COMMUN. MATH. SCI.
Vol. 21, No. 3, pp. 671–708

ADAPTIVE CENTRAL-UPWIND SCHEME ON TRIANGULAR GRIDS FOR THE SAINT-VENANT SYSTEM*

YEKATERINA EPSHTEYN† AND THUONG NGUYEN‡

Abstract. In this work, we develop a robust adaptive well-balanced and positivity-preserving central-upwind scheme on unstructured triangular grids for shallow water equations. The numerical method is an extension of the scheme from [Liu *et al.*, J. Comput. Phys., 374:213–236, 2018]. As a part of the adaptive central-upwind algorithm, we obtain a local a posterior error estimator for the efficient mesh refinement strategy. The accuracy, high-resolution and efficiency of new adaptive central-upwind scheme are demonstrated on a number of challenging tests for shallow water models.

Keywords. Saint-Venant system of shallow water equations; central-upwind scheme; well-balanced and positivity-preserving scheme; adaptive algorithm; weak local residual error estimator; unstructured triangular grid.

AMS subject classifications. 76M12; 65M08; 35L65; 86-08; 86A05.

1. Introduction

We consider the two-dimensional (2-D) Saint-Venant system of shallow water equations,

$$h_t+(hu)_x+(hv)_y=0, \tag{1.1a}$$

$$(hu)_t+\left(hu^2+\frac{g}{2}h^2\right)_x+(huv)_y=-ghB_x, \tag{1.1b}$$

$$(hv)_t+(huv)_x+\left(hv^2+\frac{g}{2}h^2\right)_y=-ghB_y, \tag{1.1c}$$

where t is the time, x and y are horizontal spatial coordinates ($(x,y)\in\Omega$), $h(x,y,t)$ is the water height, $u(x,y,t)$ and $v(x,y,t)$ are the x- and y-components of the flow velocity, $B(x,y)$ is the bottom topography, and g is the constant gravitational acceleration. The system (1.1a–1.1c) was originally proposed in [12], but it is still widely used to model water flow in rivers, lakes and coastal areas, to name a few examples. The Saint-Venant system (1.1a–1.1c) is an example of the hyperbolic system of balance/conservation laws. The design of robust and accurate numerical algorithms for the computation of its solutions is an important and challenging problem that has been extensively studied in the recent years.

An accurate numerical scheme for shallow water Equations (1.1a–1.1c) should preserve the physical properties of the flow. For example, **(i)** the numerical method should be positivity preserving, that is, the water height h should be nonnegative at all times. The positivity preserving property ensures a robust performance of the algorithm on dry (h is zero) or almost dry (h is near zero) states; **(ii)** in addition, the numerical method for system (1.1a–1.1c) should be well-balanced, the method should exactly preserve the "lake-at-rest" solution, $h+B\equiv const, u\equiv 0, v\equiv 0$. This property diminishes the appearance of unphysical oscillations of magnitude proportional to the grid size. In the past decade, several well-balanced [1, 2, 4–7, 15–18, 22, 23, 25, 30, 33–36, 39–42, 46, 47]

*Received: June 25, 2021; Accepted (in revised form): July 08, 2022. Communicated by François Bouchut.

†Department of Mathematics, The University of Utah, Salt Lake City, UT 84112, USA (epshteyn@math.utah.edu).

‡Department of Mathematics, The University of Utah, Salt Lake City, UT 84112, USA (tnguyen@math.utah.edu).

and positivity preserving [1,2,4–6,8,25,30,35,36,40] schemes (non-exhaustive lists) for shallow water models have been proposed, but only few satisfy both major properties **(i)** and **(ii)** simultaneously.

The traditional numerical methods for system (1.1a–1.1c) consider very fine fixed meshes to reconstruct delicate features of the solution. However, this can lead to high computational cost. Therefore, the main goal of this work is to design adaptive numerical algorithms for shallow water equations. In this work, we extend the numerical method in [36] to an adaptive well-balanced and positivity-preserving central-upwind finite volume method on unstructured triangular grids. The central Nessyahu-Tadmor schemes, their generalization into higher resolution central schemes and semi-discrete central-upwind schemes are a family of efficient and accurate Godunov-type Riemann problem-free projection-evolution finite volume methods for hyperbolic problems. They were originally developed in [28,31,38]. The main advantages of these numerical algorithms are high-resolution, efficiency and simplicity. The class of central-upwind methods has been successfully used for problems in science and engineering, including, for geophysical flow problems and related models, e.g. [3,5–7,9–11,25–32,36,44]. There is some very recent effort on the design of adaptive well-balanced and positivity-preserving central-upwind schemes on quad-tree grids for shallow water models [19,43], but no research has been done for the development of such adaptive schemes on unstructured triangular grids. However, triangular grids are efficient or even inevitable when dealing with complex geometries such as modeling the ocean waves near the shore, dam break, stream channels, etc. In addition, the well-balanced wet/dry reconstruction takes advantage of the triangular mesh, see Section 2 and [36], and such an algorithm is not applicable to the rectangular meshes.

This paper is organized as follows. In Section 2, we briefly review the well-balanced positivity-preserving central-upwind scheme on unstructured triangular grids [36] which serves as the underlying discretization for the developed adaptive algorithm. We give a summary of the adaptive central-upwind method in Section 3.1. We discuss the adaptive mesh refinement strategy in Section 3.2. In Section 3.3, we present the adaptive second-order strong stability preserving Runge-Kutta method, employed as a part of the time evolution for the adaptive central-upwind scheme. We derive a local a posterior error estimator in Section 3.4 which is used as a robust indicator for the adaptive mesh refinement in our work. Finally, in Section 4, we illustrate the high accuracy and efficiency of the developed adaptive central-upwind scheme on a number of challenging tests for shallow water models.

2. Semi-discrete central-upwind scheme–an overview

In this work, we employ the central-upwind scheme discussed in this section as the underlying discretization for the adaptive central-upwind algorithm, developed in Section 3. Therefore, in this section, we will briefly review a semi-discrete second-order well-balanced positivity preserving central-upwind scheme on unstructured triangular grids for the Saint-Venant system of shallow water equations [6,36].

In the first work [6], a new second-order semi-discrete central-upwind scheme was developed for computing the solutions of the system (1.1a–1.1c) on unstructured triangular grids. The key ideas in the development of the scheme in [6] were: **(1)** Change of variables from $(h,hu,hv)^T$ to variables $(w:=h+B,hu,hv)^T$. This change of variables simplifies the construction of the well-balanced scheme since in the "lake-at-rest" steady-state, it is the equilibrium variable, the water surface $w\equiv h+B$ (but not the conservative variable, the water height h) that has to stay constant; **(2)** Replacement of the bottom topography function B with its continuous piecewise linear approximation; **(3)** De-

sign of the special positivity preserving correction of the piecewise linear reconstruction for the water surface w; **(4)** Development of a special well-balanced finite-volume-type quadrature for the discretization of the cell averages of the geometric source term. The developed scheme in [6], enforced the positivity of the water height h, and preserved the "lake-at-rest" steady state in the case of fully submerged bottom topography. In the recent work [36], we further improved the well-balanced property of the scheme from [6], and extended the scheme to accurate and stable simulations of shallow water models with dry or near dry states (e.g., waves arriving or leaving the shore). We will briefly review below the central-upwind scheme from [36].

First, we rewrite the system (1.1a–1.1c) in the following equivalent form,

$$\boldsymbol{U}_t+\boldsymbol{F}(\boldsymbol{U},B)_x+\boldsymbol{G}(\boldsymbol{U},B)_y=\boldsymbol{S}(\boldsymbol{U},B), \tag{2.1}$$

where the variables $\boldsymbol{U}$ and the fluxes $\boldsymbol{F}$ and $\boldsymbol{G}$ are

$$\boldsymbol{U}=\begin{pmatrix} w \\ hu \\ hv \end{pmatrix}, \quad \boldsymbol{F}=\begin{pmatrix} hu \\ \dfrac{(hu)^2}{w-B}+\dfrac{g}{2}(w-B)^2 \\ \dfrac{(hu)(hv)}{w-B} \end{pmatrix}, \quad \boldsymbol{G}=\begin{pmatrix} hv \\ \dfrac{(hu)(hv)}{w-B} \\ \dfrac{(hv)^2}{w-B}+\dfrac{g}{2}(w-B)^2 \end{pmatrix},$$

and the source term $\boldsymbol{S}$ is

$$\boldsymbol{S}=\begin{pmatrix} 0 \\ -g(w-B)B_x \\ -g(w-B)B_y \end{pmatrix}.$$

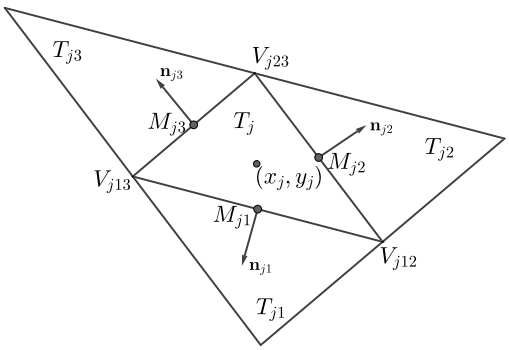


FIG. 2.1. *A typical triangular cell with three neighbors.*

As illustrated in Figure 2.1, we denote,
$\mathcal{T}:=\{T_j\}_j$ is an unstructured triangulation of the computational domain Ω;
$T_j\in\mathcal{T}$ is a triangular cell of size $|T_j|$ with the barycenter (x_j,y_j);
$V_{j\kappa}=(\widetilde{x}_{j\kappa},\widetilde{y}_{j\kappa})$, $\kappa=12,23,31$ are the three vertices of T_j;
T_{jk}, $k=1,2,3$ are the neighboring triangles that share a common side with T_j;
ℓ_{jk} is the length of the common side of T_j and T_{jk}, and M_{jk} is its midpoint;
$\boldsymbol{n}_{jk}:=(\cos(\theta_{jk}),\sin(\theta_{jk}))^\top$ is the outer unit normal to the k-th side of T_j.

Next, in order to develop the positivity-preserving and well-balanced scheme, the bottom topography B is replaced with its continuous piecewise linear approximation $\widetilde{B}$

given by

$$\begin{vmatrix} x-\widetilde{x}_{j12} & y-\widetilde{y}_{j12} & \widetilde{B}(x,y)-\widehat{B}_{j12} \\ \widetilde{x}_{j23}-\widetilde{x}_{j12} & \widetilde{y}_{j23}-\widetilde{y}_{j12} & \widehat{B}_{j23}-\widehat{B}_{j12} \\ \widetilde{x}_{j13}-\widetilde{x}_{j12} & \widetilde{y}_{j13}-\widetilde{y}_{j12} & \widehat{B}_{j13}-\widehat{B}_{j12} \end{vmatrix}=0, \quad (x,y)\in T_j,$$

where, in the case of continuous bottom topography, $\widehat{B}_{j\kappa}:=B(V_{j\kappa})$ $\kappa=12,23,31$. Then, denote:

$$B_{jk}:=\widetilde{B}(M_{jk}), \quad B_j:=\widetilde{B}(x_j,y_j)=\frac{1}{3}(\widehat{B}_{j12}+\widehat{B}_{j23}+\widehat{B}_{j13}).$$

At time t, define by $\overline{\boldsymbol{U}}_j(t)$ the approximation of the cell averages of the solution,

$$\overline{\boldsymbol{U}}_j(t)\approx\frac{1}{|T_j|}\iint\limits_{T_j}\boldsymbol{U}(x,y,t)\,dxdy.$$

Then, it can be shown (see [6,36]), that the semi-discrete second-order central-upwind scheme for the Saint-Venant system (2.1 on triangular grid is given by the following system of ODEs,

$$\frac{d\,\overline{\boldsymbol{U}}_j}{dt}=-\frac{1}{|T_j|}\big[\boldsymbol{H}_{j1}+\boldsymbol{H}_{j2}+\boldsymbol{H}_{j3}\big]+\overline{\boldsymbol{S}}_j, \tag{2.2}$$

where the numerical fluxes through the edges of the triangular cell T_j are

$$\begin{aligned}\boldsymbol{H}_{jk}=&\frac{\ell_{jk}\cos(\theta_{jk})}{a^{\rm in}_{jk}+a^{\rm out}_{jk}}\Big[a^{\rm in}_{jk}\boldsymbol{F}(\boldsymbol{U}_{jk}(M_{jk}),B_{jk})+a^{\rm out}_{jk}\boldsymbol{F}(\boldsymbol{U}_j(M_{jk}),B_{jk})\Big]\\&+\frac{\ell_{jk}\sin(\theta_{jk})}{a^{\rm in}_{jk}+a^{\rm out}_{jk}}\Big[a^{\rm in}_{jk}\boldsymbol{G}(\boldsymbol{U}_{jk}(M_{jk}),B_{jk})+a^{\rm out}_{jk}\boldsymbol{G}(\boldsymbol{U}_j(M_{jk}),B_{jk})\Big]\\&-\ell_{jk}\frac{a^{\rm in}_{jk}a^{\rm out}_{jk}}{a^{\rm in}_{jk}+a^{\rm out}_{jk}}\big[\boldsymbol{U}_{jk}(M_{jk})-\boldsymbol{U}_j(M_{jk})\big], \quad k=1,2,3.\end{aligned} \tag{2.3}$$

Here, $\boldsymbol{U}_j(M_{jk})$ and $\boldsymbol{U}_{jk}(M_{jk})$ are the reconstructed point values of $\boldsymbol{U}$ at the middle points of the edges M_{jk}. To obtain these values [36], first, a piecewise linear reconstruction of the variables $\boldsymbol{\Upsilon}:=(w,u,v)^\top$ is computed as,

$$\widetilde{\boldsymbol{\Upsilon}}(x,y)=\sum_j\boldsymbol{\Upsilon}_j(x,y)\chi_{T_j}, \quad \boldsymbol{\Upsilon}_j(x,y):=\boldsymbol{\Upsilon}_j+(\widehat{\boldsymbol{\Upsilon}}_x)_j(x-x_j)+(\widehat{\boldsymbol{\Upsilon}}_y)_j(y-y_j), \tag{2.4}$$

where χ_{T_j} is the characteristic function of the cell T_j, $\boldsymbol{\Upsilon}_j$ are the point values of $\boldsymbol{\Upsilon}$ at the cell centers and $(\widehat{\boldsymbol{\Upsilon}}_x)_j$ and $(\widehat{\boldsymbol{\Upsilon}}_y)_j$ are the limited partial derivatives. After that, the second and third components of the point values $\boldsymbol{U}_j(M_{jk})$ and $\boldsymbol{U}_{jk}(M_{jk})$ are obtained from, $\boldsymbol{\Upsilon}_j(M_{jk})$ and $\boldsymbol{\Upsilon}_{jk}(M_{jk})$,

$$\begin{aligned}&(hu)_j(M_{jk})=(w_j(M_{jk})-B_{jk})u_j(M_{jk}), \quad (hu)_{jk}(M_{jk})=(w_{jk}(M_{jk})-B_{jk})u_{jk}(M_{jk}),\\&(hv)_j(M_{jk})=(w_j(M_{jk})-B_{jk})v_j(M_{jk}), \quad (hv)_{jk}(M_{jk})=(w_{jk}(M_{jk})-B_{jk})v_{jk}(M_{jk}).\end{aligned}$$

See Section 2 in [36] for more details on the reconstruction.

Moreover, to design a well-balanced central-upwind scheme, a special second-order reconstruction of water surface is introduced in [36] which is positivity preserving for the steady-state solutions with partially flooded/dry cells. Hence, the linear approximation for the water surface is updated as follows:

- In the dry cells in which $\overline{w}_j = B_j$, the corresponding linear pieces for w in (2.4) are replaced by,

$$\widetilde{w}_j(x,y) = \widetilde{B}(x,y). \tag{2.5}$$

- If T_j is partially flooded which means $B_j < \overline{w}_j < \max\{\widehat{B}_{j23}, \widehat{B}_{j13}, \widehat{B}_{j12}\}$, the water surface is reconstructed by using two linear pieces instead of one as,

$$\widetilde{w}_j(x,y) = \begin{cases} \mathring{w}_j(x,y), & \text{if} \quad (x,y) \in T_j^{wet}, \\ \widetilde{B}(x,y), & \text{otherwise}, \end{cases} \tag{2.6}$$

  where $\mathring{w}_j(x,y)$ is a linear reconstruction of the water surface on the wet part T_j^{wet} of the cell T_j.

- If T_j is fully flooded $\overline{w}_j \geq \max\{\widehat{B}_{j23}, \widehat{B}_{j13}, \widehat{B}_{j12}\}$, no further modification for the linear approximation (2.4) is needed.

See Section 3 in [36] for more details of the reconstruction of the water surface w.

In (2.3), a_{jk}^{in} and a_{jk}^{out} are the one-sided local speeds of propagation in the directions $\pm \boldsymbol{n}_{jk}$. These speeds are related to the largest and smallest eigenvalues of the Jacobian matrix $J_{jk} = \cos(\theta_{jk}) \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{U}} + \sin(\theta_{jk}) \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{U}}$, denoted by $\lambda_+[J_{jk}]$ and $\lambda_-[J_{jk}]$, respectively, and are defined by

$$\begin{aligned} a_{jk}^{\text{in}} &= -\min\{\lambda_-[J_{jk}(\boldsymbol{U}_j(M_{jk}))], \lambda_-[J_{jk}(\boldsymbol{U}_{jk}(M_{jk})], 0\}, \\ a_{jk}^{\text{out}} &= \max\{\lambda_+[J_{jk}(\boldsymbol{U}_j(M_{jk}))], \lambda_+[J_{jk}(\boldsymbol{U}_{jk}(M_{jk})], 0\}, \end{aligned} \tag{2.7}$$

where

$$\begin{aligned} \lambda_\pm[J_{jk}(\boldsymbol{U}_j(M_{jk}))] &= \cos(\theta_{jk}) u_j(M_{jk}) + \sin(\theta_{jk}) v_j(M_{jk}) \pm \sqrt{g h_j(M_{jk})}, \\ \lambda_\pm[J_{jk}(\boldsymbol{U}_{jk}(M_{jk}))] &= \cos(\theta_{jk}) u_{jk}(M_{jk}) + \sin(\theta_{jk}) v_{jk}(M_{jk}) \pm \sqrt{g h_{jk}(M_{jk})}. \end{aligned}$$

REMARK 2.1. In order to avoid division by 0 (or by a very small positive number), the numerical flux (2.3) is replaced with

$$\begin{aligned} \boldsymbol{H}_{jk} = &\frac{\ell_{jk} \cos(\theta_{jk})}{2} [\boldsymbol{F}(\boldsymbol{U}_{jk}(M_{jk}), B_{jk}) + \boldsymbol{F}(\boldsymbol{U}_j(M_{jk}), B_{jk})] \\ &+ \frac{\ell_{jk} \sin(\theta_{jk})}{2} [\boldsymbol{G}(\boldsymbol{U}_{jk}(M_{jk}), B_{jk}) + \boldsymbol{G}(\boldsymbol{U}_j(M_{jk}), B_{jk})] \end{aligned}$$

wherever $a_{jk}^{\text{in}} + a_{jk}^{\text{out}} < \sigma$. In all of the reported numerical examples in Section 4, we have taken $\sigma = 10^{-6}$.

A fully discrete scheme can be obtained by numerically solving the ODE system (2.2), (2.3) using a stable and sufficiently accurate ODE solver. The time-step size on each cell $T_j \in \mathcal{T}$ should satisfy the CFL-type condition (see [6]), which can be expressed as,

$$\Delta t < \frac{1}{6} \min_{j,k} \left[\frac{r_{jk}}{\max(a_{jk}^{\text{in}}, a_{jk}^{\text{out}})} \right], \tag{2.8}$$

where r_{j1}, r_{j2} and r_{j3} are the three corresponding altitudes of the triangle T_j. From (2.7), the one-sided local speeds of propagation a_{jk}^{in} and a_{jk}^{out} are nonnegative for all j

and k. However, the condition (2.8) can become too restrictive on partially flooded cells. Thus, for partially flooded cells, the "draining" time-step technique is used to ensure the positivity of the scheme without reducing the time step size (2.8), [4,36]. Namely, first the "draining" time-step $\Delta t_j^{\text{drain}}$ is defined by,

$$\Delta t_j^{\text{drain}} := \frac{|T_j|\ \overline{h}_j^n}{\sum\limits_{k=1}^{3} \max(0, H_{jk}^{(1)})}$$

where $H_{jk}^{(1)}$ is the first component of the numerical flux $\boldsymbol{H}_{jk}$ given by (2.3). Notice that, for fully flooded cells $\Delta t_j^{\text{drain}} = \Delta t$, while for dry cells $\Delta t_j^{\text{drain}} = 0$. Next, the local "draining" time-step Δt_{jk} for each edge k of the cell $T_j \in \mathcal{T}$ is defined as,

$$\Delta t_{jk} = \begin{cases} \min(\Delta t, \Delta t_j^{drain}), & \text{if} \quad H_{jk}^{(1)} > 0, \\ \min(\Delta t, \Delta t_{jk}^{drain}), & \text{if} \quad H_{jk}^{(1)} \le 0, \end{cases} \tag{2.9}$$

where Δt_{jk}^{drain} is the "draining" time-step in the neighboring triangle $T_{jk} \in \mathcal{T}$ of T_j and Δt is computed by (2.8), but with the minimum taken there over the flooded cells only. This procedure of the draining time step is a part of the adaptive SSPRK2 time evolution (3.6a-3.6b) in Section 3.3.

Finally, the cell average of the source term $\boldsymbol{S}_j$ in (2.2),

$$\overline{\boldsymbol{S}}_j(t) \approx \frac{1}{|T_j|} \iint\limits_{T_j} \boldsymbol{S}\big(\boldsymbol{U}(x,y,t), B(x,y)\big)\, dxdy,$$

has to be discretized in a well-balanced manner [36]:

Quadrature for $\overline{S}_j^{(2)}$ is

$$\begin{aligned} \overline{S}_j^{(2)} = &\frac{g}{2|T_j|} \sum_{k=1}^{3} \ell_{jk} \cos(\theta_{jk}) \cdot \frac{\Delta t_{jk}}{\Delta t} \cdot \big[w(M_{jk}) - B(M_{jk})\big]^2 \\ &- \frac{g}{3}\Big[(w_{j12} - \widehat{B}_{j12})\, w_x(V_{12}) + (w_{j23} - \widehat{B}_{j23})\, w_x(V_{23}) + (w_{j13} - \widehat{B}_{j13})\, w_x(V_{13})\Big]. \end{aligned} \tag{2.10}$$

A similar quadrature for $\overline{S}_j^{(3)}$ is

$$\begin{aligned} \overline{S}_j^{(3)} = &\frac{g}{2|T_j|} \sum_{k=1}^{3} \ell_{jk} \sin(\theta_{jk}) \cdot \frac{\Delta t_{jk}}{\Delta t} \cdot \big[w(M_{jk}) - B(M_{jk})\big]^2 \\ &- \frac{g}{3}\Big[(w_{j12} - \widehat{B}_{j12})\, w_y(V_{12}) + (w_{j23} - \widehat{B}_{j23})\, w_y(V_{23}) + (w_{j13} - \widehat{B}_{j13})\, w_y(V_{13})\Big]. \end{aligned} \tag{2.11}$$

REMARK 2.2. Note, that in Section 4, we compare the performance of the adaptive central-upwind scheme developed in Section 3 with the performance of the central-upwind scheme without adaptivity from [36] (see also brief review above). We use standard SSPRK2 time discretization [20] together with the draining time step for the scheme without adaptivity from [36] in numerical experiments in Section 4.

3. Adaptive central-upwind scheme

The traditional numerical schemes are based on the use of very fine fixed meshes to reconstruct delicate features of the solution. This can lead to high computational cost, as well as poor resolution of all small scale features of the problem. In many engineering and scientific applications, it is beneficial to use adaptive meshes for improving the accuracy of the approximation at a much lower cost. Therefore, in this section, we will introduce an efficient and accurate adaptive central-upwind algorithm.

3.1. Adaptive central-upwind algorithm. The adaptive central-upwind algorithm is described briefly by the following steps.

Step 0. At time $t=t^0$, generate the initial uniform grid $\mathcal{T}^{0,0}$.

Step 1. On mesh $\mathcal{T}^{n,\mathcal{M}_n}$, evolve the cell averages $\overline{\boldsymbol{U}}^n$ of the solution from time t^n to $\overline{\boldsymbol{U}}^{n+1}$ at the next time level t^{n+1} by using the adaptive central-upwind scheme (3.6a-3.6b), see Section 3:

- At time t^n, determine the level $l=0,1,...,L$ of each cell/triangle $T_j^{n,\mathcal{M}_n}\in\mathcal{T}^{n,\mathcal{M}_n}$, (3.2), Section 3.3.
- At each time level $t_l^{n,p}, p=0,1,...,\mathcal{P}_l-1$, perform the piecewise polynomial reconstruction (2.4) and compute the point values, Section 2, Section 3.3.
- At each time level $t_l^{n,p}, p=0,1,...,\mathcal{P}_l-1$, calculate the one-sided local speeds of propagation using (2.7), Section 2, Section 3.3.
- At time t^n, calculate the reference time step Δt using (3.4), Section 3.3.
- At each time level $t_l^{n,p}, p=0,1,...,\mathcal{P}_l-1$, compute the local time step for each cell level, (3.5), Section 3.3.
- At each time level $t_l^{n,p}, p=0,1,...,\mathcal{P}_l-1$, compute numerical fluxes and source term in the adaptive central-upwind scheme (3.6a-3.6b), (2.3), (2.10-2.11), Section 2, Section 3.3.

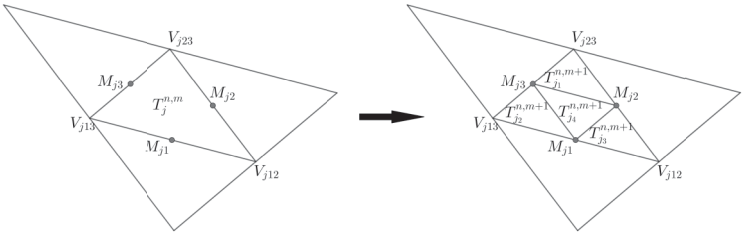
Step 2. On mesh $\mathcal{T}^{n,\mathcal{M}_n}$, compute WLR error using (3.15) in Section 3.4 and update the refinement/de-refinement status for each cell/triangle, Section 3.4.

Step 3. Generate the new adaptive mesh $\mathcal{T}^{n+1,\mathcal{M}_{n+1}}$ at t^{n+1}, Section 3.2. This step includes coarsening of some cells, refinement of some cells, and the appropriate projection of the cell averages at t^{n+1} from the mesh $\mathcal{T}^{n,\mathcal{M}_n}$ onto the new adaptive mesh $\mathcal{T}^{n+1,\mathcal{M}_{n+1}}$, Section 3.2.

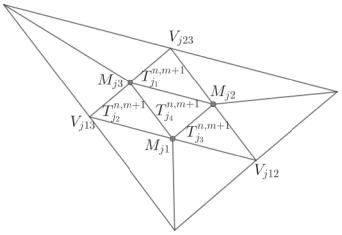
Step 4. Repeat **Step 1**-**Step 3** until final time.

3.2. Adaptive mesh refinement/coarsening. The main idea of the proposed adaptive mesh refinement algorithm is as follows. At time t^n, we start with the given mesh, denoted as $\mathcal{T}^{n,m}=\{T_j^{n,m}\}$, where $T_j^{n,m}$ is a triangular cell of size $|T_j^{n,m}|$ with the barycenter $(x_j^{n,m},y_j^{n,m})$ within the initial mesh $\mathcal{T}^{n,m}$, and index $m=0,1,2...$ is the level of refinement ($m=0$ corresponds to the mesh with no refinement and $\mathcal{T}^{n,0}\equiv\mathcal{T}^{0,0}$ for all n). To flag triangular cells in the mesh $\mathcal{T}^{n,m}$ for the refinement/de-refinement (or coarsening), we use weak local residual (WLR) error estimate, see Section 3.4. We apply "regular refinement" on the triangles flagged for refinement to obtain a new mesh $\mathcal{T}^{n,m+1}$ with the refinement level $m+1$. The "regular refinement" on a fully flooded triangle is obtained by splitting each flagged triangle ("parent" triangle) into four smaller triangles ("children" triangles) by inserting a new node at the mid-point of each edge of the "parent" triangle. We illustrate this idea using Figure 3.1 (a), where we show an example of splitting a flagged triangle $T_j^{n,m}$ by using the mid-points of the sides to

obtain the "children" cells $T_{j_s}^{n,m+1}, s=1,2,3,4$. In addition, the insertion of new nodes on the edges means that non-flagged triangles adjacent to refined triangles get hanging nodes and must also be refined. This is done by inserting a new edge between the hanging node and the opposite corner as illustrated in Figure 3.1 (b). In practice, we



(a) Triangle $T_j^{n,m}$ (left) is split into four "children" cells $T_{j_s}^{n,m+1}, s=1,2,3,4$ (right).



(b) Refinement in the neighboring cells of $T_j^{n,m}$.

FIG. 3.1. *An outline of the "regular refinement".*

may want to reach a higher level of refinement for some cells. This happens when those cells have very large WLR error (3.15), and we need to add more data points. We can obtain a finer cell by repeating the refinement for the flagged triangles in the refined mesh $\mathcal{T}^{n,m+1}$ to get the mesh with higher level $\mathcal{T}^{n,m+2}, m=0,1,2,\ldots..$ Figure 3.2 is the illustration of the "regular refinement" procedure with two levels of refinement.

In the partially flooded cells, Section 2, the approximation of the water surface $\widetilde{w}(x,y)$ at each time level t^n consists of two linear pieces, the piece for the wet and for the dry region, see Figure 3.3. This motivates an idea of the "wet/dry refinement" which uses the boundary between the wet region and the dry region of the cell to refine the partially flooded triangles as shown in the example in Figure 3.4 (left). Namely, consider a partially flooded triangle $T_j^{n,m}$ which is flagged for the refinement and has three non-flagged neighboring cells in the grid $\mathcal{T}^{n,m}$. The segment I_1I_2 is the boundary between the wet and dry interface in that triangle. Note that, the location of the nodes I_1 and I_2 is determined by the second-order water surface reconstruction developed in [36], see also Section 2. During "regular refinement" of the partially flooded cell, we first split the flagged cell $T_j^{n,m}$ into a smaller triangle and a quadrilateral using the wet/dry interface I_1I_2. We then continue to refine the quadrilateral by its diagonal. As can be seen in Figure 3.4 (left), the flagged triangle $T_j^{n,m}$ has three "children" which are either fully flooded or dry. Similarly, as in Figure 3.1, the appearance of two hanging nodes I_1 and I_2 on two sides leads to the need of the further splitting of the neighboring cells as presented in Figure 3.4 (right). The "regular wet/dry refinement" will capture the features of the wet/dry fronts and will minimize the number of partially flooded

"children" cells. However, this method may give us difficulties in controlling the shapes of triangles in the adaptive mesh. In some cases, it may produce "children" cells with unexpected large obtuse angles or very small altitudes, as shown in Figure 3.5. For cells where such situation happens, we instead use the "regular refinement" for the fully flooded cells as described above.

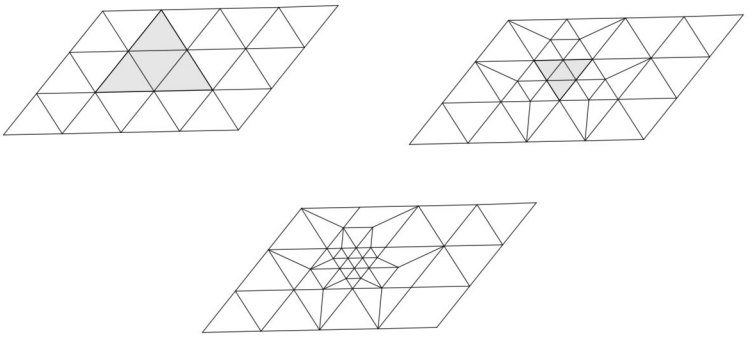


FIG. 3.2. *An example of the "regular refinement" procedure with two levels of refinement. The left figure is the initial coarse mesh* $\mathcal{T}^{n,0}$ *with the region flagged for the refinement (gray). The middle figure is the "first" level mesh* $\mathcal{T}^{n,1}$ *with the region flagged for higher level of the refinement (gray). The right figure is the "second" level mesh* $\mathcal{T}^{n,2}$.

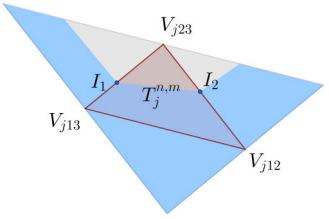


FIG. 3.3. *An example of a partially flooded triangle* $T_j^{n,m}$ *with wet (blue) and dry (gray) regions, where* $T_j^{n,m}$ *is flagged for refinement (red) and its neighboring cells are not flagged.*

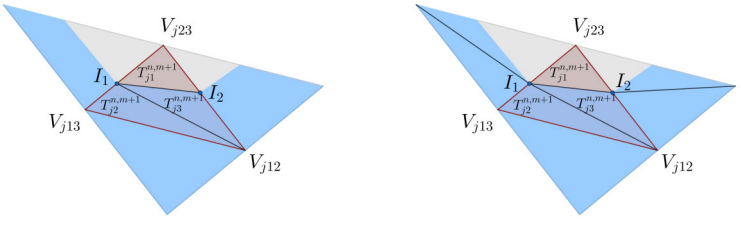


FIG. 3.4. *An example of the refinement of a partially flooded triangle by using the wet/dry interface* I_1I_2 *(left) and the refinement of the neighboring triangles (right).*

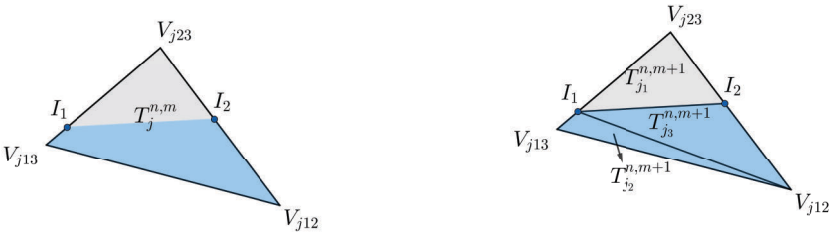


FIG. 3.5. *An example of a refinement of a partially flooded triangle (left) using idea of "wet/dry refinement" that produces "child" cell with large obtuse angle (right).*

Very often in the numerical simulations of the wave phenomena, the regions of the domain that need to be refined move with time. Hence, the refinement in some cells may be no longer needed. The de-refinement or coarsening procedure is then introduced to deactivate unnecessarily fine cells in the mesh. The de-refinement is performed by coarsening (by deactivating "children" cells) in the triangles of the mesh flagged for coarsening (and possibly deactivating finer neighboring triangles due to removal of the hanging nodes). At time t^n, "children" cells in the mesh $\mathcal{T}^{n,m+1}, m=0,1,...,M_n-1$, are deactivated based on the WLR and the corresponding "parent" cell from the mesh $\mathcal{T}^{n,m}$ is activated back. In order to minimize the complexity of the adaptive grid generation, the de-refinement should be applied on all cells flagged for coarsening prior to the refinement, see for example [21].

The refinement/de-refinement process at time t^n produces a hierarchical system of grids $\mathcal{S}^n=\{\mathcal{T}^{n,0},\mathcal{T}^{n,1},\mathcal{T}^{n,2},...,\mathcal{T}^{n,\mathcal{M}_n}\}$, where $\mathcal{T}^{n,m}$, $m=1,2,...,\mathcal{M}_n$ is the grid with the level of refinement m obtained by refining the grid $\mathcal{T}^{n,m-1}$. The term $\mathcal{M}_n$ is the highest refinement level of the hierarchical system at time t^n. In the numerical experiments, Section 4, we assign two values to $\mathcal{M}_n$ which are either $\mathcal{M}_n=1$ or $\mathcal{M}_n=2$. The final mesh $\mathcal{T}^{n,\mathcal{M}_n}\in\mathcal{S}^n$ is the mesh that is used in the adaptive central-upwind scheme at time level t^n. After the evolution of the numerical solution from time t^n to time t^{n+1} using mesh $\mathcal{T}^{n,\mathcal{M}_n}\in\mathcal{S}^n$, we proceed with the generation of a new adaptive grid $\mathcal{T}^{n+1,\mathcal{M}_{n+1}}\in\mathcal{S}^{n+1}$ from the mesh $\mathcal{T}^{n,\mathcal{M}_n}$, using WLR in Section 3.4. After a new adaptive mesh $\mathcal{T}^{n+1,\mathcal{M}_{n+1}}$ is constructed, the obtained cell averages $\overline{\boldsymbol{U}}^{n+1}$ on the mesh $\mathcal{T}^{n,\mathcal{M}_n}$ need to be projected accurately on the new mesh $\mathcal{T}^{n+1,\mathcal{M}_{n+1}}$, using the ideas as summarized briefly below.

Case 1. If a triangle $T_j^{n+1,\mathcal{M}_{n+1}}\in\mathcal{T}^{n+1,\mathcal{M}_{n+1}}$ at t^{n+1} is the same cell as in the grid $\mathcal{T}^{n,\mathcal{M}_n}$, we will keep without any change the cell averages for that triangle at t^{n+1}.

Case 2. A cell $T_j^{n+1,\mathcal{M}_{n+1}}\in\mathcal{T}^{n+1,\mathcal{M}_{n+1}}$ is obtained by coarsening some finer cells $T_{j_s}^{n,\mathcal{M}_n}\in\mathcal{T}^{n,\mathcal{M}_n}, s=1,2,..,S$. In order to enforce the conservation, the solution, $\overline{U}_j^{n+1}$ in the cell $T_j^{n+1,\mathcal{M}_{n+1}}$, is computed as

$$\overline{U}_j^{n+1}=\frac{1}{|T_j^{n+1,\mathcal{M}_{n+1}}|}\sum_{s=1}^{S}\overline{U}_{j_s}^{n+1}|T_{j_s}^{n,\mathcal{M}_n}|,$$

where $\overline{U}_{j_s}^{n+1}$ is the solution at t^{n+1} in $T_{j_s}^{n,\mathcal{M}_n}$.

Case 3. A triangle $T_j^{n+1,\mathcal{M}_{n+1}} \in \mathcal{T}^{n+1,\mathcal{M}_{n+1}}$ is obtained from the refinement of the cell $T_i^{n,\mathcal{M}_n} \in \mathcal{T}^{n,\mathcal{M}_n}$. The approximation of the cell averages of the solution at t^{n+1} in $T_j^{n+1,\mathcal{M}_{n+1}}$ is obtained by using the evaluation of the piecewise linear reconstruction (2.4–2.6) of the solution at t^{n+1} in the triangle $T_i^{n,\mathcal{M}_n}$. Namely, suppose $\boldsymbol{\Upsilon}_i(x,y) = (\widetilde{w}_i, u_i, v_i)^\top(x,y)$ is obtained via the piecewise linear reconstruction (2.4–2.6) in the coarse cell $T_i^{n,\mathcal{M}_n}$ at t^{n+1}. The cell averages in triangle $T_j^{n+1,\mathcal{M}_{n+1}}$ at t^{n+1} are calculated by,

$$\begin{aligned}
\overline{w}_j &= \frac{1}{|T_j^{n+1,\mathcal{M}_{n+1}}|} \iint\limits_{T_j^{n+1,\mathcal{M}_{n+1}}} \widetilde{w}_i(x,y)dxdy,\\
\overline{hu}_j &= (\overline{w}_j - B_j)u_i(x_j,y_j),\\
\overline{hv}_j &= (\overline{w}_j - B_j)v_i(x_j,y_j),
\end{aligned} \tag{3.1}$$

where B_j is the point value of the bottom level at the center (x_j,y_j) of triangle $T_j^{n+1,\mathcal{M}_{n+1}}$.

The adaptive grid $\mathcal{T}^{n+1,\mathcal{M}_{n+1}}$ with the cell averages of the solution reconstructed above is used to perform a new time evolution, see Section 3.3. As a part of the time evolution, at time t^{n+1}, we approximate the point values of the solution by applying the piecewise linear reconstruction (2.4–2.6) on the adaptive grid $\mathcal{T}^{n+1,\mathcal{M}_{n+1}}$.

Note that the cell averages at t^{n+1} in $\mathcal{T}^{n+1,\mathcal{M}_{n+1}}$ are calculated via the projection of the well-balanced piecewise linear reconstruction of the solution from grid $\mathcal{T}^{n,\mathcal{M}_n}$ onto grid $\mathcal{T}^{n+1,\mathcal{M}_{n+1}}$. In addition, in this research, we utilize the well-balanced discretization of the source term proposed in [36]. Therefore, the well-balanced property of the central-upwind method originally developed in [36] is also preserved in the adaptive grids.

3.3. Second-order adaptive time evolution. The CFL-type condition (2.8) is needed for numerical stability. Hence, use of a global time step in the adaptive algorithm may lead to the time step becoming very small due to the presence of much finer cells in the mesh. To improve the computational cost of the algorithm, we consider the approach based on the adaptive time step from [13, 14, 37]. The main idea of the adaptive time evolution is that we group cells into different levels based on the cell sizes. After that, we evolve the solution on each cell level individually with its local time step. This approach does not violate the stability of the explicit time discretization scheme as was shown in [13, 37]. Below, we present the brief summary of the adaptive time evolution algorithm based on the second-order strong stability preserving Runge-Kutta methods (SSPRK2) in [13, 14, 20, 37].

The idea and the order of steps of adaptive SSPRK2 are illustrated on the example in Figure 3.6. First, we group all cells $T_j^{n,\mathcal{M}_n} \in \mathcal{T}^{n,\mathcal{M}_n}$ at time t^n in cell levels $l = 0,1,..,L$ based on their sizes. We define cell levels at t^n in mesh $\mathcal{T}^{n,\mathcal{M}_n}$ as follows. A cell $T_j^{n,\mathcal{M}_n} \in \mathcal{T}^{n,\mathcal{M}_n}$ belongs to the level l, if l is the smallest positive integer satisfying,

$$2^l \geq \frac{\max\limits_j \left(\min\limits_k (r_{jk}) \right)}{\min\limits_k (r_{j,k})}, \tag{3.2}$$

where $r_{jk}, k = 1,2,3$ are three altitudes of triangle $T_j^{n,\mathcal{M}_n}$. Thus, the cell levels with larger index l will contain finer cells which will require a smaller time step (2.8) for

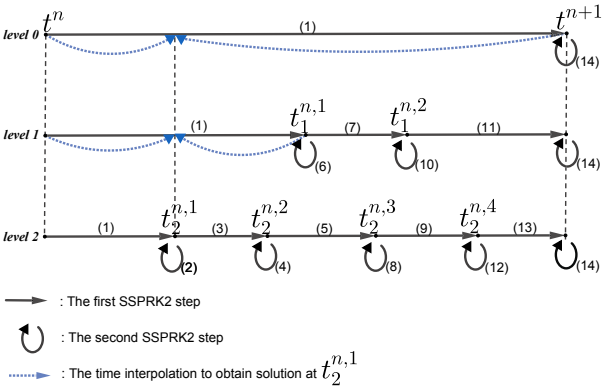


FIG. 3.6. *The example of SSPRK2 on mesh with three cell levels,* $l=0,1,2$.

the evolution from $t^n \to t^{n+1}$. Next, at t^n, we define the reference time step Δt as the local time step on the coarsest level $l=0$ of cells in the mesh $\mathcal{T}^{n,\mathcal{M}_n}$ by considering the CFL-type condition (2.8) locally on level $l=0$. Namely, define first a_{max},

$$a_{max} := \max_{j,k}(a_{jk}^{\text{in}}, a_{jk}^{\text{out}}), \tag{3.3}$$

where $(a_{jk}^{\text{in}}, a_{jk}^{\text{out}})$ are the local one-sided speeds of propagation (2.7) at t^n for sides $k=1,2,3$ in the triangle $T_j^{n,\mathcal{M}_n} \in \mathcal{T}^{n,\mathcal{M}_n}$. Then, the reference time step Δt is computed as,

$$\Delta t \equiv \Delta t_0^{n,0} = \frac{0.9 \max\limits_j \left(\min\limits_k (r_{jk}) \right)}{6 a_{max}}. \tag{3.4}$$

We set, $t^{n+1} = t^n + \Delta t$.

Assume next, that $\mathcal{P}_l$ is the number of steps taken on higher levels $l=1,..L$ to evolve from t^n to t^{n+1}, namely $[t^n, t^{n+1}] = \cup [t_l^{n,p}, t_l^{n,p+1}], p=0,...,\mathcal{P}_l - 1$ with $t_l^{n,0} \equiv t^n$, $t_l^{n,\mathcal{P}_l} \equiv t^{n+1} \quad \forall l$. We define the local time step for cells on these levels $l=1,...,L$ as,

$$\Delta t_l^{n,p} = \frac{2^{-l} \Delta t}{\max(\mu_l^{n,p}, 1)}, \tag{3.5}$$

where parameter $\mu_l^{n,p}$ takes into account change in the local one-sided speeds of the propagation,

$$\mu_l^{n,p} = \frac{\max\limits_{j,k} (a_{jk}^{\text{in}}, a_{jk}^{\text{out}})_l^{n,p}}{a_{max}},$$

where $(a_{jk}^{\text{in}}, a_{jk}^{\text{out}})_l^{n,p}$ are the local one-sided speeds of propagation at $t_l^{n,p}$ of the cell $T_j^{n,\mathcal{M}_n}$ in the level l. Therefore, on each cell $T_j^{n,\mathcal{M}_n}$ of level l, for each substep $[t_l^{n,p}, t_l^{n,p+1}] \equiv [t_l^{n,p}, t_l^{n,p} + \Delta t_l^{n,p}], p=0,1,2,3,...,\mathcal{P}_l - 1$ of the evolution from t^n to t^{n+1}, we apply the following two adaptive steps of SSPRK2 method,

$$\overline{\boldsymbol{U}}_j^{(1)} = \overline{\boldsymbol{U}}_j^{n,p} - \frac{1}{|T_j^{n,\mathcal{M}_n}|} \sum_{k=1}^{3} \Delta t_{jk}^{n,p} \boldsymbol{H}_{jk}^{n,p} + \Delta t_l^{n,p} \ \overline{\boldsymbol{S}}_j^{n,p} := \boldsymbol{R}(\overline{\boldsymbol{U}}_j^{n,p}, \Delta t_l^{n,p}), \tag{3.6a}$$

$$\overline{\boldsymbol{U}}_j^{n,p+1} = \frac{1}{2}\,\overline{\boldsymbol{U}}_j^{n,p} + \frac{1}{2}\boldsymbol{R}(\,\overline{\boldsymbol{U}}_j^{(1)}, \Delta t_l^{n,p}). \tag{3.6b}$$

Here, $\Delta t_l^{n,p}$ is the local time step of the cells of level l at time $t_l^{n,p}$ (3.5), and $\Delta t_{jk}^{n,p}$ is the "draining" time step in cell $T_j^{n,\mathcal{M}_n}$ for the local time step $\Delta t_l^{n,p}$ in level l. The flux term $\boldsymbol{H}_{jk}^{n,p}$ in (3.6a -3.6b) is the flux (2.3) computed at $t = t_l^{n,p}$. The source term $\overline{\boldsymbol{S}}_j^{n,p}$ in (3.6a -3.6b) is the source (2.10 - 2.11) computed at $t = t_l^{n,p}$ with the time step $\Delta t_l^{n,p}$ and with the corresponding local "draining" time step $\Delta t_{jk}^{n,p}$. Note that, $\overline{\boldsymbol{U}}_j^{n,0} \equiv \overline{\boldsymbol{U}}_j^{n}$ and $\overline{\boldsymbol{U}}_j^{n,\mathcal{P}_l} \equiv \overline{\boldsymbol{U}}_j^{n+1}$.

REMARK 3.1. Figure 3.6 describes an example of the adaptive time evolution where the cells in the adaptive mesh at t^n are categorized based on their sizes (see (3.2)) into three levels $l = 0,1,2$. As can be seen in Figure 3.6, the adaptive time evolution can be briefly described as follows.

- We perform two adaptive SSPRK2 steps, see Equations (3.6a–3.6b). In Figure 3.6, the first and the second adaptive SSPRK2 steps are represented by the straight and circle vectors, respectively. The order of the SSPRK2 steps is denoted by the index $(i), i = 1,2,3,...14$.
- If cells from different cell levels are neighbors, then we use the linear interpolation in time to match the time levels of such cells. See blue dash vectors in Figure 3.6 for the illustration of the interpolation.

Namely, at t^n, we apply the first SSPRK2 step (3.6a) on each level $l = 0,1,2$ to obtain the solution at time $t_l^{n,1}$, see the straight vectors (1) in Figure 3.6. One can see from Figure 3.6 that $t_2^{n,1} < t_1^{n,1} < t_0^{n,1}$ (see also Equation (3.5) for the sizes of local time steps). Hence, after the first time steps, we perform the second adaptive SSPRK2 step to update the solution at $t_2^{n,1}$ on level 2, see the circle vector (2) in Figure 3.6. Note that cells on level 2 may have some neighboring cells which are on the other levels $l = 0,1$. In that case, we need to approximate the solution at time $t_2^{n,1}$ in such neighboring cells on levels $l = 0,1$. The solutions at time $t_2^{n,1}$ on levels $l = 0,1$ can be computed by using the linear interpolation in time, see blue dash vectors in Figure 3.6. Next, we apply the first adaptive SSPRK2 step on level 2, see Equation (3.6a) and the straight vector (3) in Figure 3.6. To ensure the stability of the adaptive scheme, at time $t_2^{n,1}$, the local time-stepping size $\Delta t_2^{n,1}$ applied on level 2 is updated by using Equation (3.5). We continue to perform two adaptive SSPRK2 steps, see Equations (3.6a-3.6b), to obtain the solution at t^{n+1} on all levels.

3.4. A posteriori error estimator. Here, using the idea of Weak Local Residual (WLR) from [24, 45], we will derive local error estimator that is used as the robust indicator for the adaptive mesh refinement in our work.

Let us recall that the weak form of the mass conservation equation for the system (2.1) in $\Omega \times [0,T]$ takes the integral form,

$$\int_0^T \int_\Omega (w\phi_t(x,y,t) + hu\phi_x(x,y,t) + hv\phi_y(x,y,t)d\Omega dt + \int_\Omega w(x,y,0)\phi(x,y,0)d\Omega = 0, \tag{3.7}$$

for all sufficiently smooth test functions $\phi(x,y,t)$ with compact support on $\Omega \times [0,T)$.

Consider the example of a localized test function in time and space,

$$\phi_i^{n+\frac{1}{2}}(x,y,t) = \frac{1}{\Delta} f_i(x,y) f^{n+\frac{1}{2}}(t), \tag{3.8}$$

where $\Delta := \max(\max_{j,k}(r_{jk}), \Delta t)$, $\Delta t = t^{n+1} - t^n = t^{n+\frac{1}{2}} - t^{n-\frac{1}{2}}, n = 1,2,3...$, and r_{jk} are the heights of the triangle/cell $T_j^{n,\mathcal{M}_n} \in \mathcal{T}^{n,\mathcal{M}_n}$. Function $f^{n+\frac{1}{2}}(t)$ is a linear function in time with a local support defined as,

$$f^{n+\frac{1}{2}}(t) = \begin{cases} \dfrac{t - t^{n-\frac{1}{2}}}{\Delta t}, & \text{if} \quad t^{n-\frac{1}{2}} \le t \le t^{n+\frac{1}{2}}, \\ \dfrac{t^{n+\frac{3}{2}} - t}{\Delta t}, & \text{if} \quad t^{n+\frac{1}{2}} \le t \le t^{n+\frac{3}{2}}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.9}$$

Function $f_i(x,y), i = 1,2,3,...$ is a "hat function", namely, a piecewise linear function with compact support over all triangles with common vertex $N_i = (\widetilde{x}_i, \widetilde{y}_i)$. The function $f_i(x,y)$ takes value 1 at the vertex N_i and 0 at all other nodes. More precisely, assume that there are C_i triangles $T_{j_1}^{n,\mathcal{M}_n}$, $T_{j_2}^{n,\mathcal{M}_n}$, $T_{j_3}^{n,\mathcal{M}_n}$,..., $T_{j_{C_i}}^{n,\mathcal{M}_n} \in \mathcal{T}^{n,\mathcal{M}_n}$ which share a common vertex N_i. Thus, the function $f_i(x,y)$ is defined as,

$$f_i(x,y) = \begin{cases} a_c^{(i)}(x - \widetilde{x}_i) + b_c^{(i)}(y - \widetilde{y}_i) + 1, & \text{if} \quad (x,y) \in T_{j_c}^{n,\mathcal{M}_n}, \quad c = 1,2,...C_i \\ 0, & \text{otherwise} \end{cases}, \tag{3.10}$$

The quantity $(a_c^{(i)}, b_c^{(i)})$ is the gradient of the linear piece of $f_i(x,y)$ restricted to $T_{j_c}^{n,\mathcal{M}_n}$,

$$\begin{aligned} a_c^{(i)} &= \frac{\widetilde{y}_2 - \widetilde{y}_3}{(\widetilde{y}_3 - \widetilde{y}_i)(\widetilde{x}_2 - \widetilde{x}_i) - (\widetilde{y}_2 - \widetilde{y}_i)(\widetilde{x}_3 - \widetilde{x}_i)}, \\ b_c^{(i)} &= \frac{\widetilde{x}_3 - \widetilde{x}_2}{(\widetilde{y}_3 - \widetilde{y}_i)(\widetilde{x}_2 - \widetilde{x}_i) - (\widetilde{y}_2 - \widetilde{y}_i)(\widetilde{x}_3 - \widetilde{x}_i)}, \end{aligned} \tag{3.11}$$

where $N_i = (\widetilde{x}_i, \widetilde{y}_i), (\widetilde{x}_2, \widetilde{y}_2)$, and $(\widetilde{x}_3, \widetilde{y}_3)$ are the three vertices of triangle $T_{j_c}^{n,\mathcal{M}_n}$. Next, define the following piecewise constant approximation for the solution $\boldsymbol{U} = (w, hu, hv)$,

$$\boldsymbol{U}^{\Delta} := \overline{\boldsymbol{U}}_{j_c}^{n}, \text{ if } (x,y,t) \in T_{j_c}^{n,\mathcal{M}_n} \times [t^{n-\frac{1}{2}}, t^{n+\frac{1}{2}}]. \tag{3.12}$$

Now, using the localized test function $\phi_i^{n+\frac{1}{2}}(x,y,t)$, (3.8) together with the piecewise constant approximation $\boldsymbol{U}^{\Delta}$, (3.12) in (3.7), we can define *the weak form of the truncation error, (WLR)*, which will be used as the error indicator for refinement/derefinement in the adaptive grid,

$$\begin{aligned} E_i^{n+\frac{1}{2}} &:= E(\boldsymbol{U}^{\Delta}, \phi_i^{n+\frac{1}{2}}) \\ &= \sum_{c=1}^{C_i} \int_{t^{n-\frac{1}{2}}}^{t^{n+\frac{1}{2}}} \int_{T_{j_c}^{n,\mathcal{M}_n}} (\overline{w}_{j_c}^{n}(\phi_i^{n+\frac{1}{2}})_t + (\overline{hu})_{j_c}^{n}(\phi_i^{n+\frac{1}{2}})_x + (\overline{hv})_{j_c}^{n}(\phi_i^{n+\frac{1}{2}})_y) d\Omega dt \\ &\quad + \sum_{c=1}^{C_i} \int_{t^{n+\frac{1}{2}}}^{t^{n+\frac{3}{2}}} \int_{T_{j_c}^{n,\mathcal{M}_n}} (\overline{w}_{j_c}^{n+1}(\phi_i^{n+\frac{1}{2}})_t + (\overline{hu})_{j_c}^{n+1}(\phi_i^{n+\frac{1}{2}})_x + (\overline{hv})_{j_c}^{n+1}(\phi_i^{n+\frac{1}{2}})_y) d\Omega dt. \end{aligned} \tag{3.13}$$

After straightforward calculations, the WLR error $E_i^{n+\frac{1}{2}}$ on mesh $\mathcal{T}^{n,\mathcal{M}_n}$ is given by the formula,

$$E_i^{n+\frac{1}{2}} = \frac{1}{\Delta}(\mathcal{U}_i^{n+\frac{1}{2}} + \mathcal{F}_i^{n+\frac{1}{2}} + \mathcal{G}_i^{n+\frac{1}{2}}),$$

$$\mathcal{U}_i^{n+\frac{1}{2}} = \sum_{c=1}^{C_i} \frac{1}{3} |T_{j_c}^{n,\mathcal{M}_n}| (\overline{w}_{j_c}^n - \overline{w}_{j_c}^{n+1}),$$

$$\mathcal{F}_i^{n+\frac{1}{2}} = \sum_{c=1}^{C_i} a_c^{(i)} \frac{\Delta t}{2} |T_{j_c}^{n,\mathcal{M}_n}| ((\overline{hu})_{j_c}^n + (\overline{hu})_{j_c}^{n+1}),$$

$$\mathcal{G}_i^{n+\frac{1}{2}} = \sum_{c=1}^{C_i} b_c^{(i)} \frac{\Delta t}{2} |T_{j_c}^{n,\mathcal{M}_n}| ((\overline{hv})_{j_c}^n + (\overline{hv})_{j_c}^{n+1}). \tag{3.14}$$

The error in a cell $T_j^{n,\mathcal{M}_n} \in \mathcal{T}^{n.\mathcal{M}_n}$ is given by,

$$e_j = \max_k \left| E_{jk}^{n+\frac{1}{2}} \right|, \quad k = 1,2,3, \tag{3.15}$$

where $E_{jk}^{n+\frac{1}{2}}$ is the WLR error computed in (3.14) at a node k of triangle T_j.

In our numerical experiments, we define an error tolerance, ω as,

$$\omega = \sigma \max_j (e_j), \tag{3.16}$$

where $\sigma < 1$ is a given problem-dependent constant (see Section 4), and e_j is the WLR error in the triangle $T_j^{n.\mathcal{M}_n}$, (3.15). The error e_j in each cell $T_j^{n.\mathcal{M}_n} \in \mathcal{T}^{n,\mathcal{M}_n}$ is compared to the error tolerance (3.16), and the cell is either "flagged" for refinement/de-refinement or "no-change".

Next, we will determine the number of refinement levels for each triangle T_j by comparing the WLR error e_j in T_j with the threshold ω. Note that the adaptive central-upwind method has a second-order accuracy in space for a smooth solution, numerically shown in Section 4.1. This means that the error of the solution obtained in a grid decreases by a factor of four if we refine each cell in that grid into four smaller cells by using the midpoints, see Figure 3.1 for the illustration of the refinement. Therefore, in order to obtain the error in a cell T_j less than the threshold ω, we refine T_j for $m = 1,2,3,...$ levels if the local error e_j in cell T_j satisfies $\omega \times 4^{m-1} \le e_j < \omega \times 4^m$.

Note that, in this work, we consider only the Equation (1.1a) to obtain WLR error. The full system of shallow water equations can be used to derive WLR too, however it will make the computation of the error indicator more complex and more expensive.

4. Numerical examples

In this section, we illustrate the performance of the designed adaptive central-upwind scheme. We compare the results of the adaptive central-upwind scheme developed in this work with the results of the central-upwind scheme from [36] on uniform triangular meshes (an example of such a uniform triangular mesh is outlined in Figure 4.1). In addition, in all experiments, we compute ratio, $\mathcal{R}_{CPU} = \frac{CPU_{uniform}}{CPU_{adaptive}}$, which is the ratio of the CPU times of the central-upwind algorithm without adaptivity to the CPU time of the adaptive central-upwind algorithm. To compare L^1-errors, as well as to compare the CPU times and to compute $\mathcal{R}_{CPU}$, we consider uniform mesh and the adaptive mesh with the same size of the smallest cells. More precisely, in Table 4.1, L^1-errors, and in Tables 4.2-4.7, $\mathcal{R}_{CPU}$ are computed using the uniform meshes $2 \times N \times N, N = 100,200,400$ and using the corresponding adaptive meshes which are obtained from the coarser uniform mesh $2 \times N/2^{\mathcal{M}} \times N/2^{\mathcal{M}}$ ($\mathcal{M} = 1,2$ is the highest level of refinement in the adaptive mesh). In Example 4.1 and the first two cases in Example 4.2 with (4.1-4.2), the gravitational acceleration is set $g = 1.0$, while in the last case of

Example 4.2 with (4.3-4.4) and Example 4.4, we take $g=9.8$. We set the desingularization parameters τ and ε for calculations of the velocity components u and v to be $\tau=\max_j\{|T_j^{n,\mathcal{M}_n}|^2\}$ and $\varepsilon=10^{-4}$, except for the Example 4.2, (4.3-4.4), where $\varepsilon=10^{-2}$ (see Section 2.1 formula (2.6) in [36]).

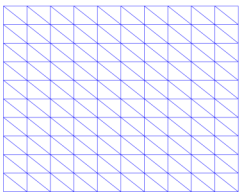


FIG. 4.1. *An outline of uniform triangular mesh.*

4.1. Example 4.1—accuracy test. Here, we consider the example from [6], and we verify experimentally the order of accuracy of the designed adaptive central-upwind scheme. We also compare the computational efficiency of the adaptive central-upwind scheme and the central-upwind scheme without adaptivity on uniform and non-uniform triangular meshes.

We consider the following initial data and the bottom topography,

$$w(x,y,0)=1,\quad u(x,y,0)=0.3,\quad v(x,y,0)=0,$$

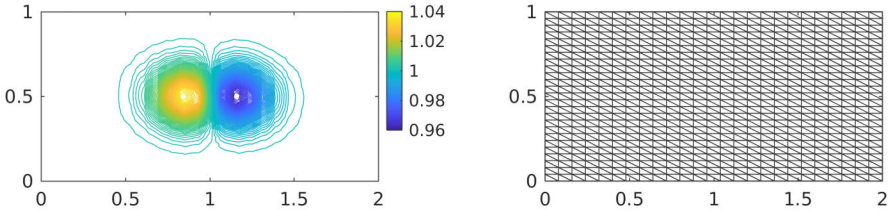
$$B(x,y)=0.5\exp(-25(x-1)^2-50(y-0.5)^2).$$

The computational domain is $[0,2]\times[0,1]$. A zero-order extrapolation is used at all boundaries. The error tolerance (3.16) for the mesh refinement in this example is set to $\omega=0.01\max_j(e_j)$.

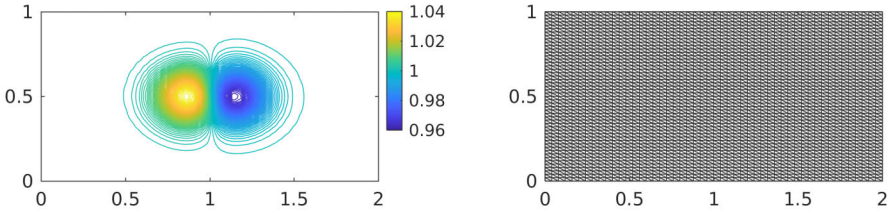
From the result reported in [6], by $t=0.07$, the numerical solution reaches the steady state. In Figure 4.2 (left column), we show the numerical solution of water surface at $t=0.07$. The solution is computed using the central-upwind scheme on uniform meshes on Figure 4.2 (a, b) and using the adaptive central-upwind scheme on Figure 4.2 (c, d). The adaptive meshes in Figure 4.2 are obtained from the uniform mesh $2\times25\times25$, Figure 4.2 (a). The adaptive mesh with one level of refinement $\mathcal{M}=1$ (as the highest level of refinement) is on Figure 4.2 (c), and with two levels of refinement $\mathcal{M}=2$ (as the highest level of refinement) is on Figure 4.2 (d).

Next, in Table 4.1 we compute the L^1-errors of the central-upwind scheme on uniform meshes and of the adaptive central-upwind scheme. To obtain the errors, the reference solution is calculated on the uniform mesh with $2\times1600\times1600$ triangles. In Table 4.2, we present the $\mathcal{R}_{CPU}$ ratio to compare the computational efficiency of the two methods. From Table 4.1 and Table 4.2, we observe that the adaptive algorithm produces similar accuracy as the scheme on fixed uniform triangular meshes, but at a less computational cost. Also, as expected, the adaptive central-upwind scheme achieves second-order accuracy in space, similar to the central-upwind scheme without adaptivity.

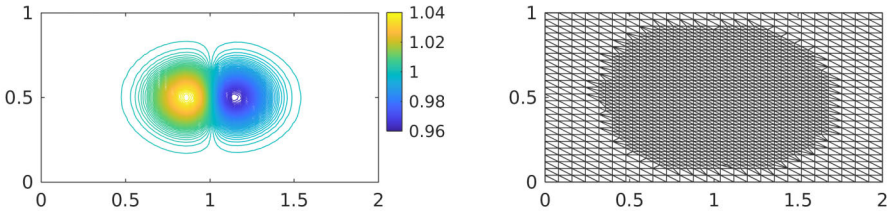
In addition, we will use this example to show that the adaptive central-upwind scheme is also effective on the unstructured triangular meshes. On Figure 4.3 (left), we plot the numerical solution at $t=0.07$ computed using the central-upwind method on



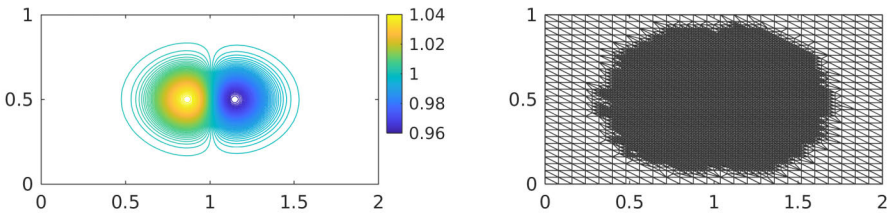
(a) Uniform mesh $2\times25\times25$.



(b) Uniform mesh $2\times50\times50$.



(c) Adaptive mesh with $\mathcal{M}=1$.



(d) Adaptive mesh with $\mathcal{M}=2$.

FIG. 4.2. *Example 4.1: Computational water surface* $w(x,y,0.07)$ *(left column) with the corresponding meshes (right column).*

a non-uniform mesh, Figure 4.3 (a), the adaptive scheme with one level of refinement, Figure 4.3 (b) and the adaptive scheme with two levels of refinement, Figure 4.3 (c). The adaptive meshes on Figure 4.3 (b, c) are obtained at $t=0.07$ as a result of applying the adaptive central-upwind scheme on the non-uniform mesh shown on the right of Figure 4.3 (a).

We then recompute the accuracy of the solution, see Table 4.3, and the CPU time ratio, see Table 4.4, obtained by using the non-uniform meshes. From Table 4.3 and

algorithm without adaptivity			adaptive algorithm					
			one level $\mathcal{M}=1$			two levels $\mathcal{M}=2$		
cells	L^1-error	rate	cells	L^1-error	rate	cells	L^1-error	rate
20000	2.23e-05		11,808	2.23e-05		10,404	3.43e-05	
80000	4.89e-06	2.19	46,892	5.76e-06	1.95	40,728	6.07e-06	2.50
320000	1.11e-06	2.14	187,726	1.67e-06	1.79	160,846	1.67e-06	1.86

TABLE 4.1. *Example 4.1: L^1-errors of the water surface w at $t=0.07$, and the convergence rates of the central-upwind scheme without adaptivity (uniform mesh $2\times N\times N, N=100,200,400$) and the adaptive scheme (the corresponding adaptive mesh is reconstructed from the uniform mesh $2\times N/2^{\mathcal{M}}\times N/2^{\mathcal{M}}$).*

uniform mesh (cells)	adaptive mesh $\mathcal{M}=1$ (cells)	$\mathcal{R}_{CPU}$ with $\mathcal{M}=1$		adaptive mesh $\mathcal{M}=2$ (cells)	$\mathcal{R}_{CPU}$ with $\mathcal{M}=2$	
		total	without grid generation		total	without grid generation
$2\times100\times100$	11,808	2.18	2.36	10,404	2.58	2.70
$2\times200\times200$	46,892	1.85	2.05	40,728	2.48	2.62
$2\times400\times400$	187,726	1.77	1.98	160,846	2.25	2.40
$\mathcal{R}_{CPU}$ average:		1.93	2.13		2.44	2.57

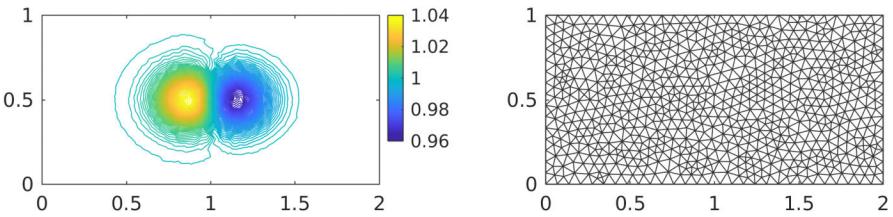
TABLE 4.2. *Example 4.1: $\mathcal{R}_{CPU}$ ratio at $t=0.07$, where for adaptive central-upwind scheme, we consider the total CPU times and CPU times without the grid generation.*

algorithm without adaptivity			adaptive algorithm					
			one level $\mathcal{M}=1$			two levels $\mathcal{M}=2$		
cells	L^1-error	rate	cells	L^1-error	rate	cells	L^1-error	rate
22,438	2.65e-05		13,036	2.54e-05		11,630	3.60e-05	
90,434	6.61e-06	2.00	51,656	6.07e-06	2.07	44,400	7.16e-06	2.33
314,708	1.24e-06	2.41	204,044	1.77e-06	1.78	176,278	1.68e-06	2.09

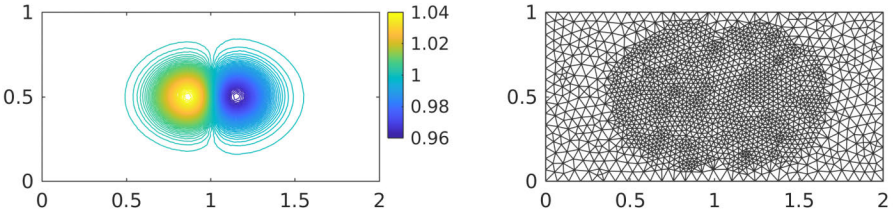
TABLE 4.3. *Example 4.1: L^1-errors of the water surface w at $t=0.07$, and the convergence rates of the central-upwind scheme without adaptivity in non-uniform mesh and the adaptive scheme (the corresponding adaptive meshes have the same size of the smallest cells with the compared non-uniform meshes).*

Table 4.4, we observe that the advantage of the adaptive scheme is still maintained on non-uniform meshes as it reduces the computational cost of the method. Next, we compare the results presented in Table 4.3 and in Table 4.1. As expected, we see that the errors obtained by using the scheme on non-uniform meshes are slightly larger than the ones using the corresponding uniform meshes (with approximately the same number of cells).

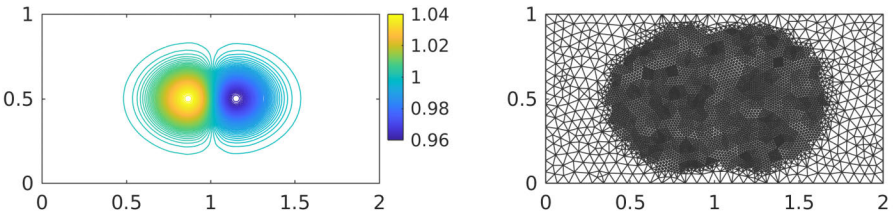
4.2. Example 4.2—well-balanced tests. The first numerical example here was proposed in [33] to test the capability of the numerical scheme to accurately resolve small perturbations of a steady state solution. We take a computational domain $[0,2]\times[0,1]$



(a) Non-uniform mesh with 1824 cells.



(b) Adaptive mesh with $\mathcal{M}=1$.



(c) Adaptive mesh with $\mathcal{M}=2$.

FIG. 4.3. *Example 4.1: Computational water surface $w(x,y,0.07)$ (left column) with the corresponding meshes (right column) obtained by applying the adaptive central-upwind scheme on a non-uniform mesh.*

and the bottom topography,

$$B(x,y)=0.8\exp(-5(x-0.9)^2-50(y-0.5)^2). \tag{4.1}$$

The initial conditions describe a flat surface of water with a small perturbation in $0.05<x<0.15$:

$$w(x,y,0)=\begin{cases}1+\epsilon, & 0.05<x<0.15,\\ 1, & \text{otherwise},\end{cases} \qquad u(x,y,0)\equiv v(x,y,0)\equiv 0, \tag{4.2}$$

where ϵ is the perturbation height. We have used zero-order extrapolation at the right and the left boundaries of the domain and periodic boundary conditions for the top and the bottom ones.

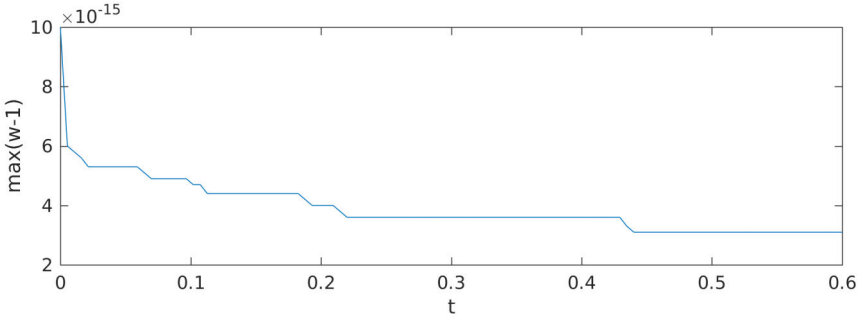
Non-uniform and non-adaptive mesh (cells)	adaptive mesh $\mathcal{M}=1$ (cells)	$\mathcal{R}_{CPU}$ with $\mathcal{M}=1$ total	$\mathcal{R}_{CPU}$ with $\mathcal{M}=1$ without grid generation	adaptive mesh $\mathcal{M}=2$ (cells)	$\mathcal{R}_{CPU}$ with $\mathcal{M}=2$ total	$\mathcal{R}_{CPU}$ with $\mathcal{M}=2$ without grid generation
22,438	13,036	1.86	1.94	11,630	3.77	3.90
90,434	51,656	2.20	2.20	44,400	3.34	3.43
314,708	204,044	2.45	2.57	176,278	3.36	3.46
$\mathcal{R}_{CPU}$ average:		2.17	2.24		3.49	3.60

TABLE 4.4. *Example 4.1: $\mathcal{R}_{CPU}$ ratio at $t=0.07$, where for adaptive central-upwind scheme, we consider the total CPU times and CPU times without the grid generation in non-uniform meshes.*

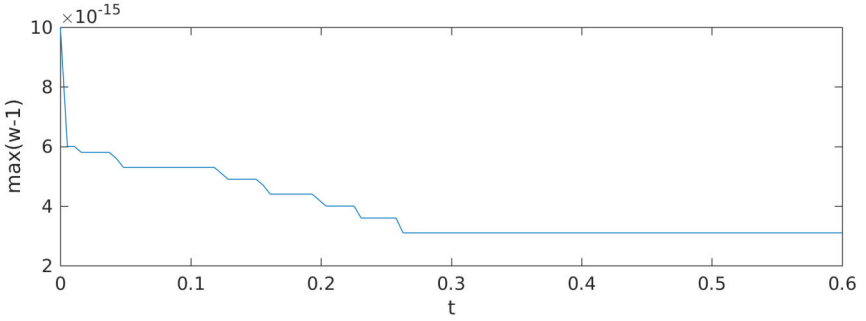
To verify the well-balanced property of the adaptive scheme, we first consider a very small perturbation $\epsilon=10^{-14}$. The adaptive meshes with levels $\mathcal{M}=1,2$ are obtained from a coarse uniform mesh $2\times25\times25$. The threshold for mesh refinement in this example is $\omega=0.1\max_j(e_j)$. We plot $\max_{x,y}(w-1)$ as a function of time for the central-upwind scheme without adaptivity on a uniform mesh in Figure 4.4 (a), for the adaptive central-upwind scheme with $\mathcal{M}=1$ in Figure 4.4 (b), and with $\mathcal{M}=2$ in Figure 4.4 (c). The results of the test show that the adaptive scheme is stable and preserves numerically the balance between the fluxes and the source term, similar to the scheme without adaptivity.

In the next numerical test, we take a larger perturbation value $\epsilon=10^{-2}$. The purpose of this test is to demonstrate the capability of the adaptive algorithm to resolve the small scale features of the solution. In Figure 4.5, we plot the water surface w obtained by the method without adaptivity on two uniform meshes $2\times100\times100$ (left) and $2\times200\times200$ (right) at $t=0.6,0.9,1.2,1.5$, and 1.8. The computed solutions of the water surface exhibit a right-going disturbance propagating past the hump. We then apply the adaptive algorithm and plot the numerical solution of the water surface w on selected meshes at different times in Figure 4.6 (left) with $\mathcal{M}=1$ and in Figure 4.7 (left) with $\mathcal{M}=2$ (the starting grid was a uniform mesh with $2\times100\times100$ and the threshold is $\omega=0.1\max_j(e_j)$). We observe from Figure 4.5, Figure 4.6 and Figure 4.7 that the adaptive central-upwind scheme delivers high resolution of the complex features of a small perturbation of the "lake-at-rest" steady state. We note also, that by increasing the level of refinement from $\mathcal{M}=1$ to $\mathcal{M}=2$, the number of cells in the mesh increases from 30,912 cells with $\mathcal{M}=1$ to 66,097 cells with $\mathcal{M}=2$, but the accuracy is clearly improved with higher resolution as seen in Figure 4.6 and Figure 4.7. We also present the corresponding adaptive meshes in Figure 4.6 (right) and in Figure 4.7 (right). Clearly, the meshes are adapted to the behavior of the flow during time evolution from $t=0.6$ to $t=1.8$. This confirms the ability of the WLR error estimator to detect the location of the steep local gradient in the solution.

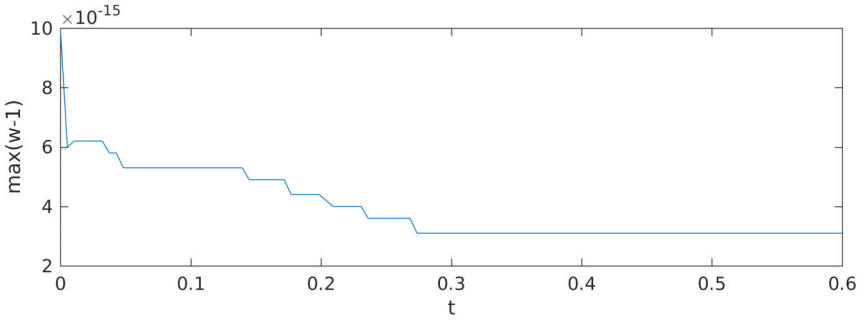
To demonstrate further the advantage of the adaptive scheme, we have compared the CPU times of the central-upwind scheme without adaptivity and with adaptivity for the solution at $t=0.9$. The computed $\mathcal{R}_{CPU}$ ratios are presented in Table 4.5. The computed ratios $\mathcal{R}_{CPU}$ show that the adaptive central-upwind scheme can reduce the



(a) Central-upwind scheme on a uniform mesh $2\times25\times25$.



(b) Adaptive central-upwind scheme on a mesh obtained from the uniform mesh $2\times25\times25$ with $\mathcal{M}=1$.



(c) Adaptive central-upwind scheme on a mesh obtained from the uniform mesh $2\times25\times25$ with $\mathcal{M}=2$.

FIG. 4.4. *Example 4.2:* $\max_{x,y}(w(x,y,t)-1)$ *is plotted as a function of* t *on the uniform grid and the adaptive grids.*

numerical cost by about three times with $\mathcal{M}=1$ and by about five times with $\mathcal{M}=2$ in comparison with the cost of the algorithm without adaptivity.

4.3. Example 3—well-balanced test with wet/dry interfaces. In this test, we consider a small perturbation of the lake-at-rest steady-state that propagates around an island. Similar examples were considered in [6, 36]. We consider a hump partially submerged in water so that there is a disk-shaped island at the origin, see Figure 4.8.

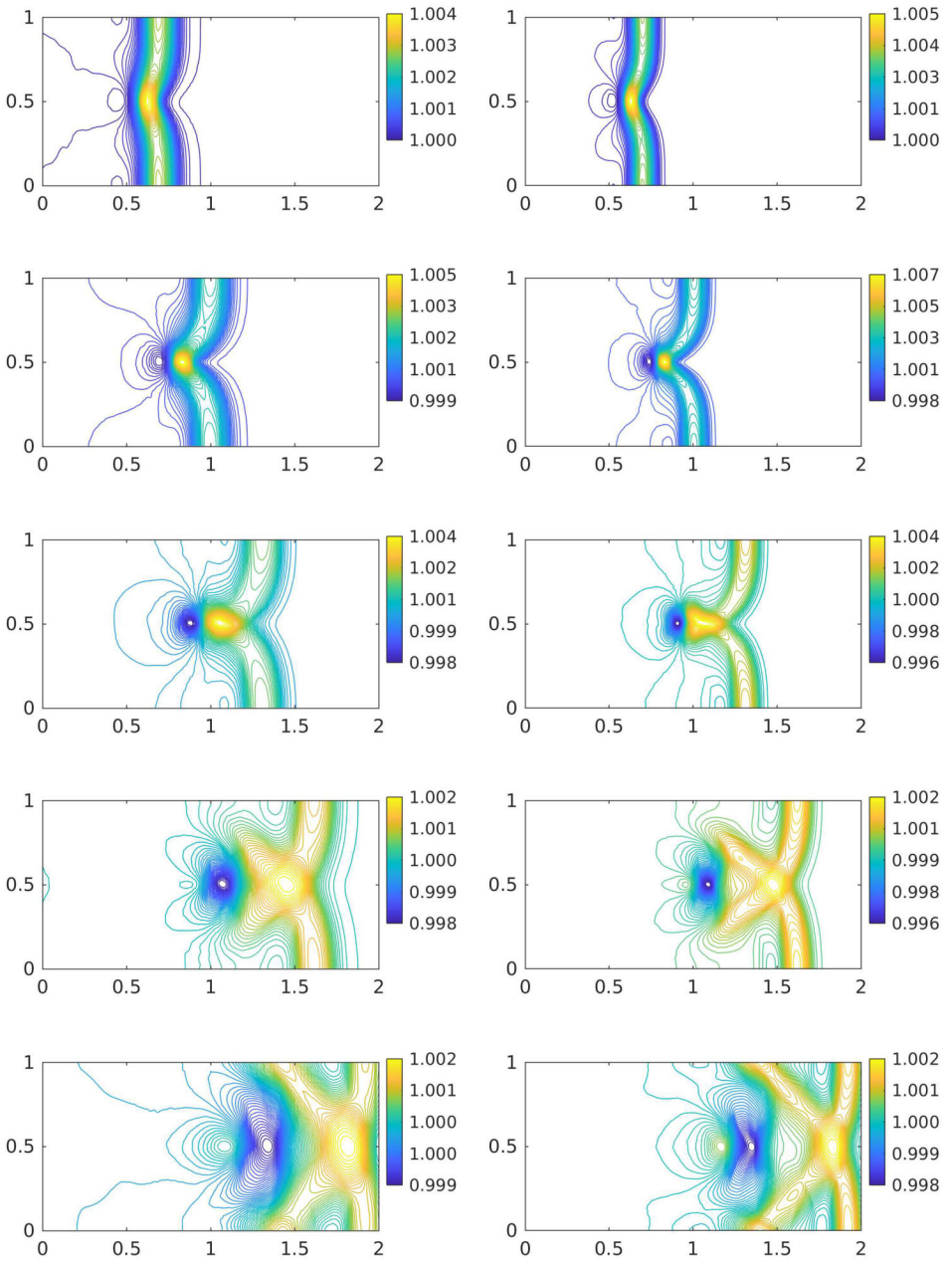


FIG. 4.5. *Example 4.2: w component of the solution of the IVP (1.1a-1.1c), (4.1-4.2) with* $\epsilon = 10^{-2}$ *at* $t = 0.6, 0.9, 1.2, 1.5,$ *and* 1.8 *(from top to bottom) obtained by the central-upwind scheme without adaptivity on uniform meshes* $2 \times 100 \times 100$ *(left column) and* $2 \times 200 \times 200$ *(right column).*

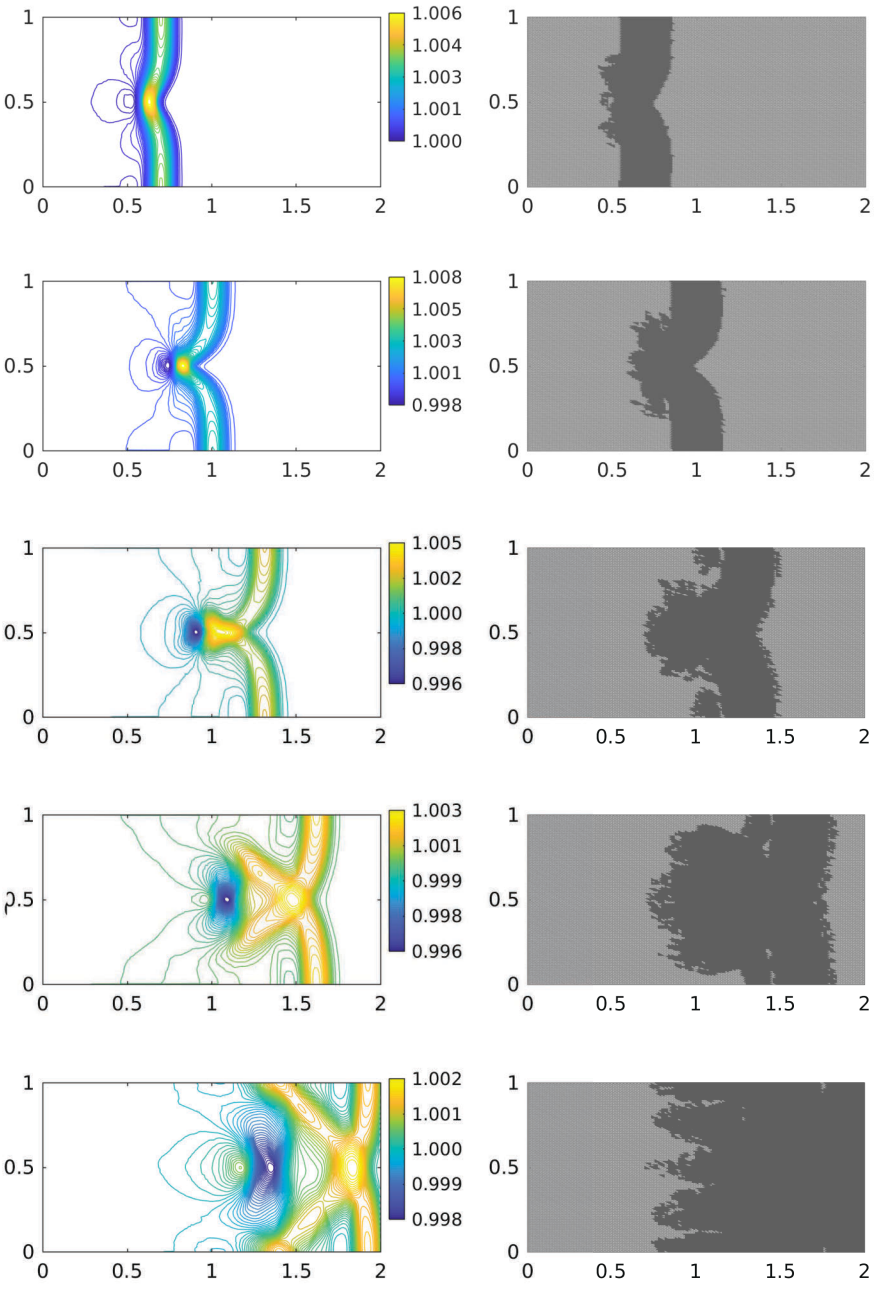


FIG. 4.6. *Example 4.2: w component of the solution of the IVP (1.1a-1.1c), (4.1-4.2) with* $\epsilon = 10^{-2}$ *(left column) at* $t = 0.6, 0.9, 1.2, 1.5,$ *and* 1.8 *(from top to bottom) obtained by the adaptive central-upwind scheme. The corresponding adaptive meshes have one level of refinement* $\mathcal{M} = 1$ *(right column).*

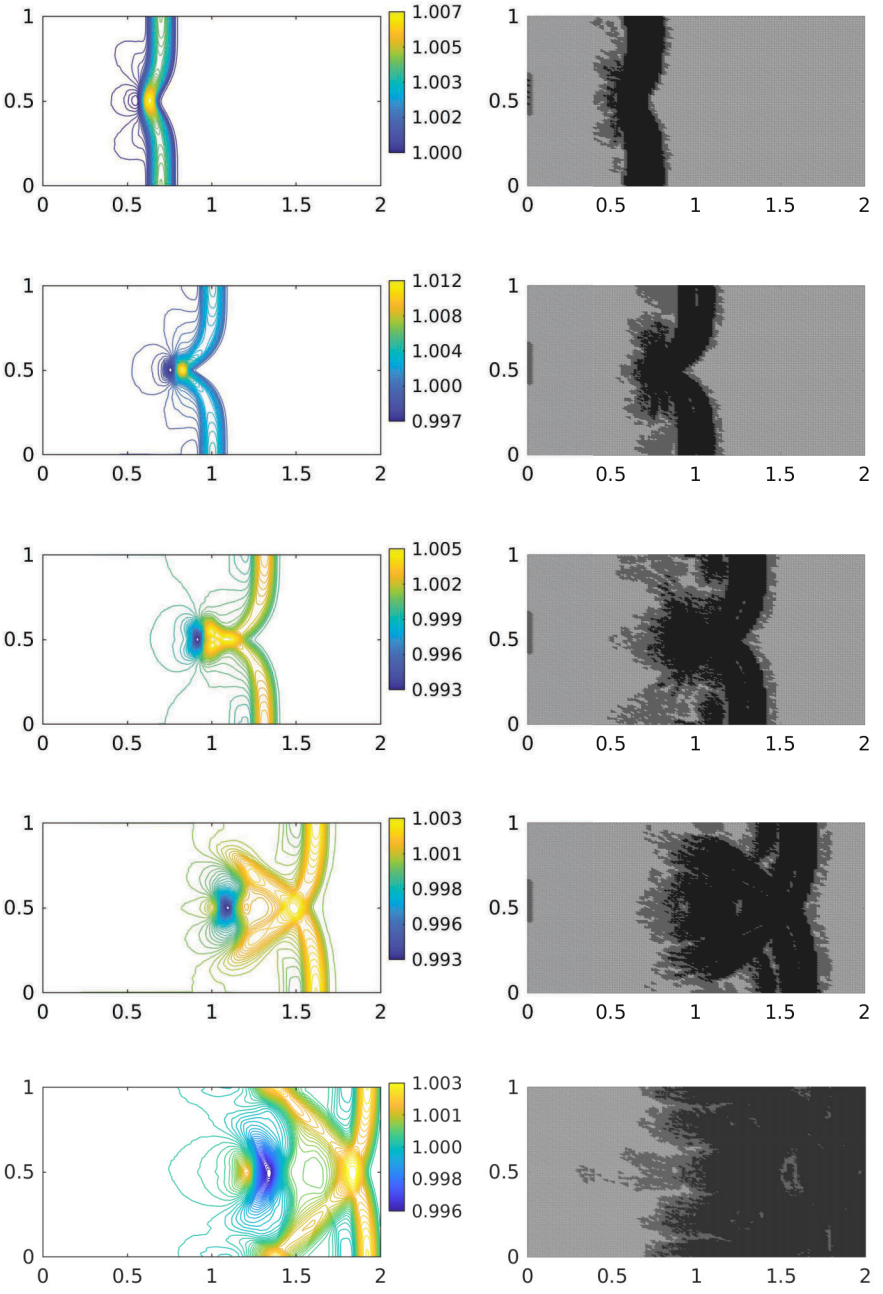


FIG. 4.7. *Example 4.2: w component of the solution of the IVP (1.1a-1.1c), (4.1-4.2) with* $\epsilon = 10^{-2}$ *(left column) at* $t = 0.6, 0.9, 1.2, 1.5,$ *and* 1.8 *(from top to bottom) obtained by the adaptive central-upwind scheme. The corresponding adaptive meshes have two levels of refinement* $\mathcal{M} = 2$ *(right column).*

uniform mesh (cells)	adaptive mesh $\mathcal{M}=1$ (cells)	$\mathcal{R}_{CPU}$ with $\mathcal{M}=1$	adaptive mesh $\mathcal{M}=2$ (cells)	$\mathcal{R}_{CPU}$ with $\mathcal{M}=2$
$2\times100\times100$	9,127	3.50	8,274	4.06
$2\times200\times200$	30,912	2.06	21,131	5.29
$2\times400\times400$	108,297	3.67	66,097	6.38
$\mathcal{R}_{CPU}$ average:		3.07		5.24

TABLE 4.5. *Example 4.2: The R_{CPU} ratio obtained in solving the IVP (1.1a-1.1c), (4.1-4.2) with $\epsilon=10^{-2}$ at $t=0.9$.*

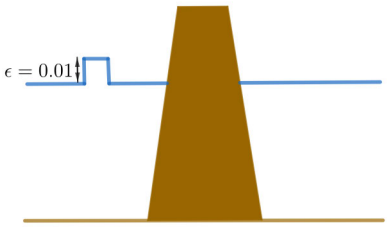


FIG. 4.8. *Example 3: 1-D slice of the bottom topography (brown) and water surface (blue) at $t=0$. The plot is not to scale.*

Hence, the bottom topography is given by

$$B(x,y)=\begin{cases}1.1, & r\le 0.1,\\ 11\times(0.2-r) & 0.1<r\le 0.2,\\ 0, & \text{otherwise},\end{cases}\qquad r:=\sqrt{(x-0.5)^2+(y-0.5)^2}. \tag{4.3}$$

We consider the following initial condition,

$$w(x,y,0)=\begin{cases}1+\epsilon, & 0.1<x<0.2,\\ \max(1,B(x,y)), & \text{otherwise},\end{cases}\qquad u(x,y,0)\equiv v(x,y,0)\equiv 0, \tag{4.4}$$

where $\epsilon=10^{-2}$ is the perturbation height. Homogeneous Neumann boundary conditions are used at all boundaries.

We first obtain the water surface using central-upwind scheme without adaptivity. Different to the results in the submerged plateau case (4.1-4.2), the right-going disturbance bends around the island and is reflected by the island.

We then compare with the performance of the adaptive scheme in this case. The adaptive grids are generated from the uniform mesh $2\times100\times100$ using the threshold $\omega=0.001\max_j(e_j)$ for one level of refinement $\mathcal{M}=1$ and $\omega=0.01\max_j(e_j)$ for $\mathcal{M}=2$. In Figure 4.10 (left) and Figure 4.11 (left), we plot the results for w (left) obtained by the adaptive scheme and the corresponding adaptive meshes (right). The results are similar to the results of the central-upwind scheme without adaptivity in Figure 4.9. We do not observe any non-physical spurious waves generated at the wet/dry front. Note that, as shown in [36], if the scheme is not well-balanced, the numerical results will be polluted by numerical oscillations. Moreover, the computed water depth is nonnegative in the

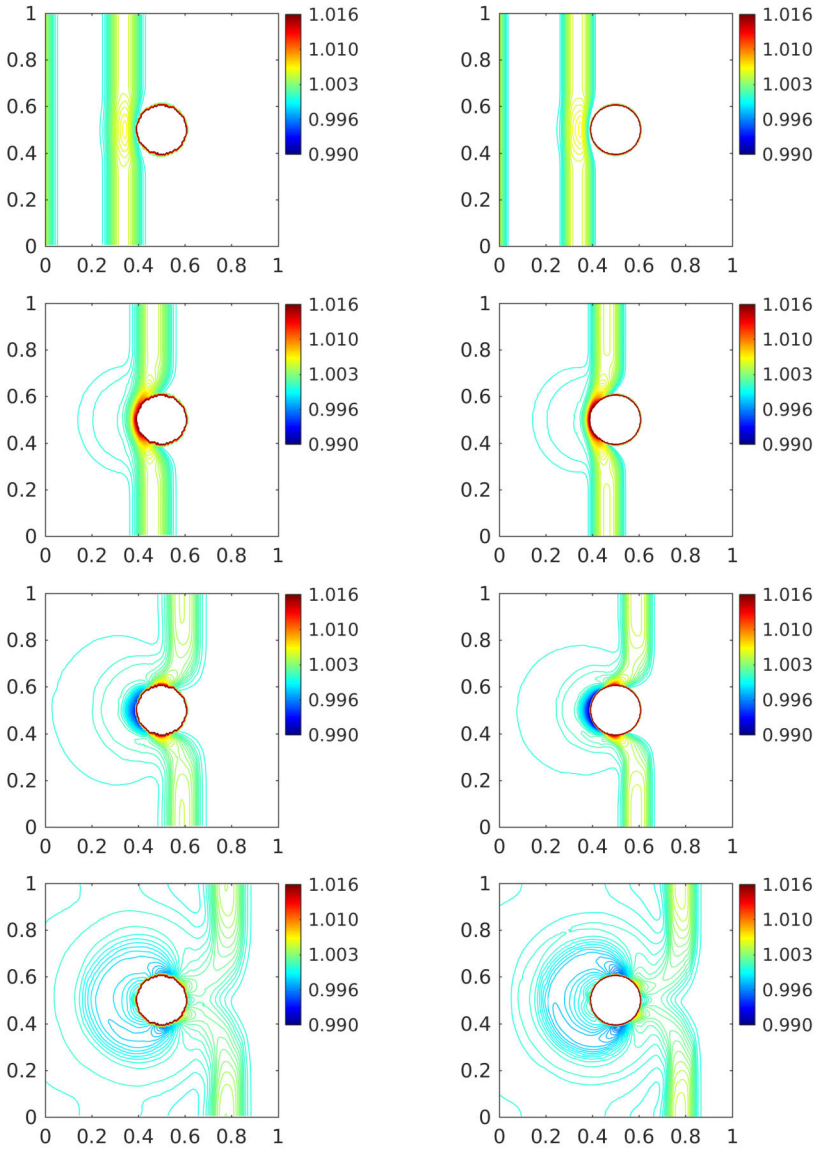


FIG. 4.9. *Example 3: w component of the solution of the IVP (1.1a-1.1c), (4.3-4.4) at $t=$ 0.06,0.1,0.14, and 0.2 (from top to bottom) obtained by the central-upwind scheme without adaptivity on uniform meshes $2\times100\times100$ (left column) and $2\times200\times200$ (right column).*

adaptive meshes. This means that the adaptive method maintains the well-balanced and positivity-preserving properties of the central-upwind scheme originally proposed in [36]. As in previous examples, the WLR error estimator accurately detects regions in the domain which are marked for adaptive refinement/coarsening.

Next, in Table 4.6, we present ratio of CPU times $\mathcal{R}_{CPU}$ to compute the solution at $t=0.1$ by the central-upwind method without adaptivity and by the adaptive algorithm. In order to compare the computational costs and calculate $\mathcal{R}_{CPU}$, we consider uniform

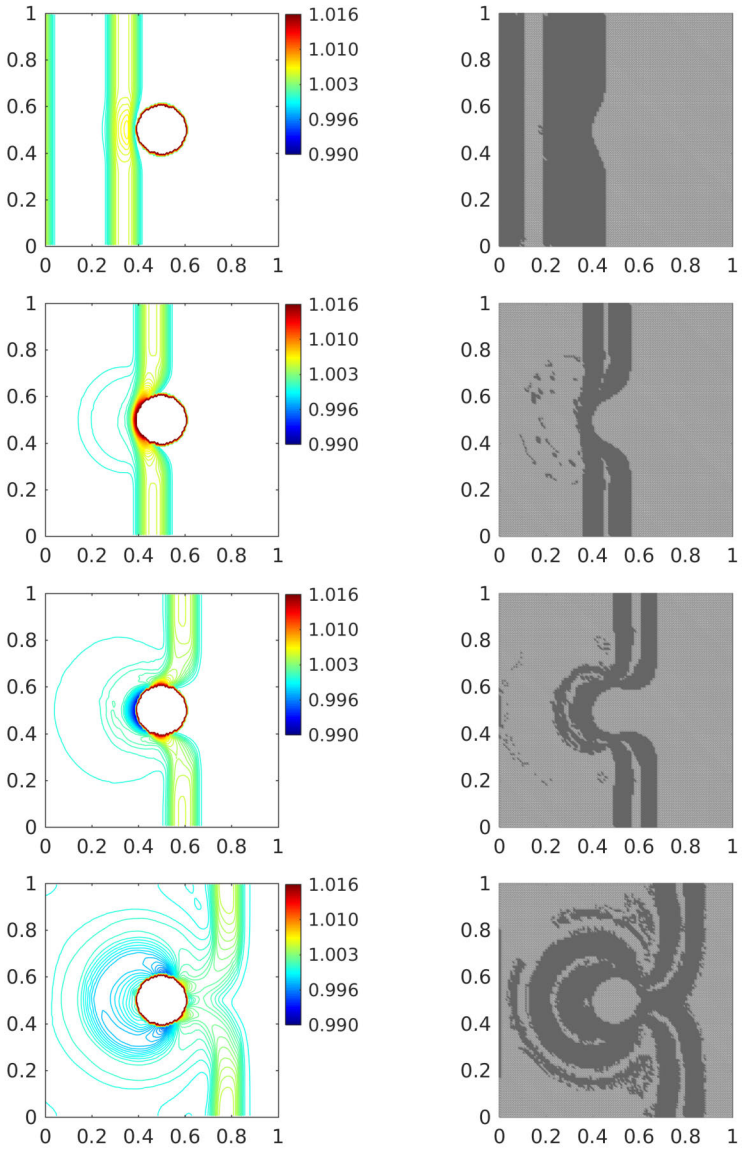


FIG. 4.10. *Example 3: w component of the solution of the IVP (1.1a-1.1c), (4.3-4.4) at $t=0.06,0.1,0.14,$ and 0.2 (from top to bottom) obtained by the adaptive central-upwind scheme (left column) and the corresponding adaptive meshes with one level of refinement $\mathcal{M}=1$ (right column).*

and adaptive meshes with the same size of the smallest cells. From Table 4.6, one can see that the adaptive algorithm reduces the CPU times up to four times. In our experiments, we considered only $\mathcal{M}=1$ and $\mathcal{M}=2$, but one can consider higher levels of refinement to further enhance the accuracy of the numerical solution at the reduced computational cost.

Finally, we illustrate the advantages of WLR error as the error indicator, and hence compare it with another example of the error indicator which uses the unlimited gradient

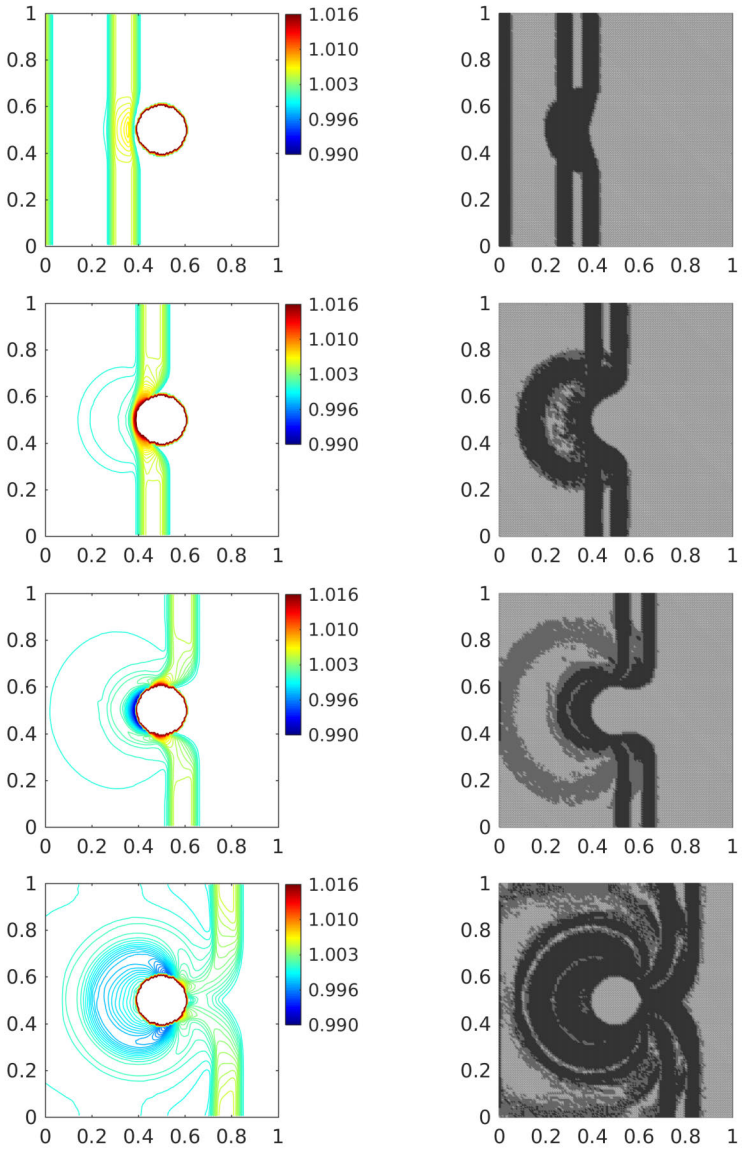


FIG. 4.11. *Example 3: w component of the solution of the IVP (1.1a-1.1c), (4.3-4.4) at $t=0.06,0.1,0.14,$ and 0.2 (from top to bottom) obtained by the adaptive central-upwind scheme (left column) and the corresponding adaptive meshes with two levels of refinement $\mathcal{M}=2$ (right column).*

of the water surface (w_x, w_y), see e.g. [19]. As shown in [19], the gradient indicator is simple and applicable. However, the gradient indicator is very general since it does not take into account more subtle features of a particular model, namely the shallow water model in this work. Meanwhile, in this research, we designed the WLR error specifically based on the SWEs, see Section 3.4. Therefore, the WLR error indicator adapts to the shallow water model under study and accurately captures the behavior of the solutions. Moreover, as the gradient indicator does not take into account the scale of

uniform mesh (cells)	adaptive mesh $\mathcal{M}=1$ (cells)	$\mathcal{R}_{CPU}$ with $\mathcal{M}=1$	adaptive mesh $\mathcal{M}=2$ (cells)	$\mathcal{R}_{CPU}$ with $\mathcal{M}=2$
$2\times100\times100$	11,831	1.91	6,155	3.04
$2\times200\times200$	31,050	2.08	25,753	3.14
$2\times400\times400$	154,616	3.16	94,357	5.82
$\mathcal{R}_{CPU}$ average:		2.38		4.00

TABLE 4.6. *Example 3: The R_{CPU} ratio for solving the IVP (1.1a-1.1c), (4.3-4.4) at $t=0.1$.*

a mesh cell, it does not provide necessary information about the number of refinement levels needed in that cell. Finally, we will illustrate the performance of the gradient and WLR error as the error indicators to obtain the adaptive meshes. We continue to consider the IVP (1.1a-1.1c), (4.1-4.2). In Figure 4.12 (left column), we first present the contour plots of the water surface w computed at $t=0.14$ and $t=0.2$ using the meshes obtained by WLR error indicator, Figure 4.12 (middle column). The adaptive meshes in Figure 4.12 (middle and right columns) are reconstructed from the uniform mesh $2\times100\times100$, respectively, by using the WLR error indicator and the gradient indicator. The highest level of refinement for the two indicators is $\mathcal{M}=3$. For WLR error indicator, the threshold is set $\omega=0.001\max_j(e_j)$, where e_j is the WLR error in each cell T_j. For the gradient indicator, a cell T_j is refined for $m=1,2,3$ levels if the gradient of the computed water surface w in T_j stays in the interval $[\omega'\times4^{m-1},\omega'\times4^m)$, where $\omega'=0.0005$ is the adhoc threshold parameter assigned for the gradient indicator. The interval $[\omega'\times4^{m-1},\omega'\times4^m)$ is taken from the idea of the refinement threshold used in the WLR error indicator, see Section 3.4. From Figure 4.12, it seems that the WLR error indicator is more sensitive to subtle features of the solution than the error indicator based on the gradient.

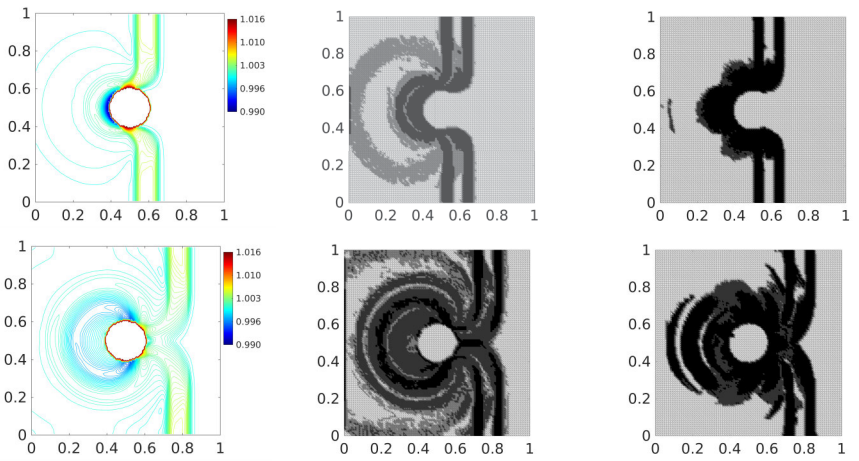


FIG. 4.12. *Example 3: w component of the solution of (4.3-4.4) at $t=0.14$ (top) and $t=0.2$ (bottom) obtained by the adaptive central-upwind scheme using WLR error indicator (left column). The corresponding adaptive meshes reconstructed from uniform mesh $2\times100\times100$ by using the WLR error indicator (middle column) and the gradient indicator (right column).*

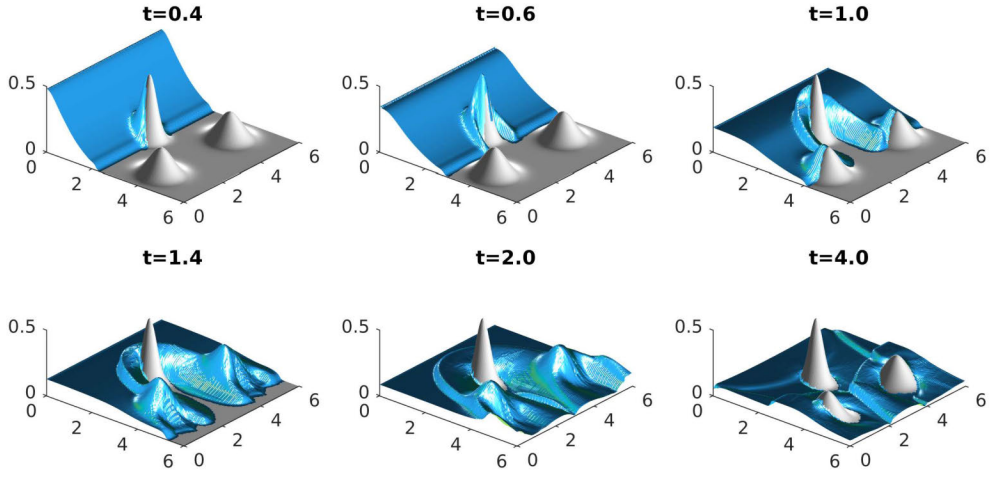


FIG. 4.13. *Example 4.4: Simulated water surface w at different times on uniform grid $2\times100\times100$.*

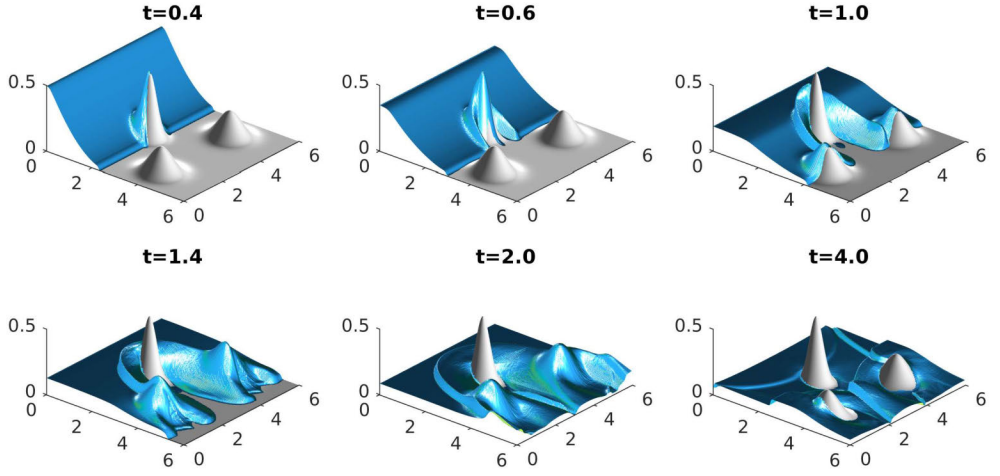


FIG. 4.14. *Example 4.4: Simulated water surface at different times on uniform grid $2\times200\times200$.*

4.4. Example 4.4—dam break test. In Example 4.4 taken from [36], we simulate the propagation of the dam-break flood wave which produces a moving wet/dry front over an irregular dry bed with three obstacles. The test allows us to verify the capability of the proposed adaptive algorithm to handle wet/dry interfaces. The bottom topography is defined by,

$$B(x,y)=\max\left[0.5e^{-8(x-2)^2-10(y-3)^2},0.2e^{-3(x-4)^2-4(y-4.8)^2},0.2e^{-3(x-4)^2-4(y-1.2)^2}\right], \tag{4.5}$$

in the computational domain $[0,6]\times[0,6]$. At $t=0$, an upstream reservoir in the region $[0,1]\times[0,6]$ filled with water up to $w(x,y,0)=0.5$ is suddenly released. Hence, the

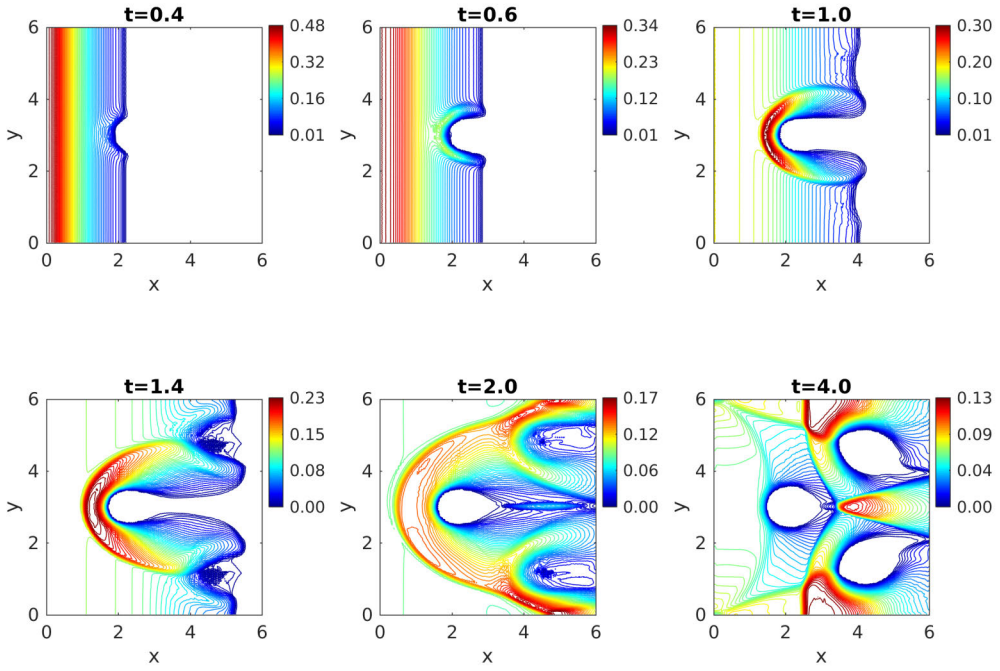


FIG. 4.15. *Example 4.4: Contour of the water depth h at different times on uniform grid* $2 \times 100 \times 100$.

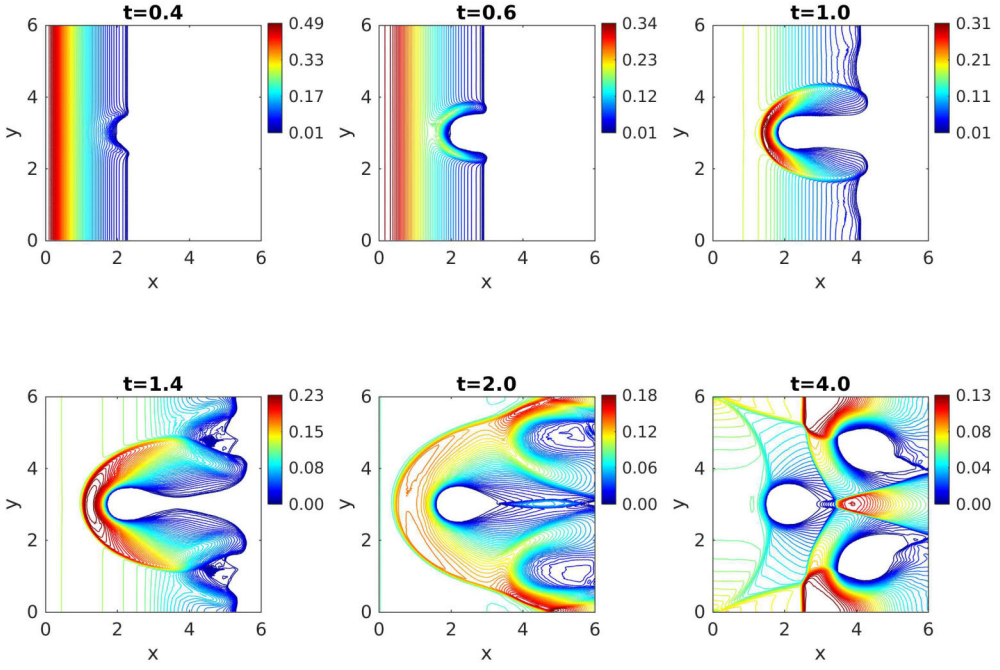


FIG. 4.16. *Example 4.4: Contour of the water depth h at different times on uniform grid* $2 \times 200 \times 200$.

following initial condition is imposed,

$$w(x,y,0)=\begin{cases}0.5, & 0\le x<1,\\ B(x,y), & \text{otherwise},\end{cases}\qquad u(x,y,0)\equiv v(x,y,0)\equiv 0. \tag{4.6}$$

In this example, we consider the friction effects by modifying the governing Equation (1.1b) and (1.1c) with the Manning friction terms as follows,

$$\begin{aligned}&(hu)_t+\left(hu^2+\frac{g}{2}h^2\right)_x+(huv)_y=-ghB_x-\frac{gn_b^2u\sqrt{u^2+v^2}}{h^{1/3}},\\ &(hv)_t+(huv)_x+\left(hv^2+\frac{g}{2}h^2\right)_y=-ghB_y-\frac{gn_b^2v\sqrt{u^2+v^2}}{h^{1/3}},\end{aligned} \tag{4.7}$$

where $n_b=0.01$ is the Manning roughness coefficient. The friction terms affect the stiffness of the underlying central-upwind scheme especially when the depth of the water is very small. To numerically solve the Saint-Venant system with the friction terms, see Equation (1.1a) and (4.7), we first rewrite the friction terms in the vector form as $\boldsymbol{I}=(0,-\frac{gn_b^2u\sqrt{u^2+v^2}}{h^{1/3}},-\frac{gn_b^2v\sqrt{u^2+v^2}}{h^{1/3}})^T$. Denote the cell averages of the friction terms at the certain time level t in cell T_j as,

$$\overline{\boldsymbol{I}}_j(t)\approx\frac{1}{|T_j|}\iint\limits_{T_j}\boldsymbol{I}(x,y)dxdy. \tag{4.8}$$

Next, we add the cell averages of the friction terms at time $t_l^{n,p}$, denoted by $\overline{\boldsymbol{I}}_j^{n,p}$, to the adaptive SSPRK2 Equations (3.6a–3.6b) as,

$$\overline{\boldsymbol{U}}_j^{(1)}=\overline{\boldsymbol{U}}_j^{n,p}-\frac{1}{|T_j^{n,\mathcal{M}_n}|}\sum_{k=1}^{3}\Delta t_{jk}^{n,p}\boldsymbol{H}_{jk}^{n,p}+\Delta t_l^{n,p}\,\overline{\boldsymbol{S}}_j^{n,p}+\Delta t_l^{n,p}\,\overline{\boldsymbol{I}}_j^{n,p}:=\boldsymbol{R}(\overline{\boldsymbol{U}}_j^{n,p},\Delta t_l^{n,p}), \tag{4.9a}$$

$$\overline{\boldsymbol{U}}_j^{n,p+1}=\frac{1}{2}\overline{\boldsymbol{U}}_j^{n,p}+\frac{1}{2}\boldsymbol{R}(\overline{\boldsymbol{U}}_j^{(1)},\Delta t_l^{n,p}). \tag{4.9b}$$

The cell averages of friction terms in a cell T_j are approximated by using the same Trapezoidal rule as the source term, see [36].

We first present the numerical solution computed by applying the central-upwind scheme without the adaptivity [36] on the uniform meshes. Figure 4.13 and Figure 4.14 are the 3-D view of the dam-break wave propagation over the initially dry bed obtained respectively on $2\times100\times100$ and $2\times200\times200$ uniform meshes at different times $t=0.4,0.6,1.0,1.4,2.0$, and 4.0. As can be observed from the figures, the water wave spreads from the reservoir and passes the obstacles. In addition, we plot the contour lines of the water depth for the solutions obtained on the uniform meshes $2\times100\times100$ (Figure 4.15) and $2\times200\times200$ (Figure 4.16). The contour lines clearly show the reflections and interactions of waves.

Now, we continue the test for the proposed adaptive central-upwind method with the refined meshes generated from the uniform mesh $2\times100\times100$. The threshold for the grid refinement is set to $\omega=0.01\max_j(e_j)$ in this example. In Figure 4.17 and Figure 4.18, we show the 3-D view of the simulated water computed by the adaptive scheme on the adaptive meshes with two cases of the highest refinement level $\mathcal{M}=1$ and $\mathcal{M}=2$. The behavior of the wave is similar to the result obtained by the central-upwind scheme without adaptivity, see Figure 4.13 and Figure 4.14. This means that

the adaptive scheme performs well in simulating the wetting/drying processes. There are no non-physical spurious waves appearing as a result of the simulation. We also present the contour lines of the water depth obtained on the adaptive meshes with $\mathcal{M}=1$ (Figure 4.19) and $\mathcal{M}=2$ (Figure 4.20). Clearly, the simulated solution captures correctly the reflections and interactions of the waves with no oscillations or disturbances showing up at the wet/dry interfaces. The considered adaptive meshes with $\mathcal{M}=1$ are plotted in Figure 4.21 and with $\mathcal{M}=2$ in Figure 4.22. As we expected, the moving refined/de-refined regions match with the wetting and drying processes in the propagation of the flow.

Finally, we present the R_{CPU} ratios at time $t=1.0$ in Table 4.7. The result in Table 4.7 shows that the average cost for the adaptive central-upwind method is about half of the cost for the central-upwind method without adaptivity. Note that at $t=1.0$, the refined region is larger than half of the computational domain, see Figure 4.21 and Figure 4.22. Therefore, the numerical cost is not as remarkably reduced with the adaptive grid as the cost in Example 2, see Table 4.6. As illustrated, the adaptive central-upwind method preserves the advantages of the well-balanced positivity preserving central-upwind scheme proposed in [36], but at a lower computational cost.

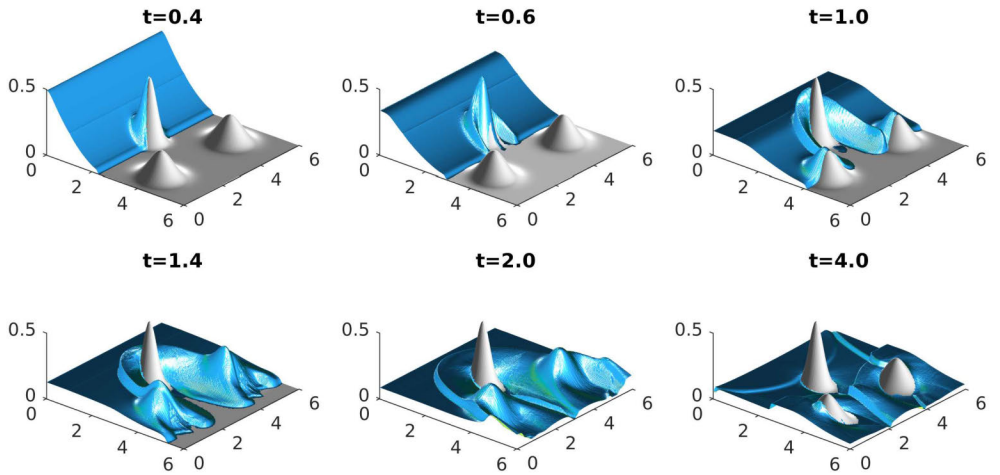


FIG. 4.17. *Example 4.4: Simulated water surface w at different times on the adaptive mesh with $\mathcal{M}=1$.*

uniform mesh (cells)	adaptive mesh $\mathcal{M}=1$ (cells)	$\mathcal{R}_{CPU}$ with $\mathcal{M}=1$	adaptive mesh $\mathcal{M}=2$ (cells)	$\mathcal{R}_{CPU}$ with $\mathcal{M}=2$
$2\times100\times100$	15,064	2.47	14,299	2.64
$2\times200\times200$	59,252	1.62	54,518	2.13
$2\times400\times400$	238,485	1.53	217,075	1.69
$\mathcal{R}_{CPU}$ average:		1.87		2.02

TABLE 4.7. *Example 4.4: The R_{CPU} ratios at $t=1.0$.*

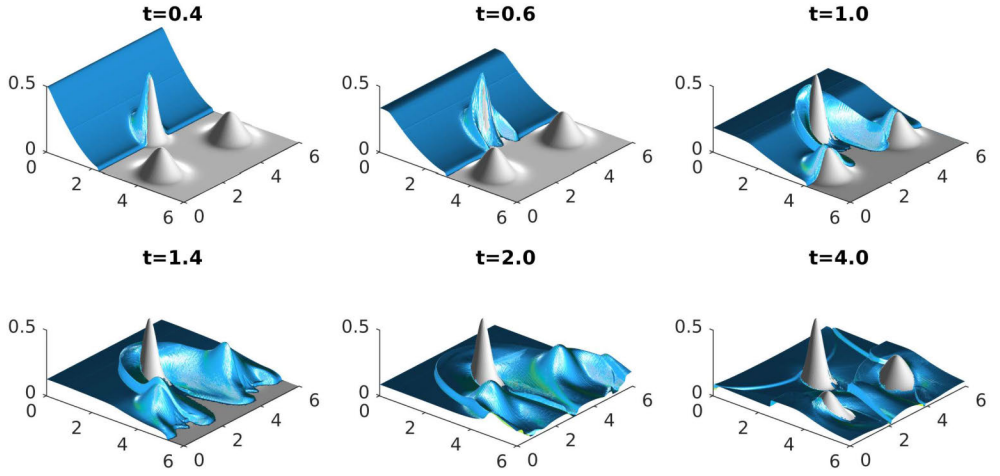


FIG. 4.18. *Example 4.4: Simulated water surface w at different times on the adaptive mesh with $\mathcal{M}=2$.*

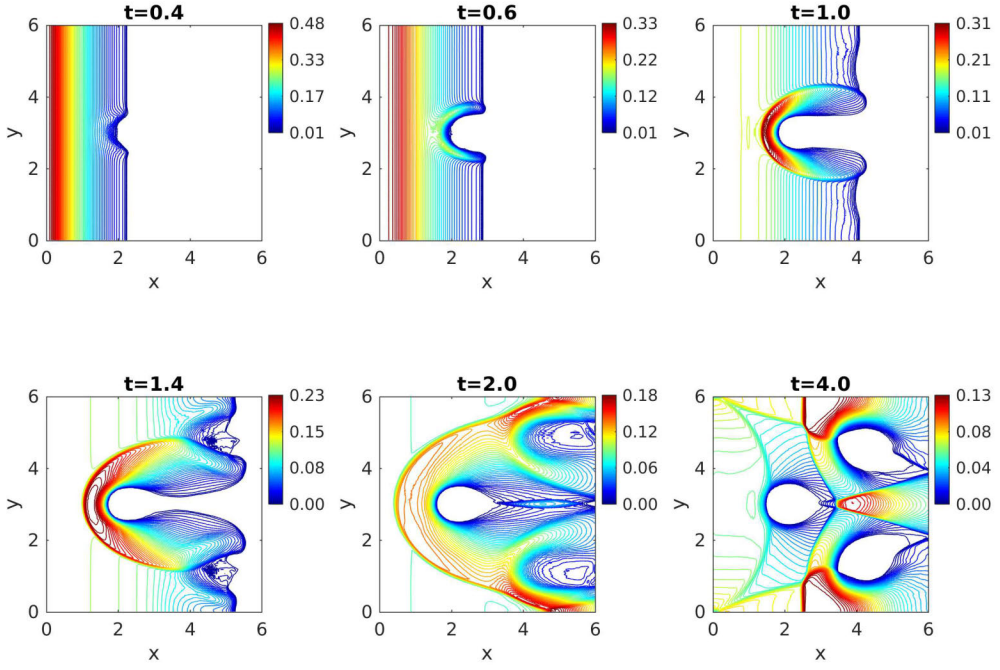


FIG. 4.19. *Example 4.4: Contour of the water depth h at different times on the adaptive mesh with $\mathcal{M}=1$.*

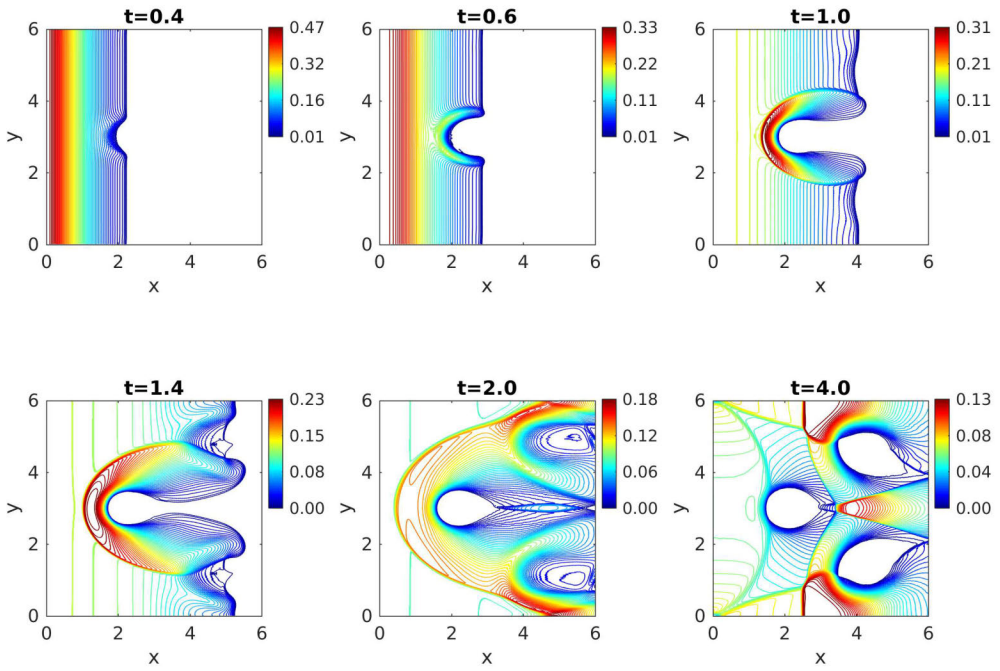


FIG. 4.20. *Example 4.4: Contour of the water depth h at different times on the adaptive mesh with* $\mathcal{M}=2$.

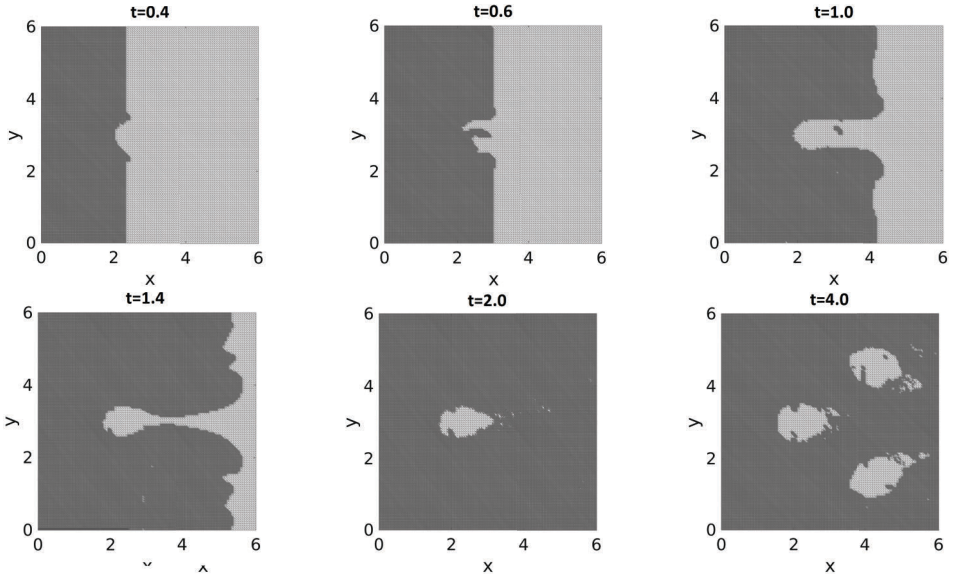


FIG. 4.21. *Example 4.4: Adaptive mesh at different times with one level of refinement* $\mathcal{M}=1$.

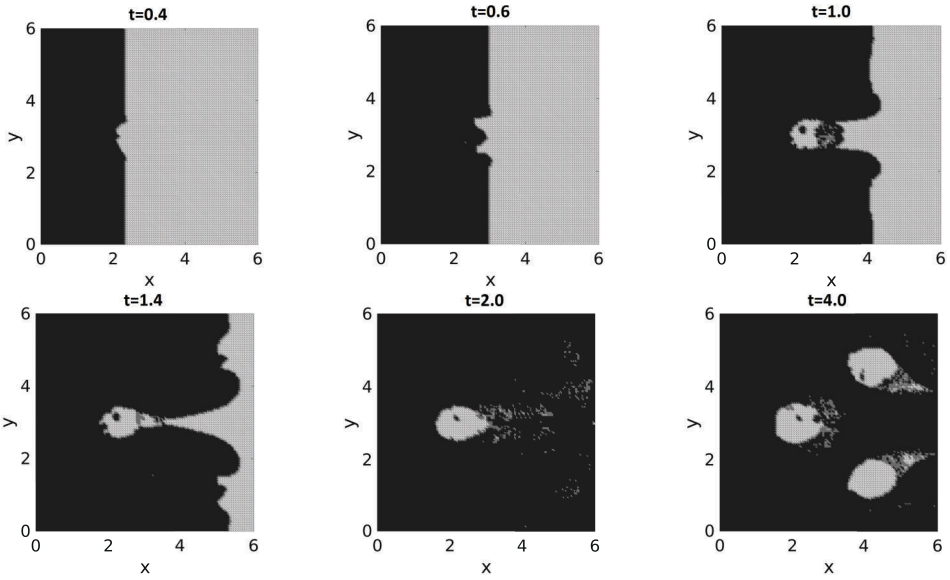


FIG. 4.22. *Example 4.4: Adaptive mesh at different times with two levels of refinement* $\mathcal{M}=2$.

5. Conclusion

We have developed a new adaptive well-balanced and positivity preserving central-upwind scheme on unstructured triangular meshes for shallow water equations. The designed scheme is an extension and improvement of the scheme in [36]. In addition, as a part of the adaptive algorithm, we obtained a robust local error indicator for the efficient mesh refinement strategy. We conducted several challenging numerical tests for shallow water equations and we demonstrated that the new adaptive central-upwind scheme maintains important stability properties (i.e., well-balanced and positivity-preserving properties) and delivers high-accuracy at a reduced computational cost.

Acknowledgments. The work of Yekaterina Epshteyn was partially supported by NSF DMS-2207207.

REFERENCES

[1] E. Audusse, F. Bouchut, M.-O. Bristeau, R. Klein, and B. Perthame, *A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows*, SIAM J. Sci. Comput., (electronic) 25:2050–2065, 2004. 1

[2] P. Azerad, J.-L. Guermond, and B. Popov, *Well-balanced second-order approximation of the shallow water equation with continuous finite elements*, SIAM J. Numer. Anal., 55:3203–3224, 2017. 1

[3] A. Bollermann, G. Chen, A. Kurganov, and S. Noelle, *A well-balanced reconstruction of wet/dry fronts for the shallow water equations*, J. Sci. Comput., 56:267–290, 2013. 1

[4] A. Bollermann, S. Noelle, and M. Lukáčová-Medvid'ová, *Finite volume evolution Galerkin methods for the shallow water equations with dry beds*, Commun. Comput. Phys., 10:371–404, 2011. 1, 2

[5] S. Bryson, Y. Epshteyn, A. Kurganov, and G. Petrova, *Central-upwind scheme on triangular grids for the Saint Venant system of shallow water equations*, AIP Conf. Proc., 1389:686–689, 2011. 1

[6] S. Bryson, Y. Epshteyn, A. Kurganov, and G. Petrova, *Well-balanced positivity preserving central-upwind scheme on triangular grids for the Saint-Venant system*, ESAIM Math. Model. Numer. Anal., 45:423–446, 2011. 1, 2, 2, 2, 4.1, 4.3

[7] S. Bryson and D. Levy, *Balanced central schemes for the shallow water equations on unstructured grids*, SIAM J. Sci. Comput., (electronic) 27:532–552, 2005. 1

[8] S. Bunya, E.J. Kubatko, J.J. Westerink, and C. Dawson, *A wetting and drying treatment for the Runge-Kutta discontinuous Galerkin solution to the shallow water equations*, Comput. Meth. Appl. Mech. Eng., 198:1548–1562, 2009. 1

[9] A. Chertock, Y. Epshteyn, H. Hu, and A. Kurganov, *High-order positivity-preserving hybrid finite-volume-finite-difference methods for chemotaxis systems*, Adv. Comput. Math., 44:327–350, 2018. 1

[10] A. Chertock, K. Fellner, A. Kurganov, A. Lorz, and P.A. Markowich, *Sinking, merging and stationary plumes in a coupled chemotaxis-fluid model: a high-resolution numerical approach*, J. Fluid Mech., 694:155–190, 2012. 1

[11] A. Chertock, A. Kurganov, and Y. Liu, *Central-upwind schemes for the system of shallow water equations with horizontal temperature gradients*, Numer. Math., 127:595–639, 2014. 1

[12] A.J.C. de Saint-Venant, *Thèorie du mouvement non-permanent des eaux, avec application aux crues des rivière at à l'introduction des marèes dans leur lit*, C.R. Acad. Sci. Paris, 73:147–154, 1871. 1

[13] M.O. Domingues, S.M. Gomes, O. Roussel, and K. Schneider, *An adaptive multiresolution scheme with local time stepping for evolutionary PDEs*, J. Comput. Phys., 227:3758–3780, 2008. 3.3

[14] R. Donat, M. Martí, A. Martínez-Gavara, and P. Mulet, *Well-balanced adaptive mesh refinement for shallow water flows*, J. Comput. Phys., 257:937–953, 2014. 3.3

[15] U.S. Fjordholm, S. Mishra, and E. Tadmor, *Energy preserving and energy stable schemes for the shallow water equations*, in F. Cucker, A. Pinkus, and M.J. Todd (eds.), Foundations of Computational Mathematics, Hong Kong 2008, Cambridge Univ. Press, Cambridge, 363:93–139, 2009. 1

[16] U.S. Fjordholm, S. Mishra, and E. Tadmor, *Well-balanced and energy stable schemes for the shallow water equations with discontinuous topography*, J. Comput. Phys., 230:5587–5609, 2011. 1

[17] T. Gallouët, J.-M. Hérard, and N. Seguin, *Some approximate Godunov schemes to compute shallow-water equations with topography*, Comput. Fluids, 32:479–513, 2003. 1

[18] D.L. George, *Finite volume methods and adaptive refinement for tsunami propagation and inundation*, Ph.D thesis, University of Washington, 2006. 1

[19] M.A. Ghazizadeh, A. Mohammadian, and A. Kurganov, *An adaptive well-balanced positivity preserving central-upwind scheme on quadtree grids for shallow water equations*, Comput. Fluids, 208:104633, 2020. 1, 4.3

[20] S. Gottlieb, C.-W. Shu, and E. Tadmor, *Strong stability-preserving high-order time discretization methods*, SIAM Rev., 43:89–112, 2001. 2.2, 3.3

[21] N. Hannoun and V. Alexiades, *Issues in adaptive mesh refinement implementation*, Elecron. J. Differ. Equ., 15:141–151, 2007. 3.2

[22] S. Jin, *A steady-state capturing method for hyperbolic systems with geometrical source terms*, M2AN Math. Model. Numer. Anal., 35:631–645, 2001. 1

[23] S. Jin and X. Wen, *Two interface-type numerical methods for computing hyperbolic systems with geometrical source terms having concentrations*, SIAM J. Sci. Comput., 26:2079–2101, 2005. 1

[24] S. Karni and A. Kurganov, *Local error analysis for approximate solutions of hyperbolic conservation laws*, Adv. Comput. Math., 22:79–99, 2005. 3.4

[25] A. Kurganov and D. Levy, *Central-upwind schemes for the Saint-Venant system*, M2AN Math. Model. Numer. Anal., 36:397–425, 2002. 1

[26] A. Kurganov and Y. Liu, *New adaptive artificial viscosity method for hyperbolic systems of conservation laws*, J. Comput. Phys., 231:8114–8132, 2012. 1

[27] A. Kurganov and J. Miller, *Central-upwind scheme for Savage-Hutter type model of submarine landslides and generated tsunami waves*, Comput. Meth. Appl. Math., 14:177–201, 2014. 1

[28] A. Kurganov, S. Noelle, and G. Petrova, *Semidiscrete central-upwind schemes for hyperbolic conservation laws and Hamilton-Jacobi equations*, SIAM J. Sci. Comput., 23:707–740, 2001. 1

[29] A. Kurganov and G. Petrova, *Central-upwind schemes on triangular grids for hyperbolic systems of conservation laws*, Numer. Meth. Partial Differ. Equ., 21:536–552, 2005. 1

[30] A. Kurganov and G. Petrova, *A second-order well-balanced positivity preserving scheme for the Saint-Venant system*, Commun. Math. Sci., 5(1):133–160, 2007. 1

[31] A. Kurganov and E. Tadmor, *New high-resolution central schemes for nonlinear conservation laws and convection-diffusion equations*, J. Comput. Phys., 160:241–282, 2000. 1

[32] A. Kurganov and E. Tadmor, *New high-resolution semi-discrete central schemes for Hamilton-Jacobi equations*, J. Comput. Phys., 160:720–742, 2000. 1

[33] R. LeVeque, *Balancing source terms and flux gradients in high-resolution Godunov methods: the quasi-steady wave-propagation algorithm*, J. Comput. Phys., 146:346–365, 1998. 1, 4.2

[34] R. LeVeque, *Finite Volume Methods for Hyperbolic Problems*, Cambridge Texts in Applied Mathematics, Cambridge University Press, Cambridge, 2002. 1

[35] M. Li, P. Guyenne, F. Li, and L. Xu, *A positivity-preserving well-balanced central discontinuous Galerkin method for the nonlinear shallow water equations*, J. Sci. Comput., 71:994–1034, 2017. 1

[36] X. Liu, J. Albright, Y. Epshteyn, and A. Kurganov, *Well-balanced positivity preserving central-upwind scheme with a novel wet/dry reconstruction on triangular grids for the Saint-Venant system*, J. Comput. Phys., 374:213–236, 2018. 1, 2, 2, 2, 2, 2, 2, 2, 2.2, 3.2, 3.2, 4, 4.3, 4.3, 4.4, 4.4, 5

[37] M. Moreira Lopes, M.O. Domingues, K. Schneider, and O. Mendes, *Local time-stepping for adaptive multiresolution using natural extension of Runge–Kutta methods*, J. Comput. Phys., 382:291–318, 2019. 3.3

[38] H. Nessyahu and E. Tadmor, *Nonoscillatory central differencing for hyperbolic conservation laws*, J. Comput. Phys., 87:408–463, 1990. 1

[39] S. Noelle, N. Pankratz, G. Puppo, and J. Natvig, *Well-balanced finite volume schemes of arbitrary order of accuracy for shallow water flows*, J. Comput. Phys., 213:474–499, 2006. 1

[40] B. Perthame and C. Simeoni, *A kinetic scheme for the Saint-Venant system with a source term*, CALCOLO, 38:201–231, 2001. 1

[41] G. Russo, *Central schemes for balance laws*, in H. Freistühler and G. Warnecke (eds.), Hyperbolic Problems: Theory, Numerics, Applications, Birkhäuser, Basel, 821–829, 2001. 1

[42] G. Russo, *Central schemes for conservation laws with application to shallow water equations*, in S. Rionero and G. Romano (eds.), Trends and Applications of Mathematics to Mechanics, Springer, Milano, 225–246, 2005. 1

[43] M.L. Sætra, A.R. Brodtkorb, and K.A. Lie, *Efficient GPU-implementation of adaptive mesh refinement for the shallow-water equations*, J. Sci. Comput., 63:23–48, 2015. 1

[44] H. Shirkhani, A. Mohammadian, O. Seidou, and A. Kurganov, *A well-balanced positivity-preserving central-upwind scheme for shallow water equations on unstructured quadrilateral grids*, Comput. Fluids, 126:25–40, 2016. 1

[45] E. Tadmor, *Local error estimates for discontinuous solutions of nonlinear hyperbolic equations*, SIAM J. Numer. Anal., 28:891–906, 1991. 3.4

[46] Y. Xing and C.-W. Shu, *High order finite difference WENO schemes with the exact conservation property for the shallow water equations*, J. Comput. Phys., 208:206–227, 2005. 1

[47] Y. Xing and C.-W. Shu, *High order well-balanced finite volume WENO schemes and discontinuous Galerkin methods for a class of hyperbolic systems with source terms*, J. Comput. Phys., 214:567–598, 2006. 1