Efficient GPU implementation of a two waves WAF method for the two-dimensional one layer Shallow Water system on structured meshes

Marc de la Asunción[a], Manuel J. Castro[c], E. D. Fernández-Nieto[b], José M. Mantas[a], Sergio Ortega Acosta[c], José Manuel González-Vida[d]

[a] *Dpto. Lenguajes y Sistemas Informáticos, Universidad de Granada*
[b] *Dpto. Matemática Aplicada I, Universidad de Sevilla*
[c] *Dpto. Análisis Matemático, Universidad de Málaga*
[d] *Dpto. Matemática Aplicada, Universidad de Málaga*

Abstract

The numerical solutions of Shallow Water Equations are useful for applications related to geophysical flows that usually take place in large computational domains and could require real time calculation. Therefore, parallel versions of accurate and efficient numerical solvers for high performance platforms are needed to be able to deal with these simulation scenarios in reasonable times. In this paper we present an efficient CUDA implementation of a first and second order HLL methods and a two-waves TVD-WAF one. We propose to write all these methods under a common framework, such as, their CUDA implementations share the same structure. In particular, the reformulation of WAF numerical flux and the improved definition of the flux limiter allows us to obtain a more robust solver in situations like wet/dry fronts. Finally, some numerical tests are presented showing that

[☆]This research has been partially supported by the Spanish Government Research projects MTM09-11923, MTM2009-07719, TIN2007-29664-E and MTM2008-06349-C03-03. The numerical computations have been performed at the Laboratory of Numerical Methods of the University of Málaga.

WAF method is slightly slower that the rst order HLL method and twice faster than the second order HLL method, but it provides numerical results almost as accurate as the second order HLL scheme.
Keywords: Shallow water, finite volume schemes, WAF scheme, CUDA,

1. Introduction

In this paper we present an efficient implementation of HLL and WAF methods applied to the two-dimensional Shallow Water Equations (SWE in what follows) with topography.

The numerical solution of SWE is useful for several applications related to geophysical flows, such as the simulation of rivers, dam-breaks, floods, etc. The simulation of these phenomena gives place to very long lasting simulations in big computational domains and could even require real time calculation. Therefore parallel versions of accurate and efficient numerical solvers for high performance platforms are needed to be able to deal with these simulation scenarios in reasonable times.

Modern Graphics Processing Units (GPUs) offer hundreds of processing units optimized for massively performing floating point operations in parallel and have shown to be a cost-effective way to obtain a substantially higher performance in the applications related to SWE [13, 7, 21, 3] due to the high exploitable parallelism which exhibits the numerical schemes to solve SWE.

Currently most of the proposals to simulate shallow flows on GPUs are based on the CUDA programming model. A CUDA solver for one-layer system based on the first order Roe scheme is described in [1] to deal with structured regular meshes. The extension of this CUDA solver for two-layer shallow water system is presented in [2]. There also exist examples of implementations in CUDA of high order schemes to simulate one-layer

systems [7, 26, 20] and of implementations of first-order schemes for one and two-layer systems on triangular meshes [13, 3].

In order to port successfully numerical schemes to CUDA-enabled GPU platforms, it is important to take into account several characteristics of the scheme:

- The scheme must exhibit a high level of potential fine-grained data parallelism in order to make it possible a good exploitation of the GPU.

- It is interesting that the numerical scheme works suitably with single precision floating point arithmetic. Currently the number of single precision arithmetic units in CUDA-enabled GPUs are much higher than the number of double precision units. Therefore, the use of single precision arithmetic always produces a better performance.

- The memory requirements of the parallel subtasks, which results from the parallel decomposition of the computations in the numerical scheme, must be as small as possible, in order to avoid that many local data can not be stored in registers and the access to those data degrades considerably the performance.

- The data access pattern of the numerical scheme must enhance the spatial locality. A high degree of spatial locality makes it possible to exploit efficiently the configurable shared memory and the texture cache of the modern CUDA-enabled GPU device. Thus it is convenient to port numerical schemes whose stencils are compact enough and with a moderate number of points.

Concerning the numerical schemes, a first and second order HLL and a two-waves WAF schemes for solving the two-dimensional SWE have been considered here. More precisely, We extend here to two-dimensional problems the well-balanced HLL method proposed in [19] for the one-dimensional

SWE with pollutant and topography. Other possible well-balanced extension of HLL method for SWE can be derived using the hydrostatic reconstruction proposed in [4]. The 2D extension is defined by using the property of invariance by rotation of the SWE. Thus, at each edge of the mesh, a 1D projected SWE equation with a transport equation is considered. For this 1D problem the unknowns are the height of the water column, the normal discharge and the tangential discharge, that it is transported by the flow. Then, a 1D numerical HLL flux as the one defined in [19] is used.

Next, a second order scheme is also proposed using a MUSCL type reconstruction operator described in [22], but following the procedure described in [11]. The major difference between the method proposed here and the one described in [22] is that they consider a continuous reconstruction of the bottom topography, while we do not use this fact. Another difference, is that they formally redefine the SWE in terms of the free surface and not in terms of the water column in order to construct a numerical scheme that exactly preserves the water at rest solutions, while here the system is always defined in terms of the conserved variables.

Finally a natural extension to two-dimensional problems of the two-waves WAF method proposed in [18] has also been carried out. WAF method is a second order TVD method proposed by E. Toro in [28]. The second order accuracy is obtained by averaging the solution of a Riemann problem considered at each interface. To approximate this solution the HLL flux is considered. As it is well known, due to Godunov's theorem, linear schemes with high order accuracy generate spurious oscillations near large gradients of the solution. To avoid this problem, the WAF method is used with a flux limiter function, getting a non-linear TVD scheme of second order accuracy. If the limiters are set to zero the HLL method is achieved. In [29] and [30]

Toro presents the application of WAF method for the homogeneous Shallow Water and Euler equations. An extension to multidimensional systems was performed by Billet and Toro in [5]. In [23] a WAF method is presented for the two-dimensional SWE with topography. The well-balanced property is obtained in this case by applying a different technique suggested by Bradford and Sander in [6]. This technique allows to preserve water at rest but present a loss of accuracy for large wave run-up.

The extension of the WAF method that we proposed here is also defined using again the property of invariance of rotation of the SWE, using the WAF flux introduced in [18] to approximate the 1D problems. In fact, a new reformulation of the numerical flux that it is equivalent to the one given in [18] has been considered here that allows us to obtain a more robust solver in situations like wet/dry fronts. Nevertheless, the WAF method thus defined is not second order of accuracy, but it produces as accurate results as the second order HLL scheme, as we will show in the numerical experiments.

The paper is organized as follows: the next section describes the SWE and the property of invariance by rotations. Section 3 presents three finite volume schemes to solve the SWE. The main parallelism sources of these numerical schemes and their GPU implementation on structured meshes are described in sections 4 and 5, respectively. Several numerical experiments, performed to compare the GPU implementations of the schemes are shown and analyzed in Section 6. Finally, some conclusions are drawn in Section 7.

2. Equations

The motion of a layer of homogeneous non-viscous fluid is supposed here to be governed by the SWE, formulated under the form of a conservation law with source terms or balance law:

$$\frac{\partial U}{\partial t} + \frac{\partial F_1}{\partial x}(U) + \frac{\partial F_2}{\partial y}(U) = S_1(U)\frac{\partial H}{\partial x} + S_2(U)\frac{\partial H}{\partial y} + S_1^F(U) + S_2^F(U), \quad (1)$$

with $U = \begin{pmatrix} h & q_x & q_y \end{pmatrix}^T$, where $h(\boldsymbol{x},t)$ and $\boldsymbol{q}(\boldsymbol{x},t) = (q_x(\boldsymbol{x},t), q_y(\boldsymbol{x},t))$ are, respectively, the thickness and the mass-flow of the layer at the point $\boldsymbol{x} \in D \subset \mathbb{R}^2$ at time t, and they are related to the velocities $\boldsymbol{u}(\boldsymbol{x},t) = (u_x(\boldsymbol{x},t), u_y(\boldsymbol{x},t))$, by the equality: $\boldsymbol{q}(\boldsymbol{x},t) = \boldsymbol{u}(\boldsymbol{x},t)h(\boldsymbol{x},t), \quad i = 1,2$; g is the gravity and $H(\boldsymbol{x})$, the depth function measured from a fixed level of reference.

$$F_1(U) = \begin{pmatrix} q_x & \frac{q_x^2}{h} + \frac{1}{2}gh^2 & \frac{q_x q_y}{h} \end{pmatrix}^T,$$

$$F_2(U) = \begin{pmatrix} q_y & \frac{q_x q_y}{h} & \frac{q_y^2}{h} + \frac{1}{2}gh^2 \end{pmatrix}^T.$$

$S_k(U), k = 1,2$ are the source terms related to the variation of the bathymetry:

$$S_k(U) = \begin{pmatrix} 0 & gh(2-k) & gh(k-1) \end{pmatrix}^T, \ k = 1,2.$$

Finally, $S_k^F(U)$, $k = 1,2$ parameterize the friction term. Here, Manning friction law is used:

$$S_k^F(U) = \begin{pmatrix} 0 & -gh(2-k)\frac{n^2\|\boldsymbol{u}\|u_x}{h^{4/3}} & -gh(k-1)\frac{n^2\|\boldsymbol{u}\|u_y}{h^{4/3}} \end{pmatrix}^T, \ k = 1,2,$$

being n the Manning coefficient.

Let us define the Jacobians matrices of the fluxes F_k, $k = 1, 2$, $J_k(U) = \dfrac{\partial F_k}{\partial U}(U)$. Let $\eta = (\eta_x, \eta_y)$ be an arbitrary unit vector and let us define $\mathcal{A}(U, \eta) = J_1(U)\eta_x + J_2(U)\eta_y$. The eigenvalues of $\mathcal{A}(U, \eta)$ are given by

$$\lambda_1 = \boldsymbol{u} \cdot \eta - \sqrt{gh},\ \lambda_2 = \boldsymbol{u} \cdot \eta,\ \lambda_3 = \boldsymbol{u} \cdot \eta + \sqrt{gh}.$$

Let us also remark that the system (1) verifies the property of invariance by rotations. Effectively, let us define

$$T_\eta = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \eta_x & \eta_y \\ 0 & -\eta_y & \eta_x \end{pmatrix},$$

and let us denote $F_\eta(U) = F_1(U)\eta_x + F_2(U)\eta_y$, $\boldsymbol{S}(U) = (S_1(U), S_2(U))$ and $\boldsymbol{S}^F(U) = (S_1^F(U), S_2^F(U))$ then

$$F_\eta(U) = T_\eta^{-1} F_1(T_\eta U),\ T_\eta \boldsymbol{S}(U) \cdot \eta = S_1(T_\eta U),\ T_\eta \boldsymbol{S}^F(U) \cdot \eta = S_1^F(T_\eta U) \quad (2)$$

Moreover, it is easy to check that $T_\eta U$ verifies the system

$$\partial_t(T_\eta U) + \partial_\eta F_1(T_\eta U) = S_1(T_\eta U)\partial_\eta H + S_1^F(T_\eta U) + Q_{\eta^\perp}, \quad (3)$$

where $Q_{\eta^\perp} = T_\eta \Big(- \partial_{\eta^\perp} F_{\eta^\perp}(U) + \boldsymbol{S}(U) \cdot \eta^\perp \partial_{\eta^\perp} H + \boldsymbol{S}^F(U) \cdot \eta^\perp \Big)$.

3. Numerical Schemes

To discretize (1) the computational domain D is decomposed into subsets with simple geometry, called cells or finite volumes: $V_i \subset \mathbb{R}^2$. Here, structured meshes are used and therefore rectangular cells whose edges are

parallel to the Cartesian axes are considered. Let us denote by $\mathcal{T}$ the structured mesh, and by NV the number of cells.

Given a finite volume V_i, $|V_i|$ will represent its area; $N_i = (x_i, y_i) \in \mathbb{R}^2$ its center; $\mathcal{N}_i$ the set of indexes j such that V_j is a neighbour of V_i; E_{ij} the common edge of two neighbouring cells V_i and V_j, and $|E_{ij}|$ its length; $\eta_{ij} = (\eta_{ij,x}, \eta_{ij,y})$ the normal unit vector at the edge E_{ij} pointing towards the cell V_j; d_{ij} the distance between N_i and N_j. Let us remark that $d_{ij} = \Delta x$ if η_{ij} is horizontal and $d_{ij} = \Delta y$ if it is vertical. U_i^n is the constant approximation to the average of the solution in the cell V_i at time t^n provided by the numerical scheme:

$$U_i^n \cong \frac{1}{|V_i|} \int_{V_i} U(\boldsymbol{x}, t^n)\, d\boldsymbol{x}.$$

3.1. HLL scheme

Taking into account the property of invariance by rotations (see [31]), the extension to two dimensional domains of the first order HLL scheme defined in [19] can be written as

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{|V_i|} \sum_{j \in \mathcal{N}_i} |E_{ij}| \mathcal{F}_{ij}^{HLL^-}(U_i^n, U_j^n, H_i, H_j) \tag{4}$$

where $\mathcal{F}_{ij}^{HLL^-}(U_i^n, U_j^n, H_i, H_j)$ is defined as follows:

1. Let us define

$$U_{\eta_{ij}} = [h\ q_{\eta_{ij}}]^T = T_{\eta_{ij}}(U)_{[1,2]}, \text{ and } U_{\eta_{ij}^\perp} = \boldsymbol{q} \cdot \eta^\perp = T_{\eta_{ij}}(U)_{[3]},$$

   where $U_{[i_1,\cdots,i_s]}$ is the vector defined from vector U, using its i_1-th, ..., i_s-th components.

2. Let $\Phi^-_{\eta_{ij}}$ the 1D HLL flux associated to the 1D one layer shallow-water

system defined using the 1-st, 2-nd equations of system (3) where the term $Q_{\eta_{ij}^{\perp}}$ has been neglected (see [19] for more details):

$$\Phi_{\eta_{ij}}^{-} = \mathcal{D}_{ij}^{-}(U_{\eta_{ij},i}, U_{\eta_{ij},j}, H_i, H_j) + \mathcal{F}_C(U_{\eta_{ij},i}),$$

where $\mathcal{F}_C(U_{\eta_{ij},i}) = \begin{pmatrix} q_{\eta_{ij},i} & \frac{q_{\eta_{ij},i}^2}{h_i} \end{pmatrix}^T$ and $\mathcal{D}_{ij}^{-}(U_{\eta_{ij},i}, U_{\eta_{ij},j}, H_i, H_j)$ is given by

$$\mathcal{D}_{ij}^{-}(U_{\eta_{ij},i}, U_{\eta_{ij},j}, H_i, H_j) = \frac{1}{2}\left(\mathcal{RS}_{ij} - (\alpha_{ij,0}\mathcal{DU}_{ij} + \alpha_{ij,1}\mathcal{RS}_{ij})\right) \quad (5)$$

where

$$\mathcal{RS}_{ij} = \mathcal{F}(U_{\eta_{ij},j}) - \mathcal{F}(U_{\eta_{ij},i}) - \mathcal{S}_{ij}(H_j - H_i),$$

with $\mathcal{S}_{ij}(H_j - H_i) = \left(0 \; g\frac{h_i + h_j}{2}\right)^T (H_j - H_i)$, $\mathcal{F}(U_{\eta_{ij}}) = F_1(T_{\eta_{ij}}U)_{[1,2]}$ and

$$\mathcal{DU}_{ij} = \begin{pmatrix} h_j - H_j - (h_i - H_i) & q_{\eta_{ij},j} - q_{\eta_{ij},i} \end{pmatrix}^T.$$

The coefficients $\alpha_{ij,0}$ and $\alpha_{ij,1}$ are defined by

$$\alpha_{ij,0} = \frac{S_{R,ij}|S_{L,ij}| - S_{L,ij}|S_{R,ij}|}{S_{R,ij} - S_{L,ij}}, \quad \alpha_{ij,1} = \frac{|S_{R,ij}| - |S_{L,ij}|}{S_{R,ij} - S_{L,ij}}, \quad (6)$$

where $S_{R,ij}$ (respectively $S_{L,ij}$) is an approximation of the fastest (respectively slowest) wave of the 1D system (3). Here we use the approximation proposed by Davis in [16]:

$$S_{L,ij} = \min(\lambda_{\min,i}, \lambda_{\min,ij}), \quad S_{R,ij} = \max(\lambda_{\max,j}, \lambda_{\max,ij}), \quad (7)$$

where $\lambda_{\min,l} = u_{\eta_{ij},l} - c_l$ and $\lambda_{\max,l} = u_{\eta_{ij},l} + c_l$ with $c_l = \sqrt{gh_l}$ and $u_{\eta_{ij},l} = q_{\eta_{ij},l}/h_l$, $l = i, j$; $\lambda_{\min,ij} = u_{ij} - c_{ij}$ and $\lambda_{\max,ij} = u_{ij} + c_{ij}$,

where $c_{ij} = \sqrt{g(h_i + h_j)/2}$ and

$$u_{ij} = \frac{u_{\eta_{ij},i}\sqrt{h_i} + u_{\eta_{ij},j}\sqrt{h_j}}{\sqrt{h_i} + \sqrt{h_j}}.$$

3. Let us define

$$\Phi^-_{\eta^\perp_{ij}} = (\Phi^-_{\eta_{ij}})_{[1]} u^{ij}_{\eta^\perp_{ij}}, \tag{8}$$

where $u^{ij}_{\eta^\perp_{ij}}$ is defined as follows

$$u^{ij}_{\eta^\perp_{ij}} = \begin{cases} \boldsymbol{u}_i \cdot \eta^\perp_{ij} & \text{if } (\Phi_{\eta_{ij}})_{[1]} > 0 \\ \\ \boldsymbol{u}_j \cdot \eta^\perp_{ij} & \text{otherwise} \end{cases} \tag{9}$$

Let us remark that $\Phi^-_{\eta^\perp_{ij}}$ is the numerical flux associated to the 3-rd equation of system (3) where, again, the term $Q_{\eta^\perp_{ij}}$ has been neglected. Its derivation has been done following the main ideas of the HLLC method for the shallow water system introduced in [19] as $q_{\eta^\perp_{ij}}$ can be seen as a passive scalar that is advected by the flow.

4. Finally, $\mathcal{F}^{HLL^-}_{ij}$ is defined by $\mathcal{F}^{HLL^-}_{ij} = T^{-1}_{\eta_{ij}} F^-_{ij}$, where

$$F^-_{ij} = \begin{bmatrix} (\Phi^-_{\eta_{ij}})_{[1]} & (\Phi^-_{\eta_{ij}})_{[2]} & \Phi^-_{\eta^\perp_{ij}} \end{bmatrix}.$$

3.2. Two-waves TVD-WAF scheme

Let us consider now the natural extension of the two-waves TVD-WAF method introduced in [18] to 2D domains using the method of lines. WAF schemes were first introduced by Prof. E. Toro in [28] in the framework of conservative systems. They are second order accurate for one dimensional conservative systems but its extension to multidimensional problems by the method of lines is no more second order. Nevertheless, WAF method pro-

vides as good results as a multidimensional second order scheme with less computational effort as we will show in the numerical test Section. The expression of the two-waves TVD-WAF method that we use here is the following

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{|V_i|} \sum_{j \in \mathcal{N}_i} |E_{ij}| \mathcal{F}_{ij}^{WAF^-}(U_i^n, U_j^n, H_i, H_j) \tag{10}$$

where $\mathcal{F}_{ij}^{WAF^-}(U_i^n, U_j^n, H_i, H_j)$ is defined as $\mathcal{F}_{ij}^{HLL^-}(U_i^n, U_j^n, H_i, H_j)$ following the four steps previously enumerated, where the only differences between them are located in step two and three. More precisely in step two, the coefficients $\alpha_{ij,k}$, $k = 0, 1$ in (5) are now defined by

$$\begin{aligned} \alpha_{ij,0} &= (\mathcal{L}_{ij}(S_{L,ij}, \chi_{L,ij}) - \mathcal{L}_{ij}(S_{R,ij}, \chi_{R,ij}) \frac{S_{L,ij} S_{R,ij}}{S_{R,ij} - S_{L,ij}} \\ \alpha_{ij,0} &= \frac{\mathcal{L}_{ij}(S_{R,ij}, \chi_{R,ij}) S_{R,ij} - \mathcal{L}_{ij}(S_{L,ij}, \chi_{L,ij}) S_{L,ij}}{S_{R,ij} - S_{L,ij}}, \end{aligned} \tag{11}$$

where

$$\mathcal{L}_{ij}(S, \chi) = \text{sign}(S)(1 - \chi) + \frac{\Delta t}{d_{ij}} \chi \, S.$$

$\chi_{L,ij}$ (equally $\chi_{R,ij}$) is defined by

$$\chi_{L,ij} = \begin{cases} 1 & \text{if } p_{\max,L} < \beta \\ \phi(r_{L,ij}) & \text{otherwise} \end{cases} \tag{12}$$

where

$$\phi(x) = \frac{x(1+x)}{1+x^2},$$

and $r_{L,ij}$ is defined by

$$r_{L,ij} = \frac{p_{\min,L}}{p_{\max,L}}$$

where

$$p_{\min,L} = \begin{cases} \min(|e_{ij}|, |e_{ij,L}|) & \text{if } S_{L,ij} > 0 \\ \min(|e_{ij}|, |e_{ij,R}|) & \text{if } S_{L,ij} <= 0 \end{cases}$$

and

$$p_{\max,L} = \begin{cases} \max(|e_{ij}|, |e_{ij,L}|) & \text{if } S_{L,ij} > 0 \\ \max(|e_{ij}|, |e_{ij,R}|) & \text{if } S_{L,ij} <= 0 \end{cases}$$

where $e_{ij} = h_j - H_j - (h_i - H_i)$, $e_{ij,L} = h_i - H_i - (h_{i,L} - H_{i,L})$ and $e_{ij,R} = h_{j,R} - H_{j,R} - (h_j - H_j)$, where $h_{i,L}$ (respectively $h_{j,R}$) and $H_{i,L}$ (respectively $H_{j,R}$) are the water height and bottom topography corresponding to cell $V_{i,L}$ (respectively $V_{j,R}$) shown in Figure 1. The parameter β is set to d_{ij}^3.

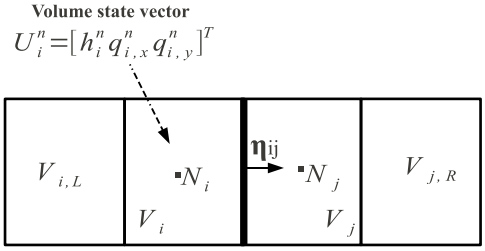


Figure 1: Stencil of the WAF method for a vertical edge

Finally, let us now define $u_{\eta_{ij}^\perp}^{ij}$ in (8) by

$$u_{\eta_{ij}^\perp}^{ij} = \frac{\boldsymbol{u}_j \cdot \eta_{ij}^\perp + \boldsymbol{u}_i \cdot \eta_{ij}^\perp}{2} - \frac{1}{2}\left(\text{sign}((\Phi_{\eta_{ij}}^-)_{[1]})(1-\chi^\perp) + \frac{\Delta t}{d_{ij}} u_{ij}^* \chi^\perp\right)(\boldsymbol{u}_j \cdot \eta_{ij}^\perp - \boldsymbol{u}_i \cdot \eta_{ij}^\perp),$$

where

$$u_{ij}^* = \text{sign}\left((\Phi_{\eta_{ij}}^-)_{[1]}\right)|u_{ij}|,$$

and $\chi^\perp$ is defined as $\chi_{L,ij}$ by setting $e_{ij} = \boldsymbol{u}_j \cdot \eta_{ij}^\perp - \boldsymbol{u}_i \cdot \eta_{ij}^\perp$, $e_{ij,L} = \boldsymbol{u}_i \cdot \eta_{ij}^\perp - \boldsymbol{u}_{i,L} \cdot \eta_{ij}^\perp$, and $e_{ij,R} = \boldsymbol{u}_{j,R} \cdot \eta_{ij}^\perp - \boldsymbol{u}_j \cdot \eta_{ij}^\perp$ and using u_{ij}^* instead of $S_{L,ij}$.

Remark 1. *Let us remark that if* $\chi_{L,ij} = \chi_{R,ij} = \chi^{\perp} = 0$*, then, the two-waves WAF scheme previously described exactly coincides with the HLL scheme given in Section 3.1.*

3.3. Second order HLL scheme

Finally, let us briefly describe the second order HLL scheme that we use here. More details about the construction of second or higher order finite volume schemes for balance laws and nonconservative problems can be found in [11].

Let us first introduce the second order reconstruction operator defined at cell V_i by

$$U_i(\boldsymbol{x}, t) = U_i(t) + (U_x(t))_i(x - x_i) + (U_y(t))_i(y - y_i), \tag{13}$$

where $U_i(t)$ is the cell average of the solution at time t provided by the numerical scheme and $(U_x(t))_i$ (respectively $(U_y(t))_i$) is a constant approximation of the partial derivative of the solution with respect to x (respectively y). Here we use the MUSCL type reconstruction described in [22] where

$$(U_x(t))_i = \text{minmod}\left(\theta\frac{U_i - U_{i,W}}{\Delta x}, \frac{U_{i,E} - U_{i,W}}{2\Delta x}, \theta\frac{U_{i,W} - U_i}{\Delta x}\right), \quad \theta \in [1, 2], \tag{14}$$

$$(U_y(t))_i = \text{minmod}\left(\theta\frac{U_i - U_{i,S}}{\Delta y}, \frac{U_{i,N} - U_{i,S}}{2\Delta y}, \theta\frac{U_{i,N} - U_i}{\Delta y}\right), \quad \theta \in [1, 2], \tag{15}$$

and

$$\text{minmod}(z_1, z_2, \ldots) = \begin{cases} \min_j z_j & \text{if } z_j > 0 \,\forall\, j, \\ \max_j z_j & \text{if } z_j < 0 \,\forall\, j, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

$U_{i,l}$, $l = E, W, S, N$ are those given in Figure 2. The parameter θ can be

used to control the amount of numerical viscosity present in the resulting scheme: larger values of θ correspond to less dissipative, but, in general, more oscillatory scheme. Here θ is fixed to 1.2.

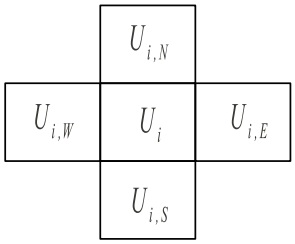


Figure 2: Stencil of the second order reconstruction.

Remark 2. *In order to obtain a exactly well-balanced scheme for water at rest solutions, first, the reconstruction of the free surface $z_s = h - H$ and the bottom topography H are computed following the previous procedure and next, the reconstruction of the water depth at cell V_i is computed as*

$$h_i(\boldsymbol{x}, t) = z_{si}(\boldsymbol{x}, t) + H_i(\boldsymbol{x}).$$

This simple procedure guarantees that the reconstruction operator is exactly well-balanced in the sense defined in [11] for the water at rest solutions. In regions close to dry areas, the previous procedure does not guarantee the positivity of the reconstructed water depth. Here we follow [22] to modify the reconstruction in order to preserve the positivity of the water depth.

Finally, the semi-implicit expression of the second order HLL scheme is

as follows:

$$
\begin{aligned}
U_i'(t) &= -\frac{1}{|V_i|}\sum_{j\in\mathcal{N}_i}|E_{ij}|\mathcal{F}_{ij}^{HLL^-}(U_{ij}^-(t),U_{ij}^+(t),H_{ij}^-,H_{ij}^+) \\
&\quad -\frac{1}{|V_i|}\int_{V_i}\begin{pmatrix} 0 \\ gh_i(\boldsymbol{x},t)(z_{sx}(t))_i \\ gh_i(\boldsymbol{x},t)(z_{sy}(t))_i \end{pmatrix}
\end{aligned} \tag{17}
$$

where $U_{ij}^-(t)$ and H_{ij}^-(respectively $U_{ij}^+(t)$ and H_{ij}^+) is the value of the reconstruction defined by (13) at cell V_i (respectively V_j) at the center of the edge E_{ij} at time t, and $(z_{sx}(t))_i$ (respectively $(z_{sy}(t))_i$) is the constant approximation of the partial derivative of free surface with respect to x (respectively y) at cell V_i provided by the reconstruction.

In order to obtain a fully discrete scheme, the integral appearing in (17) is approximated by the trapezoidal rule and a second order Runge-Kutta TVD scheme is used to approximate the time derivative (see [27]).

Remark 3. *As mentioned before, the major difference between this scheme and the one presented in [22] is that they consider a continuous reconstruction of the bottom topography, while we do not use this fact.*

Remark 4. *Let us remark that the usual CFL condition must be imposed to ensure stability of the three schemes.*

Remark 5. *The three schemes considered here are path-conservative in the sense introduced by Pares in [25], being the underlying path the segment connecting the left and the right state. Therefore, they are well-balanced for stationary solutions corresponding to water at rest.*

4. Parallelism Sources

In this section we describe the main steps of the three numerical schemes explained in section 3 and their main sources of parallelism.

4.1. Parallelism sources of the HLL and TVD-WAF schemes

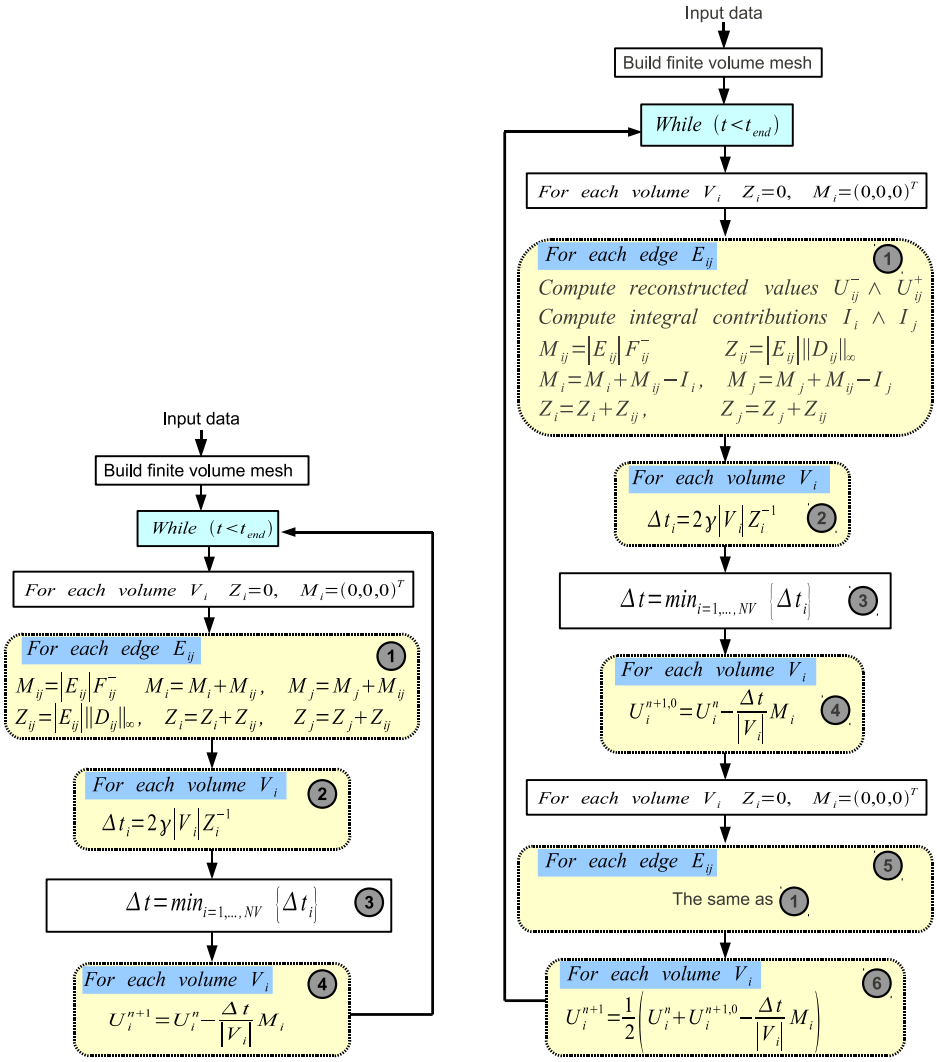
The main steps of the first order HLL and WAF schemes can be represented by a single diagram, shown in Figure 3(a). The main calculation phases are identified with circled numbers and they present a high degree of parallelism because the computation performed at each edge or volume is independent with respect to that performed at other edges or volumes.

When the finite volume mesh has been constructed, the time stepping process is repeated until the final simulation time is reached:

1. **Edge-based calculations**: For each edge E_{ij} which communicates two cells V_i and V_j, two computations are performed:

   a) Vector $M_{ij} = \mid E_{ij} \mid F_{ij}^{-} \in \mathbb{R}^3$ is computed independently for each edge and represents the contribution of an edge to the calculation of the new states of its adjacent cells V_i and V_j. This contribution must be added to the partial sums M_i and M_j associated to V_i and V_j, respectively. In the first order HLL scheme, F_{ij}^{-} corresponds to $\mathcal{F}_{ij}^{HLL^-}$ (see section 3.1), and in the WAF method, F_{ij}^{-} corresponds to $\mathcal{F}_{ij}^{WAF^-}$ (see section 3.2).

   b) The value $Z_{ij} = \mid E_{ij} \mid \parallel \mathcal{D}_{ij} \parallel_\infty$ is also computed independently for each edge, and represents the contribution of each edge to the calculation of the local Δt values of its adjacent cells V_i and V_j. This contribution must be added to the partial sums Z_i and Z_j associated to V_i and V_j, respectively.



(a) First order HLL and WAF methods (b) Second order HLL

Figure 3: Parallel algorithms

2. **Computation of the local** Δt_i: For each volume V_i, the local Δt_i is obtained by applying: $\Delta t_i = 2\gamma \left|V_i\right| Z_i^{-1}$. In the same way, the computation for each volume can be performed in parallel.
3. **Computation of** Δt: The minimum of all the local Δt_i values previously computed for each volume is obtained. This minimum Δt represents the next time step size which will be applied in the simulation.
4. **Computation of** U_i^{n+1}: The $(n+1)$-th state of each volume (U_i^{n+1}) is calculated from the n-th state and the data computed in previous phases. In the first order HLL method, the formula corresponds to equation (4), and in the WAF scheme corresponds to equation (10). This phase can also be performed in parallel.

4.2. Parallelism sources of the second order HLL scheme

Figure 3(b) shows graphically the main steps of the second order HLL scheme by using the previously introduced notation. As can be seen, this scheme also exhibits a high degree of parallelism.

When the finite volume mesh has been constructed from the input data, the time stepping is repeated, by applying a second order Runge-Kutta TVD method which consists in two stages. These two stages mainly involve the following computing phases:

1. **Edge-based calculations**: For each edge E_{ij} which communicates two cells V_i and V_j, four computations are performed:

   a) The reconstruction values, U_{ij}^-, U_{ij}^+, as well as the reconstructed topography values, H_{ij}^-, H_{ij}^+ are computed.

b) Vector $M_{ij} = |E_{ij}|\mathcal{F}_{ij}^{HLL^-}(U_{ij}^-(t), U_{ij}^+(t), H_{ij}^-, H_{ij}^+)$ is computed independently for each edge (see (17)) and added to M_i and M_j (see Figure 3(b)).

c) The value $Z_{ij} = |\,E_{ij}\,| \parallel \mathcal{D}_{ij} \parallel_\infty$ is also computed independently for each edge as in the previous subsection.

d) The contributions to the numerical approximation of the volume integral (see (17)) is computed for volumes V_i (I_i) and V_j (I_j) and deducted from M_i and M_j, respectively.

These calculations must be performed for each stage of the Runge-Kutta method.

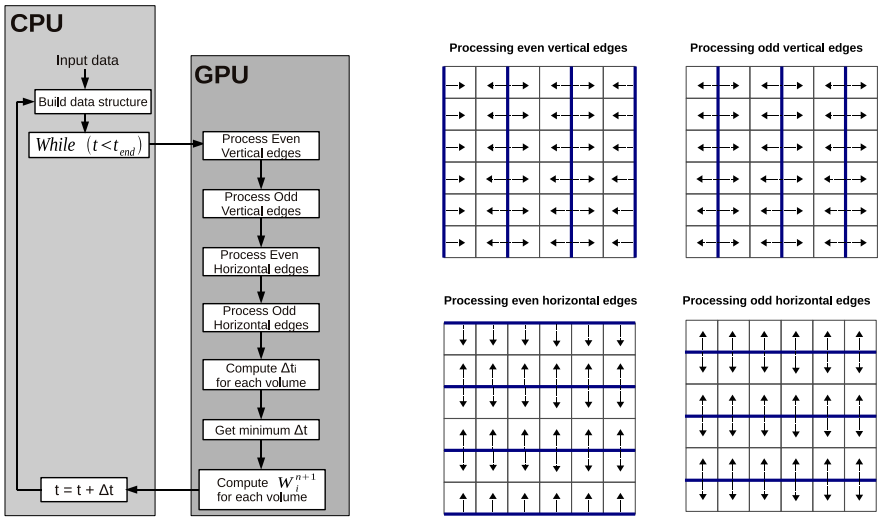
2. **Computation of the local** Δt_i: For each volume V_i, the local Δt_i is obtained in the first stage of the Runge-Kutta method.
3. **Computation of** Δt: The minimum of all the local Δt_i values previously computed for each volume is computed in the first stage.
4. **Computation of** $U_i^{n+1,0}$: For each volume, the vector $U_i^{n+1,0}$ is computed independently in the first stage from U_i^n and M_i.
5. **Computation of** U_i^{n+1}: The $(n+1)$-th state of each volume (U_i^{n+1}) is calculated from U_i^n, $U_i^{n+1,0}$ and M_i.

5. CUDA Implementation of the schemes

In this section we describe the main details of the CUDA implementation of the algorithms we have developed for one-layer shallow water systems.

5.1. Implementation of the HLL and TVD-WAF schemes

Since the structure of the first order HLL scheme is very similar to the TVD-WAF scheme, their CUDA implementations share the same structure



(a) General structure (b) Different Edge Processing steps

Figure 4: Steps in the CUDA implementation for the HLL and TVD-WAF schemes

(taking the most important details into account). In fact, the CUDA algorithm used to accelerate these schemes is a variant of the algorithm described in [1]. The general steps of this CUDA algorithm are depicted in Figure 4(a). Each processing step executed on the GPU is assigned to a CUDA kernel. Next, we describe in detail each step:

- **Build data structure**: In this step, the data structure that will be used on the GPU is built. For each volume, we store its initial state (h, q_x and q_y) and its depth H. To store these volume data, we define an array of NV `float4` elements which is stored as a 2D texture. The area of the volumes and the length of the vertical and horizontal edges are precalculated and passed to the CUDA kernels that need them.

  We can know at runtime if an edge or volume is frontier or not and the value of η_{ij} of an edge by checking the thread position in the grid.

- **Edge processing**: We use four edge processing kernel launches to process all the edges. In these edge processing kernels, each thread represents an edge, and computes the contribution of the edge to their adjacent volumes as described in Section 4. We have specific kernels to process each of four disjoint edge sets: even vertical edges, odd vertical edges, even horizontal edges, and odd horizontal edges. Here, the terms even and odd refers to the column numbers for the vertical edges and the row numbers for the horizontal edges in the finite volume mesh, assuming that they are numbered starting with 0 (see Figure 4(b)). There are several reasons to use four kernel launches:

  a) For the vertical edges, $\eta_{ij,y} = 0$, and for horizontal edges, $\eta_{ij,x} = 0$. Therefore, all the operations where these terms take part can be avoided, increasing efficiency.

  b) With this partitioning, in each kernel execution there is not any volume which is accessed by more than one thread. This is shown in Figure 4(b) which illustrates the different edge processing substeps. As a consequence, the edges (i.e. threads) does not need to synchronize each other when contributing to a particular volume. Thus, we only need one accumulator array to store the contributions of the edges (in [1] we used two accumulators for regular meshes) and an important reduction of the device memory requirements is achieved. This accumulator is an array of NV `float4` elements. In the n-th time step, the element of the accumulator which corresponds to the volume V_i stores the contribution of its neighbouring edges to the state of the volume (U_i) (a 3×1 vector M_i) and to the local Δt of that volume (a `float` value Z_i).

c) Additionally, the use of one accumulator array makes it possible to keep the positivity of the water height.

- **Compute Δt_i for each volume**: In this step, each thread represents a volume and computes the local Δt_i of the volume V_i as described in Section 4. The final Z_i value is obtained from the position corresponding to the volume V_i in the accumulator.

- **Get minimum Δt**: This step finds the minimum of the local Δt_i of the volumes by applying a reduction algorithm on the GPU. The reduction algorithm applied is the most optimized kernel of the reduction sample included in the CUDA Software Development Kit [15].

- **Compute U_i^{n+1} for each volume**: In this step, each thread represents a volume and updates the state U_i of the volume V_i as described in section 4. The final M_i value is obtained from the position corresponding to the volume V_i in the accumulator and the result is also saved in that position of the accumulator. After this step, the 2D texture containing the volume data must be updated from the accumulator array.

5.2. Implementation of the second order HLL scheme

The general steps of the CUDA implementation of the second order HLL scheme are depicted in Figure 5. Each processing step in GPU which is enclosed by a rectangle in the figure has been assigned to a CUDA kernel.

- **Build data structure**: This step is very similar to the corresponding step of the CUDA implementation for the 1st order HLL scheme excepting that an additional array with the same structure and size

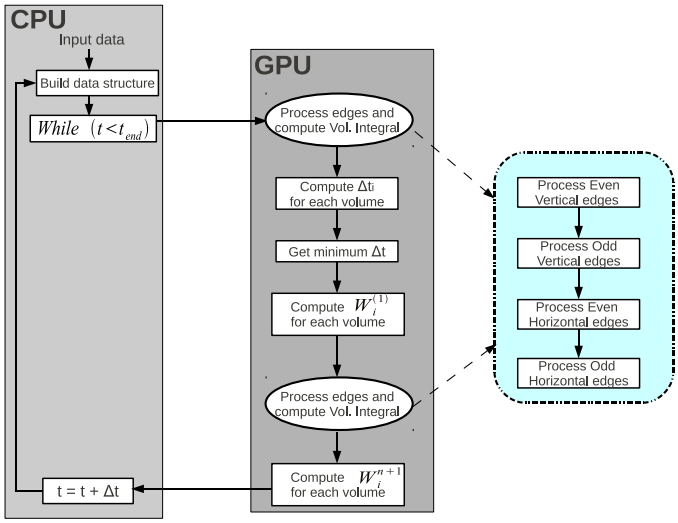


Figure 5: General steps of the CUDA implementation for the second order HLL scheme.

as the volume data array (see 5.1) must be defined to store the new obtained vector state (U^{n+1}) before finishing the current time step. This avoids to lose the initial state vector in the first Runge-Kutta stage of the next time step and allows us to use it in the second stage.

- **Edge Processing and integral computation**: In a similar way as in 5.1, this step is divided into four substeps (each substep is implemented using a different CUDA kernel as shown in Figure 5) according to the type of edge which is affected. Thus the efficiency is improved and it is possible to control the positivity of the water height In each substep, each thread represents an edge E_{ij} (which can be even vertical, odd vertical, even horizontal or odd horizontal) and computes the contribution to their adjacent volumes V_i and V_j and also computes part of the volume integral appearing in (17). The resulting contributions are also added to an accumulator array with NV `float4` elements which is stored in global memory. For that, the reconstruction values,

U_{ij}^-, U_{ij}^+, as well as the reconstructed topography values, H_{ij}^-, H_{ij}^+, must be previously computed and used to calculate the contribution of the edge E_{ij} to V_i and V_j.

Additionally, each thread also computes one of the terms which result from applying the trapezoidal rule to approximate the volume integral. The result is deducted from the previously computed contributions, before adding them in the corresponding accumulator positions.

- **Compute Δt_i for each volume and Get minimum Δt**: These two kernels are very similar to the corresponding kernel shown in 5.1.

- **Compute $U_i^{(1)}$ for each volume**: This step corresponds with the first stage of the Runge-Kutta TVD scheme and is almost identical to the step which compute W_i in 5.1.

- **Compute U_i^{n+1} for each volume**: This step implements the second stage of the Runge-Kutta scheme. This step is also very similar to the corresponding step in 5.1. After this step, U^{n+1} must be copied to the additional CUDA array bound to a 2D texture, in order use it in the second stage of the next time step.

6. Experimental Results

In this section we compare the first and second order HLL schemes and the WAF scheme described in section 3 both computationally and numerically by applying them to several test problems.

6.1. Circular dam break problem

The first test problem consists in a circular dam break in the $[-2, 2] \times [-2, 2]$ domain. The depth function is $H(x, y) = 1 - 0.8\, e^{-x^2-y^2}$ and the

Table 1: Execution times in seconds for all the meshes, programs and graphics cards.

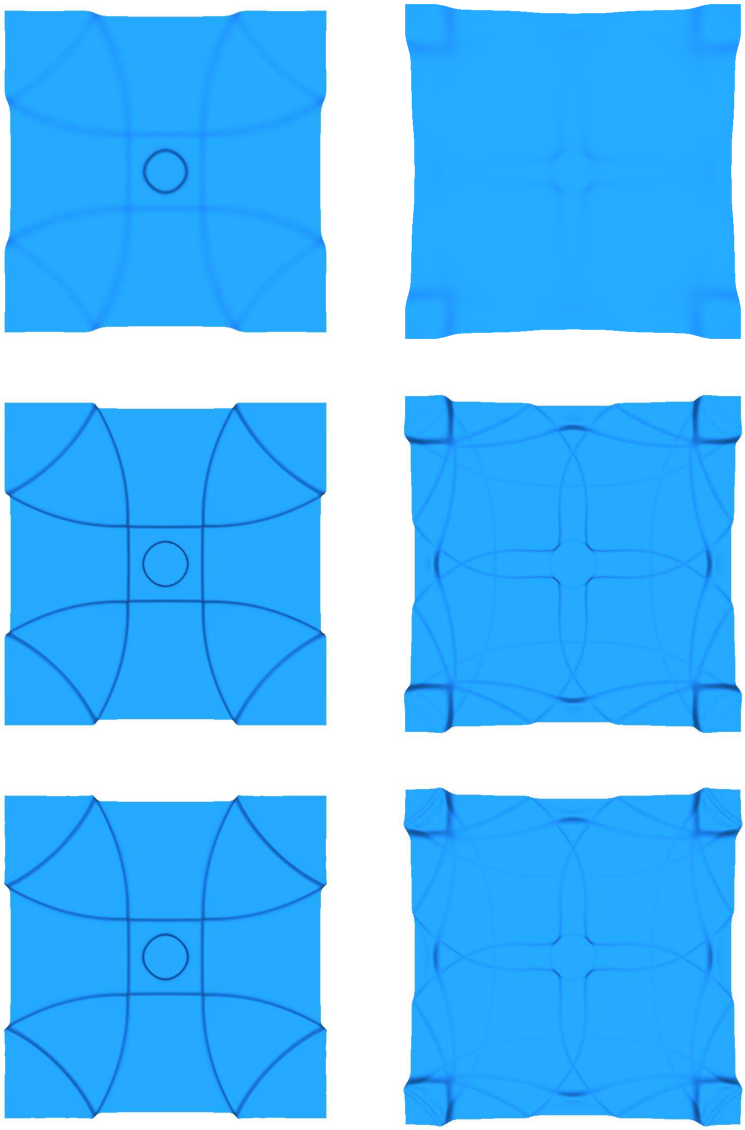
N. cells	GTX 480			GTX 580		
	1^{st} order HLL	2^{nd} order HLL	WAF	1^{st} order HLL	2^{nd} order HLL	WAF
100×100	0.0048	0.011	0.0058	0.0043	0.0097	0.0051
200×200	0.026	0.059	0.031	0.022	0.051	0.026
400×400	0.17	0.41	0.21	0.15	0.34	0.17
800×800	1.30	3.09	1.56	1.10	2.60	1.31
1600×1600	10.24	24.30	12.22	8.62	20.45	10.25
2000×2000	19.95	47.36	23.78	16.79	39.85	19.94

initial condition is $U(\boldsymbol{x}, 0) = (h(\boldsymbol{x}, 0),\ 0,\ 0)$, where $h(\boldsymbol{x}, 0) = H(x, y) + f(\boldsymbol{x})$, being

$$f(\boldsymbol{x}) = \begin{cases} 0.5 & \text{if } \sqrt{x^2 + y^2} < 0.5 \\ 0 & \text{otherwise} \end{cases}$$

All the numerical schemes are run for different mesh sizes. Simulations are carried out in the time interval [0,0.1]. CFL parameter is 0.9 and wall boundary conditions ($\mathbf{q} \cdot \boldsymbol{\eta} = 0$) are considered. The Cuda programs are executed on a GeForce GTX 480 and a GeForce GTX 580. Table 1 shows all the execution times in seconds. Figure 6 shows a top view of the fluid evolution for the 400×400 mesh size for the three numerical schemes at different time instants. As it can be seen, for both two cards, the WAF method is slightly slower that the first order HLL method and twice faster than the second order HLL method, but it provides numerical results almost as accurate as the second order HLL scheme.

We have also implemented a serial CPU and a quadcore OpenMP version for the three numerical schemes, both written in C++ and using the Eigen library [17], reaching speedups of more than 200 for the three numerical schemes in both graphics cards with respect to the monocore CPU version.



(a) $t = 1s$ (b) $t = 2s$

Figure 6: Top view of the evolution of the circular dam break problem at different time instants with the 400×400 mesh. From top to bottom: 1^{st} order HLL, 2^{nd} order HLL, and WAF.

Table 2: Accuracy test: HLL scheme. L^1 errors and orders.

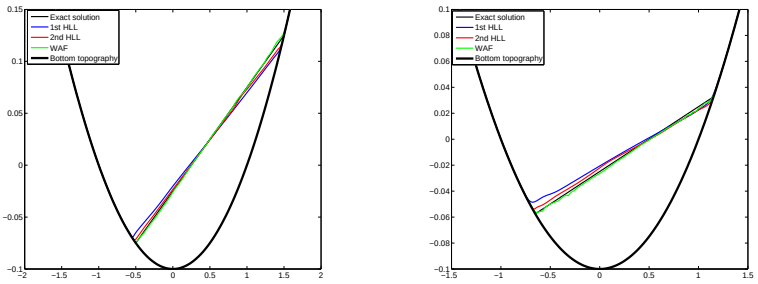
N. cells	error h	order h	error q_x	order q_x	error q_y	order q_y
25×25	$3.28 \cdot 10^{-01}$	–	$7.96 \cdot 10^{-01}$	–	1.79	–
50×50	$1.75 \cdot 10^{-01}$	0.90	$4.65 \cdot 10^{-01}$	0.77	1.04	0.78
100×100	$8.97 \cdot 10^{-02}$	0.97	$2.48 \cdot 10^{-01}$	0.91	$5.57 \cdot 10^{-01}$	0.91
200×200	$4.32 \cdot 10^{-02}$	1.05	$1.22 \cdot 10^{-01}$	1.02	$2.73 \cdot 10^{-01}$	1.02
400×400	$2.11 \cdot 10^{-02}$	1.05	$5.88 \cdot 10^{-02}$	1.05	$1.30 \cdot 10^{-01}$	1.05

Table 3: Accuracy test: 2^{nd} order HLL scheme. L^1 errors and orders.

N. cells	error h	order h	error q_x	order q_x	error q_y	order q_y
25×25	$8.67 \cdot 10^{-02}$	–	$2.87 \cdot 10^{-01}$	–	$4.90 \cdot 10^{-02}$	–
50×50	$3.22 \cdot 10^{-02}$	1.42	$1.09 \cdot 10^{-01}$	1.38	$2.01 \cdot 10^{-02}$	1.28
100×100	$8.81 \cdot 10^{-03}$	1.87	$3.12 \cdot 10^{-02}$	1.81	$6.18 \cdot 10^{-03}$	1.70
200×200	$2.32 \cdot 10^{-03}$	1.92	$8.12 \cdot 10^{-03}$	1.94	$1.67 \cdot 10^{-04}$	1.88
400×400	$6.02 \cdot 10^{-04}$	1.95	$2.11 \cdot 10^{-03}$	1.94	$4.38 \cdot 10^{-05}$	1.93

Table 4: Accuracy test: WAF scheme. L^1 errors and orders.

N. cells	error h	order h	error q_x	order q_x	error q_y	order q_y
25×25	$1.12 \cdot 10^{-01}$	–	$3.81 \cdot 10^{-01}$	–	$6.94 \cdot 10^{-01}$	–
50×50	$4.37 \cdot 10^{-02}$	1.36	$1.79 \cdot 10^{-01}$	1.08	$2.74 \cdot 10^{-01}$	1.33
100×100	$1.78 \cdot 10^{-02}$	1.29	$8.22 \cdot 10^{-02}$	1.12	$1.14 \cdot 10^{-01}$	1.25
200×200	$7.19 \cdot 10^{-03}$	1.31	$3.54 \cdot 10^{-02}$	1.21	$5.04 \cdot 10^{-02}$	1.18
400×400	$2.89 \cdot 10^{-03}$	1.31	$1.52 \cdot 10^{-02}$	1.21	$2.17 \cdot 10^{-02}$	1.21



(a) $t = 3T$ (b) $t = 3T + T/6$

Figure 7: 2-d oscillating lake: surface elevation vs y-coordinate, for $y = 0$ at different times steps: Exact solution in black, 1^{st} order HLL in blue, 2^{nd} order HLL in red and WAF in green

6.2. Accuracy test.

Next, we consider a test proposed in [33] in order to measure the accuracy of the three schemes for a non-stationary smooth solution. Specifically, the bottom topography is defined as $H(\boldsymbol{x}) = 2 - \sin(2\pi x) - \cos(2\pi y)$, and the initial water height is $h(\boldsymbol{x}, 0) = 10 + e^{\sin(2\pi x)}\cos(2\pi y)$, while the initial discharges are given by

$$q_x(\boldsymbol{x}, 0) = \sin(\cos(2\pi x))\sin(2\pi y), \quad q_y(\boldsymbol{x}, 0) = \cos(2\pi x)\cos(\sin(2\pi y)).$$

The computational domain is the unit square and periodic boundary conditions have been imposed.

Tables 2-4 show the results obtained at time $t = 0.05$ for the three schemes considered here, as shocks developed later for this problem. A reference solution has been computed using the second order HLL scheme on a mesh with 1600×1600 grid points. The CFL number has been set to 0.5. As it can be seen, HLL achieves first order accuracy (see Table 2), the

second order HLL scheme achieves second order and the WAF scheme is not second order of accuracy, but its convergence rate is grater that one for this tests (see Table 4).

6.3. A two-dimensional oscillating lake

This numerical test is design to show its performance in solutions where wet/dry fronts appear. In this case we follow [8] in order to modify the numerical scheme in such situations. Let us remark that in the case of the second order HLL scheme is critical to ensure the positivity of the reconstruction of the water height at the intercells. Here we follow the technique proposed in [22].

Let us consider the paraboloidal topography defined by the depth function

$$H(\boldsymbol{x}) = h_0\left(1 - \frac{x^2+y^2}{a^2}\right), \quad \boldsymbol{x} \in [-2,2]\times[-2,2],$$

together with the periodic analytical solution of the two-dimensional shallow water equations stated in [32]:

$$h(\boldsymbol{x},t) = \max\left(0, \frac{\sigma h_0}{a^2}\big(2x\cos(\omega t) + y\sin(\omega t) - \sigma\big) + H(\boldsymbol{x})\right),$$
$$u_x(\boldsymbol{x},t) = -\sigma\omega\sin(\omega t), \quad u_y(\boldsymbol{x},t) = \sigma\omega\cos(\omega t),$$

where u_x and u_y are the velocities in the x and y directions, and $\omega = \sqrt{2gh_0}/a$. The values $a = 1$, $\sigma = 0.5$ and $h_0 = 0.1$ have been considered for this test.

The computations have been performed using a quadrilateral mesh with $\Delta x = \Delta y = 0.02$ and CFL number 0.7. Comparisons between the numerical and the analytical free surfaces at different times are shown in Figure 7, where T represents the oscillation period. Although a small distortion near

the shorelines can be observed in some cases, they can be reduced using a finer spatial discretization. On the other hand, the planar form of the free surface is maintained throughout the computation. Note that the quality of the solution of the WAF method is as good as the one provided by the second order HLL scheme in this test case, being the first order HLL method the more diffusive one as expected.

6.4. Dam break problem over real topography

Finally, let us consider a dam break problem over a real topography. More precisely, the considered zone corresponds to the neighbourhood of El Limonero Dam. This dam is located on the Guadalmedina River at around 5.5 km upstream from the estuary and at least 1 km from the city of Málaga. The dam is built with non cohesive heterogeneus loose material with a impermeable centre. The essential objective of this dam is the protection of the city of Málaga against freshets from the Guadalmedina river. The total reservoir capacity is about 30.00 hm^3.

The considered domain is a square of 3260 m width and 8000 m long, discretized using 1.043 millions cell of 5m $\times$ 5m. Only the closest portion of the dam is considered in the domain. Wall boundary conditions are imposed and as initial condition we consider that the water is at rest and confined inside the dam, that it is fill up to 90% of its total capacity. The CFL parameter is set to 0.8 and Manning coefficient n is set to 0.03. At time $t = 0$, the dam is partially broken and a flood starts. Figure 8 shows the evolution of the flood at different time steps computed with the WAF method.

7. Conclusions

In this paper first we present a reformulation of a first and second order HLL method and a two-waves TVD-WAF method under a similar structure that allows us to design the same structure for their CUDA implementations. The application to the two-dimensional SWE is done using its property of invariance by rotation. Then, at each edge of the mesh, a 1D projected SWE equation with a transport equation is considered. This technique is is specially suitable for GPU implementation. This two dimensional WAF method is not second order of accuracy, but we show in the numerical tests that it is twice faster than the second order HLL method, providing almost the same numerical results. This reformulation of the WAF method and an improved definition of the flux limiters allows us to obtain a fast and accurate solver, well suitable for GPU implementation and more robust in situations like wet/dry fronts. This technique is is specially suitable for GPU implementation. This two dimensional WAF method is not second order of accuracy, but we show in the numerical tests that it is twice faster than the second order HLL method, providing almost the same numerical results. This reformulation of the WAF method and an improved definition of the flux limiters allows us to obtain a fast and accurate solver, well suitable for GPU implementation and more robust in situations like wet/dry fronts.

References

[1] M. de la Asunción, J. M. Mantas, M. J. Castro, Simulation of one-layer shallow water systems on multicore and CUDA architectures, J. Supercomput. (DOI: 10.1007/s11227-010-0406-2) 2010.

[2] M. de la Asunción, J. M. Mantas, M. J. Castro, Programming CUDA-based GPUs to simulate two-layer shallow water flows, in: Euro-Par 2010 - Parallel Processing, Vol. 6272 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2010, pp. 353–364.

[3] M. de la Asunción, J. M. Mantas, M. J. Castro, E.D. Fernández, An MPI-CUDA implementation of an improved Roe method for two-layer shallow water systems, J. Parallel and Distributed Computing (In Press, DOI:10.1016/j.jpdc.2011.07.012).

[4] E. Audusse, F. Bouchut, M.-O. Bristeau, R. Klein, B. Perthame (2004). A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. SIAM J. Sci. Comp., 25(6):2050–2065.

[5] S.J. Billet, E.F. Toro (1997). On WAF-Type Schemes for Multidimensional Hyperbolic Conservation Laws. J. Comput. Phys. 130, 1–24.

[6] Bradford, S. F., and Sanders, B. F. (2002). Finite-volume model for shallow-water ooding of arbitrary topography. Journal of Hydraulic Engineering 128(3), 289-298.

[7] A. Brodtkorb, T. Hagen, K.-A. Lie, J. Natvig, Simulation and visualization of the saint-venant system using GPUs, Computing and Visualization in Science (2011) 1–13.

[8] M.J. Castro, A. Ferreiro, J.A. García, J. González-Vida, J. Macías, C. Parés, M.E. Vázquez-Cendón (2005). On the numerical treatment of wet/dry fronts in shallow flows: application to one-layer and two-layers systems. Math. and Comp. Model., 42(3-4): 419–439.

[9] M.J. Castro, J.A. García, J.M. González and C. Parés (2006). A parallel 2d finite volume scheme for solving systems of balance laws with non-conservative products: application to shallow flows. Comp. Meth. Appl. Mech. Eng. 196, 2788-2815.

[10] M.J. Castro, J.A. García, J.M. González and C. Parés (2008). Solving shallow-water systems in 2D domains using finite volume methods and multimedia SSE instructions. J. Comput. App. Math., 221: 16-32.

[11] M.J. Castro, E.D. Fernández, A.M. Ferreiro, A. García, C. Parés (2009). High order extension of Roe schemes for two dimensional nonconservative hyperbolic systems. J. Sci. Comput. 39, 67–114.

[12] M.J. Castro, T. Chacón Rebollo, E.D. Fernández-Nieto, J.M. González Vida and C. Parés (2008). Well-balanced finite volume schemes for 2D non-homogeneous hyperbolic systems. Application to the dam break of Aznalcóllar. Comput. Methods Appl. Mech. Engrg. 197, 3932-3950

[13] M. J. Castro, S. Ortega, M. de la Asunción, J. M. Mantas, J. M. Gallardo, GPU computing for shallow water flow simulation based on finite volume schemes, Comptes Rendus Mécanique 339 (2-3) (2011) 165–184, High Performance Computing.

[14] M.J. Castro, E.D. Fernández-Nieto *A class of computationally fast first order finite volume solvers: PVM methods*. Submitted to SINUM, 2010.

[15] NVIDIA, CUDA home page, http://www.nvidia.com/object/cuda_home_new.html.

[16] S. F. Davis. *Simplified Second-Order Godunov-Type Methods*. SIAM J. Sci. Stat. Comput., 9:445473, (1988)

[17] Eigen 3.0.1., http://eigen.tuxfamily.org.

[18] E.D. Fernández-Nieto, G. Narbona-Reina (2008). Extension of waf type methods to nonhomogeneous shallow water equations with pollutant, J. Sci. Comput. **36**(2), 193–217.

[19] E.D. Fernández-Nieto, D. Bresch, J. Monnier (2008). A consistent intermediate wave speed for a well-balanced HLLC solver. C. R. Math. Acad. Sci. Paris 346, no. 13-14, 795-800.

[20] J. Gallardo, S. Ortega, M. de la Asunción, J. M. Mantas, Two-dimensional compact third-order polynomial reconstructions. Solving nonconservative hyperbolic systems using GPUs, Journal of Scientific Computing (2011) 1–23.

[21] M. Geveler, D. Ribbrock, S. Mallach, D. G A Simulation Suite for Lattice-Boltzmann based Real-Time CFD Applications Exploiting Multi-Level Parallelism on modern Multi- and Many-Core Architectures, Journal of Computational Science **2**(2), 113–123, 2011.

[22] A. Kurganov, G. Petrova (2007). A second-order well-balanced positivity preserving central-upwind scheme for the Saint-Venant system. Commun. Math. Sci. Vol. 5, n. 1 133–160.

[23] Y. Loukili, A. Soulaymani, *Numerical Tracking of Shallow Water Waves by the Unstructured Finite Volume WAF Approximation*, Int. J. Comput. Meth. Eng. Sci. Mech. **8**(2), pp- 75-88 (2007)

[24] C. Parés, M.J. Castro (2004). On the well-balance property of Roe's method for nonconservative hyperbolic systems. Applications to shallow-water systems. ESAIM: M2AN, 38(5):821–852.

[25] C. Parés (2006). Numerical methods for nonconservative hyperbolic systems: a theoretical framework. SIAM J. Num. Anal. 44, 300– 321.

[26] M. L. Saetra, A. R. Brodtkorb, Shallow water simulations on multiple GPUs, Proceedings of the Para 2010 Conference, Lecture Notes in Computer Science (2011) Accepted for publication.

[27] C.-W. Shu. (1988) Total variation diminishing time discretizations. SIAM J. Sci. Statist. Comput. 9(6): 1073–1084.

[28] E.F. Toro (1989). A weighted average flux method for hyperbolic conservation laws. Proc. R. Soc. Lond. A 423, 401–418.

[29] E.F. Toro (1992). Riemann problems and the the WAF method for solving two-dimensional shallow water equations. Phil. Trans. Roy. Soc. London, A338, 43–68.

[30] E.F. Toro (1992). The weighted average flux method applied to the time dependent euler equations. Phil. Trans. Roy. Soc. London, A341, 499–530.

[31] E.F. Toro (2001). *Shock-Capturing Methods for Free-Surface Shallow Flows*, Wiley, England.

[32] W.C. Thacker (1981). Some exact solutions to the nonlinear shallow-water wave equations. J. Fluid Mech. 107: 499–508.

[33] Y. Xing, C.-W. Shu (2005). High order finite difference WENO schemes with the exact conservation property for the shallow water equations, J. Comput. Phys. 208: 206–227.

(a) t=1m 50s

(b) t=4m 45s

(c) t=5m 41 s

(d) t=19m 59s

Figure 8: Views of the evolution of the flood after the dam break computed with the WAF method.