y} \left( G_{i,j+1/2}^n - G_{i,j-1/2}^n \right) \right] - \Delta t \cdot S_{i,j}^n \varphi = 0
$$

(21)

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

```cpp
HLLG(Hijp12D, Ujp12D, Vjp12D, Hjp12U, Ujp12U, Vjp12U, p);
HLLG(Hijm12D, Ujm12D, Vjm12D, Hjm12U, Ujm12U, Vjm12U, m);
```
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:

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

\[ u(x,y,t) = -\eta \omega \sin(\omega t), \]

\[ v(x,y,t) = \eta \omega \cos(\omega t), \]

where the frequency \(\omega\) is defined as \(\omega = \sqrt{2gh_0/a}\) and \(\eta\) 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:

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

and \(E(hv, N_i) = |(hv)_{num}(N_i) - (hv)_{ex}(N_i)|_{L^2}\) 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:

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

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

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

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

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.
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 (LIAD, 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


