Computers & Fluids 80 (2013) 441–452

Contents lists available at SciVerse ScienceDirect

Computers & Fluids

journal homepage: www.elsevier.com/locate/compfluid

Efficient GPU implementation of a two waves TVD-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, Spain
[b] Dpto. Matemática Aplicada I, Universidad de Sevilla, Spain
[c] Dpto. Análisis Matemático, Universidad de Málaga, Spain
[d] Dpto. Matemática Aplicada, Universidad de Málaga, Spain

ARTICLE INFO

Article history:
Received 21 September 2011
Received in revised form 4 January 2012
Accepted 22 January 2012
Available online 3 February 2012

Keywords:
Finite volume schemes
TVD-WAF scheme
CUDA
Shallow-water equations

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 TVD-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 the TVD-WAF method is slightly slower that the first order HLL method and two times faster than the second order HLL method, but it provides numerical results almost as accurate as the second order HLL scheme.

1. Introduction

In this paper we present an efficient implementation of HLL (Harten–Lax–van Leer [19]) and Total Variation Diminishing-Weighted Average Flux (TVD-WAF [26]) methods applied to the two-dimensional shallow water equations (SWEs in what follows) with topography.

The numerical solutions of SWE are useful for several applications related to geophysical flows, such as the simulation of rivers, dam-breaks, and floods. 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 [11,8,18,3] 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 [1–3,8,11,18,24]. A CUDA solver for one-layer system based on the first order Roe scheme [22] 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 [8,24,17] and of implementations of first-order schemes for one and two-layer systems on triangular meshes [11,3].

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.

* Corresponding author. Tel.: +34 951952508; fax: +34 95213 2746.
E-mail address: jgv@uma.es (J.M. González-Vida).

0045-7930/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compfluid.2012.01.012

- 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 TVD-WAF schemes for solving the two-dimensional SWE have been considered. More precisely, we extend here to two-dimensional problems the well-balanced HLL method proposed in [16] 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 performed by using the property of invariance by rotation of the SWE. Thus, at each edge of the mesh, a 1D projected SWE is considered, where the unknowns are the height of the water column and the normal and the tangential discharges. A second order scheme is also proposed using a MUSCL type reconstruction operator described in [20], but following the procedure described in [10].

Finally, the 2D extension of the two-waves TVD-WAF method proposed in [15] has also been carried out. TVD-WAF method is a second order TVD method proposed by Toro in [26]. 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 TVD-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 [27] and [28] Toro presents the application of TVD-WAF method for the homogeneous shallow water and Euler equations. An extension to multidimensional systems was performed by Billet and Toro in [5]. In [21] a TVD-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 [7]. This technique allows to preserve water at rest but present a loss of accuracy for large wave run-up.

The extension of the TVD-WAF method that we proposed here is also defined using again the property of invariance by rotation of the SWE, and the TVD-WAF flux introduced in [15] to approximate the 1D problems. In fact, a new reformulation of the numerical flux that it is equivalent to the one given in [15] has been considered here that allows us to obtain a more robust solver in wet/dry fronts. Nevertheless, the TVD-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. Section 3 presents the 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:

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

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 mean velocities $\boldsymbol{u}(\boldsymbol{x},t)=(u_x(\boldsymbol{x},t),\ u_y(\boldsymbol{x},t))$, by the equality: $\boldsymbol{q}(\boldsymbol{x},t)=\boldsymbol{u}(\boldsymbol{x},t)h(\boldsymbol{x},t)$, $i=1,2$; g is the gravity and $H(\boldsymbol{x})$, the depth function measured from a fixed level of reference.

$$F_1(U)=\begin{pmatrix} q_x & \frac{q_x^2}{h}+\frac{1}{2}gh^2 & \frac{q_xq_y}{h}\end{pmatrix}^T,$$
$$F_2(U)=\begin{pmatrix} q_y & \frac{q_xq_y}{h} & \frac{q_y^2}{h}+\frac{1}{2}gh^2\end{pmatrix}^T.$$

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

$$S_1(U)=\begin{pmatrix}0 & gh & 0\end{pmatrix}^T,\ S_2(U)=\begin{pmatrix}0 & 0 & gh\end{pmatrix}^T.$$

Finally, $S^F(U)$, parameterize the friction term. Here, Manning friction law is used:

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

being n the Manning coefficient.

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

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

3. Numerical schemes

In this section, we present a brief description of the 2D first and second order HLL and the two-waves TVD-WAF finite volume schemes when they are applied to the SWE (1). In this section, the term $S^F(U)$ is supposed to be zero, as $S^F(U)$ will be discretized semi-implicitly.

First, 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 neighbor of V_i; E_{ij} the common edge of two neighboring 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. H_i is the average of the bathymetry in the cell V_i:

$$H_i=\frac{1}{|V_i|}\int_{V_i}H(\boldsymbol{x})d\boldsymbol{x},$$

and 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:

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

In order to extend to 2D problems, the 1D HLL and two-waves TVD-WAF Riemann solvers, a family of projected Riemman problems in the normal direction to each edge of the mesh is

considered. These projected Riemann problems can be easily defined as system (1) verifies the property of invariance by rotations. Thus, if we defined

$$T_{\eta_{ij}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \eta_{ij,x} & \eta_{ij,y} \\ 0 & -\eta_{ij,y} & \eta_{ij,x} \end{pmatrix},$$

where η_{ij} is the normal unit vector at edge E_{ij} pointing towards the cell V_j and if we denote $F_{\eta_{ij}}(U) = F_1(U)\eta_{ij,x} + F_2(U)\eta_{ij,y}$, $\boldsymbol{S}(U) = (S_1(U), S_2(U))$ then

$$F_{\eta_{ij}}(U) = T^{-1}_{\eta_{ij}} F_1(T_{\eta_{ij}}U), \ \boldsymbol{S}(U)\cdot\eta_{ij} = T^{-1}_{\eta_{ij}} S_1(T_{\eta_{ij}}U). \tag{2}$$

Moreover, it is easy to check that $T_{\eta_{ij}}U$ verifies the system

$$\partial_t(T_{\eta_{ij}}U) + \partial_{\eta_{ij}} F_1(T_{\eta_{ij}}U) = S_1(T_{\eta_{ij}}U)\partial_{\eta_{ij}}H + Q_{\eta^\perp_{ij}}, \tag{3}$$

where $Q_{\eta^\perp_{ij}} = T_{\eta_{ij}}\left(-\partial_{\eta^\perp_{ij}} F_{\eta^\perp_{ij}}(U) + \boldsymbol{S}(U)\cdot\eta^\perp_{ij}\partial_{\eta^\perp_{ij}}H\right)$. Now, the 1D projected Riemman problems are defined by considering system (3), neglecting the tangential term $Q_{\eta^\perp_{ij}}$.

Thus, a general form of a first order finite volume scheme for SWE (1) is given by

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

where $\mathcal{F}^-_{ij}\left(U_i^n, U_j^n, H_i, H_j\right)$ is defined from a 1D Riemman solver. In the next two subsections we detail the definition of $\mathcal{F}^-_{ij}\left(U_i^n, U_j^n, H_i, H_j\right)$ when the HLL (respectively two-waves TVD-WAF) solver is used.

3.1. HLL scheme

Following [29], the numerical fluxes $\mathcal{F}^{HLL^\pm}_{ij}\left(U_i^n, U_j^n, H_i, H_j\right)$ constructed from the 1D HLL scheme introduced in [16] can be written as follows:

1. Let us define

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

   where $W_{[i_1,\dots,i_s]}$ is the vector defined from vector W, using its i_1th,…,i_sth components.
2. Let $\Phi^\pm_{\eta_{ij}}$ be the 1D HLL flux associated to the 1D one layer shallow-water system defined using the 1st, 2nd equations of system (3) where the term $Q_{\eta^\perp_{ij}}$ has been neglected:

   $$\Phi^\pm_{\eta_{ij}} = \mathcal{D}^\pm_{ij}(U_{\eta_{ij},i}, U_{\eta_{ij}j}, H_i, H_j) \mp \mathcal{F}_C(U_{\eta_{ij},i}),$$

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

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

   where

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

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

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

   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 [13]:

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

   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

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

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

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

   Let us remark that $\Phi^\pm_{\eta^\perp_{ij}}$ is the numerical flux associated to the 3-rd equation of system (3) where, again, the term $Q_{\eta^\perp_{ij}}$ has been neglected.
4. Finally, $\mathcal{F}^{HLL^\pm}_{ij}$ is defined by $\mathcal{F}^{HLL^\pm}_{ij} = T^{-1}_{\eta_{ij}}F^\pm_{ij}$, where

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

Remark 1. Note that $q_{\eta^\perp_{ij}}$ can be seen as a passive scalar that is advected by the flow. Thus, F^-_{ij} can be interpreted as s HLLC (Harten–Lax–van Leer-Contact, see [29]) numerical flux associated to the 1D system (3).

Remark 2. The numerical flux introduced in [16] and the one presented here are equivalent, but they are not written in the same form. Here, we have rewritten the HLL flux as a flux-difference scheme instead of using its standard writing. The main reason is that flux-difference schemes are easier to correct for dealing with wet/dry fronts and, from the computational point of view, they usually require less computational effort.

3.2. Two-waves TVD-WAF scheme

Let us now define the numerical flux $\mathcal{F}^{WAF^\pm}_{ij}\left(U_i^n, U_j^n, H_i, H_j\right)$ corresponding to the natural extension of the two-waves TVD-WAF method introduced in [15] to 2D domains, using the method of lines. TVD-WAF schemes were first introduced by Toro in [26] 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, this extension provides as good results as a multidimensional second order scheme with less computational effort as we will show in the numerical test Section. In [15], authors propose a redefinition of the two-waves TVD-WAF method that mimics the structure of the HLL scheme for the 1D SWE. Thus, using the same ideas, $\mathcal{F}^{WAF^\pm}_{ij}\left(U_i^n, U_j^n, H_i, H_j\right)$ can be defined by the same four steps, previously enumerated, where the coefficients $\alpha_{ij,k}$, $k = 0,1$ in (5) are now given by

$$\begin{aligned} \alpha_{ij,0} &= (\mathcal{L}_{ij}(S_{L,ij}, \chi_{L,ij}) - \mathcal{L}_{ij}(S_{R,ij}, \chi_{R,ij}))\frac{S_{L,ij}S_{R,ij}}{S_{R,ij} - S_{L,ij}} \\ \alpha_{ij,1} &= \frac{\mathcal{L}_{ij}(S_{R,ij}, \chi_{R,ij})S_{R,ij} - \mathcal{L}_{ij}(S_{L,ij}, \chi_{L,ij})S_{L,ij}}{S_{R,ij} - S_{L,ij}}, \end{aligned} \tag{10}$$

with

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

being χ a flux limiter function. $\chi_{L,ij}$ (equally $\chi_{R,ij}$) is defined using the Van Albada's flux limiter:

$$\chi_{L,ij} = \begin{cases} 1 & \text{if } p_{\max,L} < \beta \\ \phi(r_{L,ij}) & \text{otherwise,} \end{cases} \tag{11}$$

where β is set to d_{ij}^3 and

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

Here, we propose a definition of $r_{L,ij}$ that is different to the usual one, and from our experience, it provides better results:

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

with

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

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 $e_{ij,R} = h_{j,R} - H_{j,R} - (h_j - H_j)$, being $h_{i,L}$ (respectively $h_{j,R}$) and $H_{i,L}$ (respectively $H_{j,R}$) the water height and bottom topography corresponding to cell $V_{i,L}$ (respectively $V_{j,R}$) shown in Fig. 1.

Finally, the value $u^{ij}_{\eta^\perp_{ij}}$ in (8) is replaced by

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

where

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

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

Remark 3. Note that if $\chi_{L,ij} = \chi_{R,ij} = \chi^\perp = 0$, then, the two-waves TVD-WAF scheme previously described exactly coincides with the HLL scheme given in Section 3.1.

Remark 4. As in the previous section, the two-waves TVD-WAF numerical flux introduced in [15] and the one presented here are equivalent, but they are not written in the same form. Here, we have rewritten the TVD-WAF flux as a flux-difference scheme instead of using its standard writing.

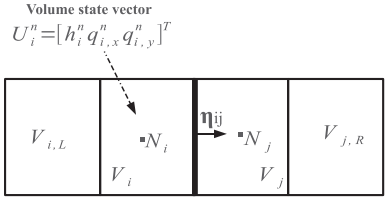


Fig. 1. Stencil of the TVD-WAF method for a vertical edge.

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 [10].

First, we consider the second order reconstruction operator defined at cell V_i by

$$U_i(\boldsymbol{x}, t) = U_i(t) + (U_x(t))_i(x - x_i) + (U_y(t))_i(y - y_i), \tag{12}$$

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 [20] where

$$(U_x(t))_i = \text{minmod}\left(\theta\frac{U_i - U_{i,W}}{\Delta x}, \frac{U_{i,E} - U_{i,W}}{2\Delta x}, \theta\frac{U_{i,W} - U_i}{\Delta x}\right), \quad \theta \in [1,2], \tag{13}$$

$$(U_y(t))_i = \text{minmod}\left(\theta\frac{U_i - U_{i,S}}{\Delta y}, \frac{U_{i,N} - U_{i,S}}{2\Delta y}, \theta\frac{U_{i,N} - U_i}{\Delta y}\right), \quad \theta \in [1,2], \tag{14}$$

and

$$\text{minmod}(z_1, z_2, \ldots) = \begin{cases} \min\limits_j z_j & \text{if } z_j > 0 \ \forall j, \\ \max\limits_j z_j & \text{if } z_j < 0 \ \forall j, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

$U_{i,l}$, $l = E, W, S, N$ are those given in Fig. 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.

Remark 5. In order to obtain an 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 [10] 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 [20] 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:

$$U_i'(t) = -\frac{1}{|V_i|}\sum_{j\in\mathcal{N}_i}|E_{ij}|\mathcal{F}^{HLL^-}_{ij}\left(U^-_{ij}(t), U^+_{ij}(t), H^-_{ij}, H^+_{ij}\right) - \frac{1}{|V_i|}\int_{V_i}\begin{pmatrix} 0 \\ gh_i(\boldsymbol{x}, t)(z_{sx}(t))_i \\ gh_i(\boldsymbol{x}, t)(z_{sy}(t))_i \end{pmatrix} \tag{16}$$

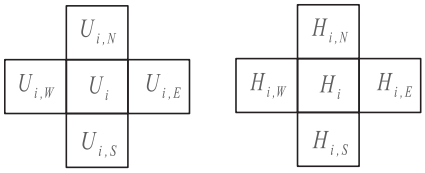


Fig. 2. Stencil of the second order reconstruction.

where $U_{ij}^{-}(t)$ and H_{ij}^{-} (respectively $U_{ij}^{+}(t)$ and H_{ij}^{+}) are the values of the reconstruction defined by (12) 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 (16) is approximated by the trapezoidal rule and a second order Runge–Kutta TVD scheme is used to approximate the time derivative (see [25]).

Remark 6. As mentioned before, the major difference between this scheme and the one presented in [20] is that they consider a continuous reconstruction of the bottom topography, while we do not use this fact.

Remark 7. Let us remark that the usual Courant–Friedrichs–Lewy (CFL) condition must be imposed to ensure stability of the three schemes.

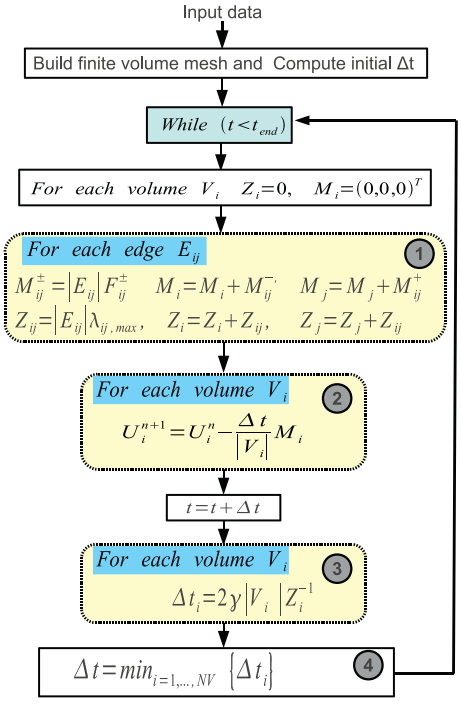
Remark 8. To ensure the positivity of the numerical schemes, first we use the desingularization formula (see [20])

$$u_\alpha = \frac{\sqrt{2}hq_\alpha}{\sqrt{h^4 + \max(h^4, \epsilon)}}, \quad \alpha = x, y \tag{17}$$

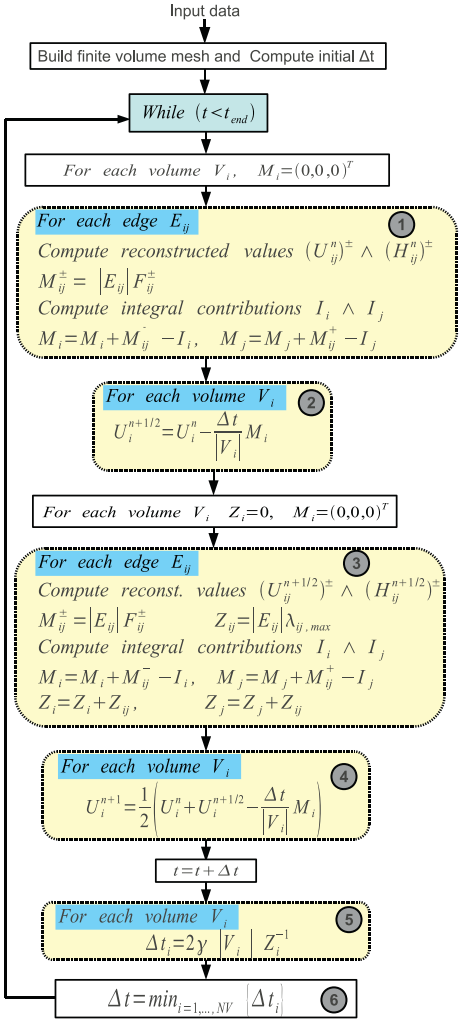
where $\epsilon = \max\{(\Delta x)^4, (\Delta y)^4\}$, and q_x (respectively q_y) is redefined by $q_x := u_x h$ (respectively $q_y := u_y h$). Next, if the second order HLL scheme is used, we follow Kurganov and Petrova [20] to modify the reconstruction of the water height to guarantee the positivity of its reconstructed value at the edges. Finally, we follow a similar procedure to the one described in [6] to compute a local Δt that limits the outflow across the edges closed to a wet/dry front that violates the positivity of the water height. As pointed in [6], this procedure has no impact on the global time step, that is computed by the usual CFL condition.

Let us finally remark that in wet/dry fronts with emerging bottom topographies, the numerical fluxes are modified following [9] to avoid spurious pressure forces.

Remark 9. The three schemes considered here are path-conservative in the sense introduced by Pares in [23], 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.



(a) First order HLL and TVD-WAF methods



(b) Second order HLL

Fig. 3. Parallel algorithms.

Remark 10. Note that $S^F(U)$ can be considered as a function of h, $\|q\|$, q_x and q_y, that is, $S^F(U) = S^F(h, \|q\|, q_x, q_y)$. Thus, if the first order HLL scheme is used (the same procedure can be applied if the two-waves TVD-WAF method is used), the discretization of SWE (1), including the friction term is as follows:

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{|V_i|} \sum_{j \in \mathcal{N}_i} |E_{ij}| \mathcal{F}_{ij}^{HLL^-} \left(U_i^n, U_j^n, H_i, H_j \right) + \Delta t S^F \left(h_i^n, \|\boldsymbol{q}_i^n\|, q_{x,i}^{n+1}, q_{y,i}^{n+1} \right). \quad (18)$$

Note that U_i^{n+1} is updated by solving two simple linear equations: one for $q_{x,i}^{n+1}$ and another for $q_{y,i}^{n+1}$. This procedure can be easily extend to second order schemes by considering the corresponding second order Runge–Kutta TVD scheme.

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 TVD-WAF schemes can be represented by a single diagram, shown in Fig. 3a. 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.

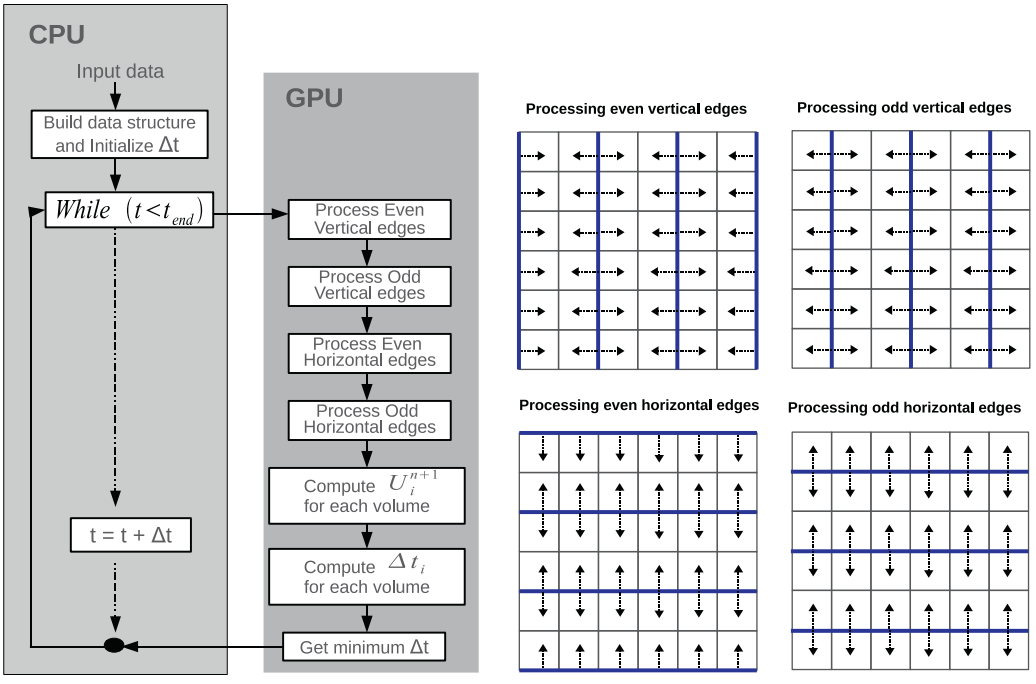
When the finite volume mesh has been constructed from the input data and an initial time step Δt is estimated, the time stepping process is repeated until the final simulation time is reached:

1. *Edge-based calculations*: For each edge E_{ij}, common to cells V_i and V_j, two computations are performed:
   (a) Vectors $M_{ij}^{\pm} = |E_{ij}| F_{ij}^{\pm} \in \mathbb{R}^3$ are 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, $F_{ij}^{\pm}$ corresponds to $\mathcal{F}_{ij}^{HLL^{\pm}}$ (see Section 3.1), and in the TVD-WAF method, $F_{ij}^{\pm}$ corresponds to $\mathcal{F}_{ij}^{WAF^{\pm}}$ (see Section 3.2).
   (b) The value $Z_{ij} = |E_{ij}| \lambda_{ij,max}$, where $\lambda_{ij,max} = \max(|S_{L,ij}|, |S_{R,ij}|)$, 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.
2. *Computation of* U_i^{n+1}: The $(n+1)$th state of each volume $\left(U_i^{n+1}\right)$ is calculated from the nth state and the data computed in previous phases. The formula corresponds to Eq. (4). If the friction term $S^F(U)$ is considered, then the discretization of this term is performed here using the formula (18). This phase can also be performed in parallel.
3. *Computation of the local* Δt_i: For each volume V_i, the local Δt_i is obtained by applying: $\Delta t_i = 2\gamma |V_i| Z_i^{-1}$. In the same way, the computation for each volume can be performed in parallel.
4. *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.

4.2. Parallelism sources of the second order HLL scheme

Fig. 3b shows graphically the main steps of the second order HLL scheme. 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 and an initial time step Δt is computed, the time stepping is repeated, by applying a second order Runge–Kutta TVD method



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

Fig. 4. Steps in the CUDA implementation for the HLL and TVD-WAF schemes.

which consists in two stages. These two stages mainly involve the following computing phases:

1. *Edge-based calculations*: For each edge E_{ij}, common to cells V_i and V_j, three computations are performed:
   (a) The reconstructed values, $\left(U_{ij}^n\right)^-, \left(U_{ij}^n\right)^+$, as well as the reconstructed topography values, H_{ij}^-, H_{ij}^+ are computed using the stencil values shown in Fig. 2 according to Eqs. (12)–(14).
   (b) Vector $M_{ij}^{\pm} = |E_{ij}|\mathcal{F}_{ij}^{HLL\pm}\left(\left(U_{ij}^n\right)^-, \left(U_{ij}^n\right)^+, H_{ij}^-, H_{ij}^+\right)$ is computed independently for each edge (see (16)) and added to M_i and M_j (see Fig. 3b).
   (c) The contributions of the integral term of Eq. (16) for volumes V_i (I_i) and V_j (I_j) are computed using the trapezoidal rule and added to M_i and M_j, respectively.
2. *Computation of* $U_i^{n+1/2}$: For each volume, the vector $U_i^{n+1/2}$ is computed independently from U_i^n and M_i. As in the previous section, if the friction term $S^F(U)$ is considered, then the first step of its discretization is performed here.
3. *Edge-based calculations for the second stage*: For each edge E_{ij} common to cells V_i and V_j, four computations are performed:
   (a) The reconstructed values, $\left(U_{ij}^{n+1/2}\right)^-, \left(U_{ij}^{n+1/2}\right)^+$, as well as the reconstructed topography values, H_{ij}^-, H_{ij}^+ are computed using the suitable stencil values (obtained from $U^{n+1/2}$ and H) as in the previous edge-based step.
   (b) Vector $M_{ij}^{\pm} = |E_{ij}|\mathcal{F}_{ij}^{HLL\pm}\left(\left(U_{ij}^{n+1/2}\right)^-, \left(U_{ij}^{n+1/2}\right)^+, H_{ij}^-, H_{ij}^+\right)$ is computed independently for each edge (see (16)) and added to M_i and M_j (see Fig. 3b).
   (c) The contributions of the integral term of Eq. (16) for volumes V_i (I_i) and V_j (I_j) are also computed and added to M_i and M_j, respectively.
   (d) The value $Z_{ij} = |E_{ij}|\lambda_{ij,max}$ is also computed independently for each edge as in the previous subsection.
4. *Computation of* U_i^{n+1}: The $(n+1)$th state of each volume $\left(U_i^{n+1}\right)$ is computed from U_i^n, $U_i^{n+1/2}$ and M_i. If the friction term $S^F(U)$ is considered, then the second step of its discretization is performed here.
5. *Computation of the local* Δt_i: For each volume V_i, the local Δt_i is computed.
6. *Computation of* Δt: The minimum of all the local Δt_i values previously computed for each volume is computed.

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 (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 Fig. 4a. Each processing step executed on the GPU is assigned to a CUDA kernel. Next, we describe in detail each step:

- *Build data structure and initialize* Δt: In this step, the data structure that will be used on the GPU is built and an initial time step Δt is computed. 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.

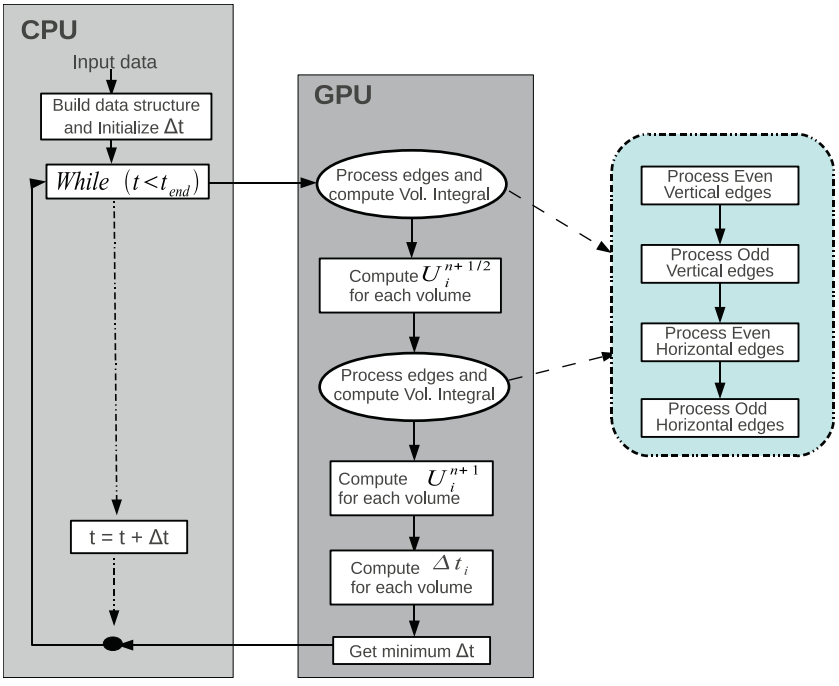
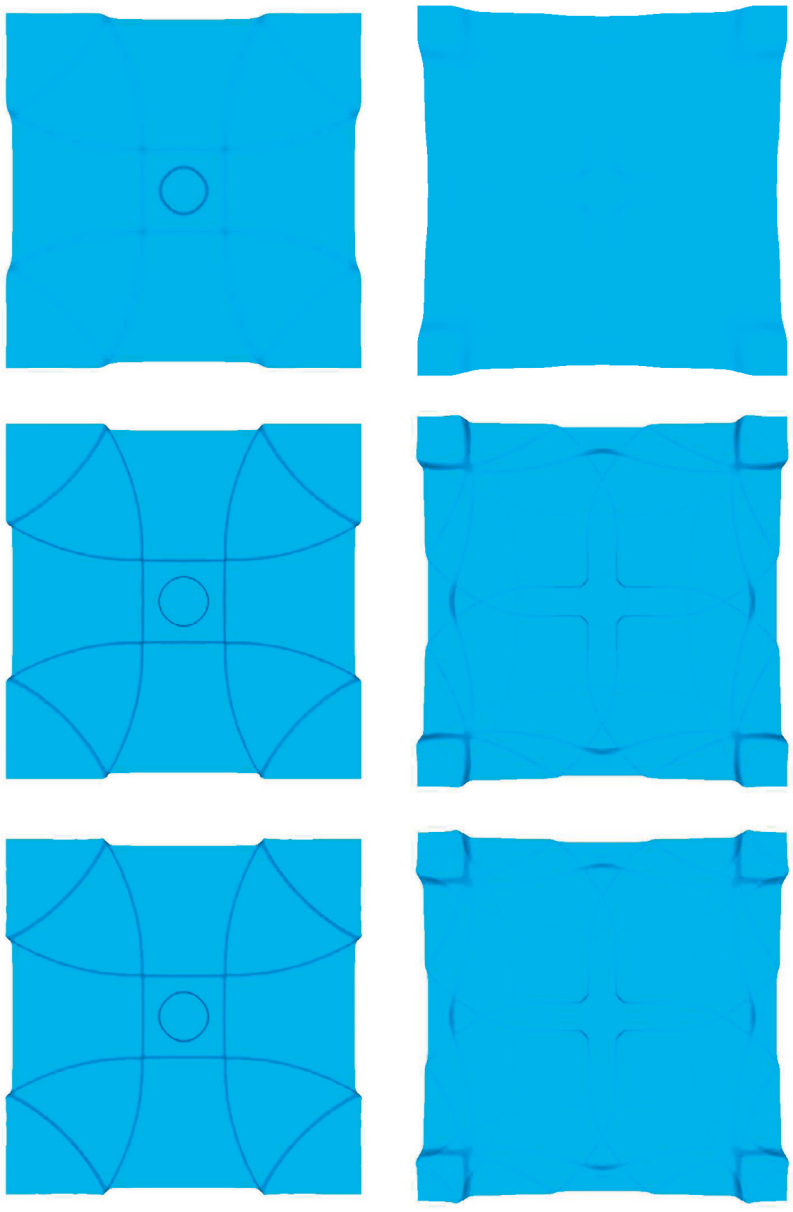


Fig. 5. General steps of the CUDA implementation for the second order HLL scheme.

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

N. cells	GTX 480			GTX 580		
	1st order HLL	2nd order HLL	TVD-WAF	1st order HLL	2nd order HLL	TVD-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



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

Fig. 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: 1st order HLL, 2nd order HLL, and TVD-WAF.

- *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

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	3.28×10^{-01}	–	7.96×10^{-01}	–	1.79	–
50×50	1.75×10^{-01}	0.90	4.65×10^{-01}	0.77	1.04	0.78
100×100	8.97×10^{-02}	0.97	2.48×10^{-01}	0.91	5.57×10^{-01}	0.91
200×200	4.32×10^{-02}	1.05	1.22×10^{-01}	1.02	2.73×10^{-01}	1.02
400×400	2.11×10^{-02}	1.05	5.88×10^{-02}	1.05	1.30×10^{-01}	1.05

the row numbers for the horizontal edges in the finite volume mesh, assuming that they are numbered starting with 0 (see Fig. 4b). 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 Fig. 4b which illustrates the different edge processing substeps. As a consequence, the edges (i.e. threads) of a single kernel does not need to synchronize each other when contributing to a particular volume (the synchronization between threads of different kernels is achieved by the implicit synchronization between CUDA kernels). 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 stored in global memory. In the *n*th time step, the element of the accumulator which corresponds to the volume V_i stores the contribution of its neighboring 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 division of the edge processing into these four kernels makes it possible to add a positivity step in each edge processing kernel, which consists in an adjustment of the edge contributions in order to guarantee the positivity of the water heights. In the first order HLL scheme each edge needs the data of its two adjacent volumes, while in the TVD-WAF method each edge needs the data of four volumes: its two closer left and right volumes if the edge is vertical, or its two closer upper and lower volumes if the edge is horizontal. In the four edge processing kernels we assign a size of 48 KB to L1 cache and 16 KB to shared memory.

- *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. In the same way as the reading of the Z_i value, since there is only one accumulator, no further reduction step is needed for obtaining the M_i value. Since the kernel cannot write directly into the 2D texture used to store U_i^{n+1}, after this step, this 2D texture (containing the volume data) must be updated by copying the accumulator array to the CUDA array bound to the texture.
- *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. Since there is only one accumulator, the Z_i value is obtained directly without performing any reduction operation.
- *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 [12].

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 Fig. 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 and initialize Δt*: 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 as the volume data array (see Section 5.1) is defined to store the new vector state $U^{n+1/2}$. This array is also stored as a 2D texture.
- *Edge processing*: In a similar way as in Section 5.1, this step is divided into four substeps (each substep is implemented using a different CUDA kernel as shown in Fig. 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 (each edge needs the data of the same four volumes as in the TVD-WAF scheme) and also computes part of the volume integral appearing in (16) using the trapezoidal rule. The resulting contributions are also added to the partial sums M_i and M_j associated to V_i and V_j, respectively. Note that, the previous computations require the use of the reconstruction values, U_{ij}^-, U_{ij}^+, as well as the reconstructed topography values, H_{ij}^-, H_{ij}^+. The kernels of this step require more registers of each multiprocessor than in Section 5.1 and are more intensive computationally because the reconstructed values and the integral terms must be computed as intermediate results. In this kernels we also assign a size of 48 KB to L1 cache and 16 KB to shared memory.
- *Compute Δt_i for each volume and Get minimum Δt*: These two kernels are similar to the corresponding kernels defined in Section 5.1.
- *Compute $U_i^{n+1/2}$ for each volume*: This step corresponds with the first stage of the Runge–Kutta TVD scheme and it is identical to the step which compute U_i^{n+1} in Section 5.1. After this step, the 2D texture used to store $U_i^{n+1/2}$ is updated from the accumulator array.
- *Compute U_i^{n+1} for each volume*: This step implements the second stage of the Runge–Kutta scheme. After repeating the Edge processing, U_i^{n+1} is computed using U_i^n, $U_i^{n+1/2}$ and M_i using a similar procedure than the one described in Section 5.1. After this step, the 2D texture containing the volume data must be updated from the accumulator array.

6. Experimental results

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

Table 3
Accuracy test: 2nd 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	8.67×10^{-02}	–	2.87×10^{-01}	–	4.90×10^{-02}	–
50×50	3.22×10^{-02}	1.42	1.09×10^{-01}	1.38	2.01×10^{-02}	1.28
100×100	8.81×10^{-03}	1.87	3.12×10^{-02}	1.81	6.18×10^{-03}	1.70
200×200	2.32×10^{-03}	1.92	8.12×10^{-03}	1.94	1.67×10^{-04}	1.88
400×400	6.02×10^{-04}	1.95	2.11×10^{-03}	1.94	4.38×10^{-05}	1.93

Table 4
Accuracy test: TVD-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	1.12×10^{-01}	–	3.81×10^{-01}	–	6.94×10^{-01}	–
50×50	4.37×10^{-02}	1.36	1.79×10^{-01}	1.08	2.74×10^{-01}	1.33
100×100	1.78×10^{-02}	1.29	8.22×10^{-02}	1.12	1.14×10^{-01}	1.25
200×200	7.19×10^{-03}	1.31	3.54×10^{-02}	1.21	5.04×10^{-02}	1.18
400×400	2.89×10^{-03}	1.31	1.52×10^{-02}	1.21	2.17×10^{-02}	1.21

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 $H(x,y) = 1 - 0.8e^{-x^2-y^2}$ and the 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. Fig. 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 TVD-WAF method is slightly slower that the first order HLL method and two times 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 [14]. These versions have been run on an Intel Core i7 920 processor with 4 GB RAM. The GPU implementations reach speedups of more than 200 for the three numerical schemes in both graphics cards with respect to the monocore CPU version, and approximately 80 with respect to the quadcore version using the GTX 580 card.

We have analyzed the influence of the wet/dry treatment and the friction term in the speedup reached. Since the dealing of wet/dry fronts involves very few mathematical operations, its influence in the speedup is almost negligible. On the other hand, the dealing of the friction term is more complex and the speedup achieved has reduced approximately the 4% when adding this treatment in both CPU and GPU implementations.

6.2. Accuracy test

Next, we consider a test proposed in [31] 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

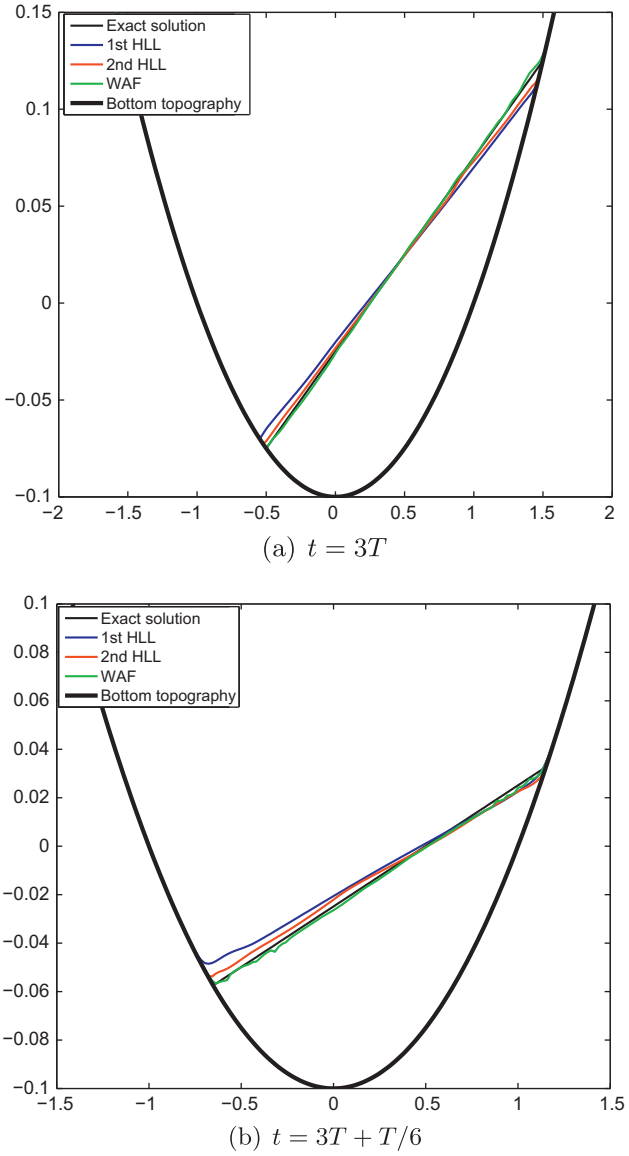


Fig. 7. 2-d oscillating lake: surface elevation vs. y-coordinate, for $y = 0$ at different times steps: Exact solution in black, 1st order HLL in blue, 2nd order HLL in red and TVD-WAF in green. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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

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 TVD-WAF scheme is not second order of accuracy, but its convergence rate is grater than one for this test (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 Castro et al. [9] 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 [20].

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 [30]:

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

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 Fig. 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 TVD-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 neighborhood of El Limonero Dam. This dam is located on the

(a) t=1m 50s (b) t=4m 45s

(c) t=5m 41s (d) t=19m 59s

Fig. 8. Views of the evolution of the flood after the dam break computed with the TVD-WAF method.

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 heterogeneous loose material with a impermeable center. 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 h m^3.

The considered domain is a square of 3260 m width and 8000 m long, discretized using 1.043 millions cell of 5 m $\times$ 5 m. 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. Fig. 8 shows the evolution of the flood at different time steps computed with the TVD-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 is considered. This technique is specially suitable for GPU implementation. This two dimensional TVD-WAF method is not second order of accuracy, but we show in the numerical tests that it is two times faster than the second order HLL method, providing almost the same numerical results. This reformulation of the TVD-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.

Acknowledgements

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.

References

[1] de la Asunción M, Mantas JM, Castro MJ. Simulation of one-layer shallow water systems on multicore and CUDA architectures. J Supercomput 2011;58(2):206–14.
[2] de la Asunción M, Mantas JM, Castro MJ. Programming CUDA-based GPUs to simulate two-layer shallow water flows. In: Euro-Par 2010 – parallel processing. Lecture notes in computer science, vol. 6272. Berlin/Heidelberg: Springer; 2010. p. 353–64.
[3] de la Asunción M, Mantas JM, Castro MJ, Fernández ED. An MPI-CUDA implementation of an improved Roe method for two-layer shallow water systems. J Parallel Distrib Comput, in press, doi:10.1016/j.jpdc.2011.07.012.
[4] Audusse E, Bouchut F, Bristeau M-O, Klein R, Perthame B. A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. SIAM J Sci Comput 2004;25(6):2050–65.
[5] Billet SJ, Toro EF. On WAF-type schemes for multidimensional hyperbolic conservation laws. J Comput Phys 1997;130:1–24.
[6] Bollermann A, Noelle S, Lukácová-Medvid'ová M. Finite volume evolution Galerkin methods for the shallow water equations with dry beds. Commun Comput Phys 2011;10:371–404.
[7] Bradford SF, Sanders BF. Finite-volume model for shallow-water flooding of arbitrary topography. J Hydraul Eng 2002;128(3):289–98.
[8] Brodtkorb A, Hagen T, Lie K-A, Natvig J. Simulation and visualization of the Saint–Venant system using GPUs. Comput Visual Sci 2011:1–13.
[9] Castro MJ, Ferreiro A, García JA, González-Vida J, Macías J, Parés C, et al. On the numerical treatment of wet/dry fronts in shallow flows: application to one-layer and two-layers systems. Math Comput Model 2005;42(3-4):419–39.
[10] Castro MJ, Fernández ED, Ferreiro AM, García A, Parés C. High order extension of Roe schemes for two dimensional nonconservative hyperbolic systems. J Sci Comput 2009;39:67–114.
[11] Castro MJ, Ortega S, de la Asunción M, Mantas JM, Gallardo JM. GPU computing for shallow water flow simulation based on finite volume schemes. CR Méc 2011;339(2–3):165–84.
[12] NVIDIA, CUDA home page. <http://www.nvidia.com/object/cuda_home_new.html>.
[13] Davis SF. Simplified second-order Godunov-type methods. SIAM J Sci Stat Comput 1988;9:445–73.
[14] Eigen 3.0.1. <http://eigen.tuxfamily.org>.
[15] Fernández-Nieto ED, Narbona-Reina G. Extension of waf type methods to nonhomogeneous shallow water equations with pollutant. J Sci Comput 2008;36(2):193–217.
[16] Fernández-Nieto ED, Bresch D, Monnier J. A consistent intermediate wave speed for a well-balanced HLLC solver. CR Math Acad Sci Paris 2008;346(13-14):795–800.
[17] Gallardo J, Ortega S, de la Asunción M, Mantas JM. Two-dimensional compact third-order polynomial reconstructions. Solving nonconservative hyperbolic systems using GPUs. J Sci Comput 2011:1–23.
[18] Geveler M, Ribbrock D, Mallach S. A simulation suite for lattice-Boltzmann based real-time CFD applications exploiting multi-level parallelism on modern multi- and many-core architectures. J Comput Sci 2011;2(2):113–23.
[19] Harten A, Lax PD, van Leer B. On upstream differencing and Godunovtype schemes for hyperbolic conservation laws. SIAM Rev 1983;25(1):35–61.
[20] Kurganov A, Petrova G. A second-order well-balanced positivity preserving central-upwind scheme for the Saint–Venant system. Commun Math Sci 2007;5(1):133–60.
[21] Loukili Y, Soulaymani A. Numerical tracking of shallow water waves by the unstructured finite volume WAF approximation. Int J Comput Methods Eng Sci Mech 2007;8(2):75–88.
[22] Parés C, Castro MJ. On the well-balance property of Roe's method for nonconservative hyperbolic systems. Applications to shallow-water systems. ESAIM: M2AN 2004;38(5):821–52.
[23] Parés C. Numerical methods for nonconservative hyperbolic systems: a theoretical framework. SIAM J Numer Anal 2006;44:300–21.
[24] Saetra ML, Brodtkorb AR. Shallow water simulations on multiple GPUs. In: Proceedings of the Para 2010 conference Part II, lecture notes in computer science, vol. 7134. Springer; 2012. p. 56–66.
[25] Shu C-W. Total variation diminishing time discretizations. SIAM J Sci Stat Comput 1988;9(6):1073–84.
[26] Toro EF. A weighted average flux method for hyperbolic conservation laws. Proc Roy Soc Lond A 1989;423:401–18.
[27] Toro EF. Riemann problems and the WAF method for solving two-dimensional shallow water equations. Phil Trans Roy Soc Lond 1992;A338:43–68.
[28] Toro EF. The weighted average flux method applied to the time dependent euler equations. Phil Trans Roy Soc Lond 1992;A341:499–530.
[29] Toro EF. Shock-capturing methods for free-surface shallow flows. England: Wiley; 2001.
[30] Thacker WC. Some exact solutions to the nonlinear shallow-water wave equations. J Fluid Mech 1981;107:499–508.
[31] Xing Y, Shu C-W. High order finite difference WENO schemes with the exact conservation property for the shallow water equations. J Comput Phys 2005;208:206–27.