Nat Hazards (2018) 91:567–586
https://doi.org/10.1007/s11069-017-3141-7

ORIGINAL PAPER

Hydrodynamic modeling of flash flood in mountain watersheds based on high-performance GPU computing

Xiaozhang Hu[1] · **Lixiang Song**[1]

Received: 16 September 2017 / Accepted: 27 November 2017 / Published online: 8 December 2017

Abstract Numerical accuracy and computational efficiency are the two key factors for flash flood simulation. In this paper, a two-dimensional fully hydrodynamic model is presented for the simulation of flash floods in mountain watersheds. A robust finite volume scheme is adopted to accurately simulate the overland flow with wet/dry fronts on highly irregular topography. A graphics processing unit-based parallel method using OpenACC is adopted to realize high-performance computing and then improve the computational efficiency. Since the finite volume scheme is explicit which involves many computationally intensive loop structures without data dependence, the parallel flash flood model can be easily realized by using OpenACC directives in an incremental developing way based on the serial model codes, except that data structure and transportation should be optimized for parallel algorithm. Model accuracy is validated by benchmark cases with exact solutions and experimental data. To further analyze the performance of the model, we considered a real flash flooding-prone area in China using a NVIDIA Tesla K20c card and three grid division schemes with different resolution. Results show that the proposed model can fast simulate the rainfall−runoff process related to the rapid mountain watersheds response, and a higher speedup ratio can be achieved for finer grids resolution. The proposed model can be used for real-time prediction of large-scale flash flood on high-resolution grids and thus has bright application prospects.

Keywords Flash flood · Mountain watershed · Hydrodynamic model · Finite volume · GPU computing · Numerical simulation

✉ Lixiang Song
slx.hust@live.cn

[1] Department of Water Resources and Environment, Pearl River Hydraulic Research Institute, Guangzhou 510611, China

1 Introduction

Flash flood due to excessive rainfall is a serious threatening environment hazard which causes loss of human life and substantial socioeconomic damage. Climate change of increasing the intensity of rainfall in some parts of the world may lead to more severe flash flooding. In addition, increased urbanization in flash flooding-prone zones will result in larger segments of population being alive under high level of flood risk. So great attention has been paid to flash flood disaster in the past few decades, and many works have been focused on establishing effective early warning systems and further disaster mitigation measures (Liu et al. 2010; Kourgialas et al. 2012; Yang et al. 2016; Zeng et al. 2016; Elfeki et al. 2017; Lian et al. 2017). For example, an accurate and timely warning can provide flood control departments with more time to prepare an effective response, such as moving people to safer locations and dispatching flood material. As an important approach to help improve warning lead times, simulation of flash flood is of interest and is required to identify the threshold value of critical rainfall and support practical flood warning (Pender et al. 2010).

Flash floods have a complex flow pattern in which not only general subcritical flow, critical flow, and supercritical flow tend to occur at the same time, but also many computational grids would be partially wetted by overland flow. In the rainfall−runoff process related to the rapid mountain watersheds response, the flooding process often occurs in dry slope land and river beds, giving rise to a number of difficulties in the numerical calculations, such as negative water depth due to overdraft, unphysical large velocity at wet/dry fronts, discontinuous flow with large surface gradients, and so on. To improve the numerical accuracy and stability, computational mesh refinement is always required. However, the runtime would increase significantly as the number of computational meshes increases; for example, the computational cost of two-dimensional hydrodynamic model at scales approaching 1 m would be roughly 10^{12} operations per km^2 per day (Sanders et al. 2010). In general, flash flood is a rapid flood event that is characterized by the time-to-peak of the hydrograph being less than 6 h (Tao and Barros 2013). So a practical warning service should be provided where times are shorter enough than this. Therefore, numerical models used for simulation of flash flood should overcome simultaneously the two key problems of accuracy and efficiency (Zhang et al. 2016, 2017).

Although numerical models of river flood or urban flood simulation have been studied for many years, two persistent problems dog the schemes in practice for flash flood simulation in mountain watersheds. Firstly, when modeling realistic flash flood in mountain watersheds over highly irregular bottom topography, negative water depth computed by numerical model would be notable and induce serious water non-conservation problem. Secondly, flash flood in mountain watersheds involves overland flow which is characterized by very small water depth. Use of a naive method can lead to predict unphysical high velocities at wet/dry fronts. So a robust model is critical to simulate flash flood in mountain watersheds (Begnudelli and Sanders 2006; Liang and Borthwick 2009; Song et al. 2011; Kim et al. 2014; Guinot et al. 2017). For example, Song et al. (2011) proposed a robust finite volume model for shallow water flows with wet/dry fronts over irregular terrain. A new formulation of the shallow water equations and sloping bottom model are adopted to preserve mass conservation in both fully and partially wetted cells, and the friction terms are solved by a semi-implicit scheme that resolves the stiff problem.

Computational efficiency is very important for flash flood disaster prediction. However, in a watershed scale the number of computational grids is commonly very large, and the computational complexity is much more than that of a river floods modeling. So high-

performance computing technology should be adopted to improve the traditional serial computing model. In recent years, parallel algorithms using multi-CPU-based and graphics processing unit (GPU)-based hardware architecture have been introduced to improve the computational efficiency of numerical models (Sanders et al. 2010; Lai and Khan 2016; Herdman et al. 2012; Wang et al. 2014; Liang et al. 2016; Zhang et al. 2016, 2017). For multi-CPU-based parallel computing models, message-passing interface (MPI) method and shared memory method (OpenMP) are widely used for many years. In the view of hardware architecture, a piece of GPU card has multi-streaming processors with thousands of computing cores, which are very important for dealing with computationally intensive task. Thus, one GPU card could provide the same powerful computational capabilities as many-CPU cores and GPU-based models can run with high speedup ratio and parallel efficiency.

With the development of GPU hardware and coding platforms in recent years, GPU-based high-performance computing models are now one of the most popular parallel algorithms in the world. There are various approaches and platforms for GPU parallel computation implementation (Herdman et al. 2012). Among those methods, CUDA programming language such as CUDA Fortran/C or OpenCL, and OpenACC command language would be appropriate for hydrodynamic codes development. CUDA programming language is a low-level development approach with high flexibility in parallel programming, while OpenACC command language supports an incremental coding way by using compiler directives and runtime routines. When using the CUDA programming language, the thread structure of GPU device should be defined explicitly by developer as a specific form, i.e., <<<grid, block, thread>>>. The grid, block, and thread represent three different levels in the thread structure of CUDA. And then, the parallel regions should be coded as kernels for GPU parallel computing. When using the OpenACC supported by PGI compiler, the parallel region code is automatically transplanted from CPU to GPU by the compiler, and massive amounts of data are handled in the accelerator device with auto-optimal parameters of executive model. So OpenACC is easier to use as well as has the advantages of better portability. Herdman et al. (2012) proposed a finite volume model of solving compressible Euler equations by using OpenACC directives, and a max speedup of 4.91 is achieved by using X2090 processors. Zhang et al. (2017) developed a dam-break flood parallel computation model based on OpenACC applications, and a max speedup of 20.7 is achieved by using NVIDIA Kepler K20c platform when the number of computational grids is 337084, while the speedup ratio is 10.77 when the number of computational grids is 18109. Liang et al. (2016) proposed a GPU-based catchment-scale high-resolution flash flood model coded by OpenCL language, and the simulation of 5 million computational nodes is nearly 5 times faster than real time by using a single Tesla K80 GPU.

In this work, the two-dimensional shallow water model proposed by Song et al. (2011) is further developed and applied to simulate the flash flood in mountain watersheds with rainfall−runoff process. A GPU-based parallel method using OpenACC applications is adopted to realize high-performance computing. On the basis of serial model codes which have been validated through various benchmarks and applications, the OpenACC directives are used in an incremental developing way with minimal recoding work, except that the data structure and transportation are optimized for parallel algorithm. The applicability of the model is demonstrated as applied to real flash flooding-prone areas in China. Three grid division schemes with different resolution are adopted to further analyze the performance of the model. Results show that the proposed parallel model can fast simulate the rainfall−runoff process related to the rapid mountain watersheds response, and a higher speedup ratio can be achieved for finer grids resolution.

2 Governing equations

The classical shallow water equations are referred to the formulation of hydrostatic pressure and bottom slope effects and their division between fluxes and source terms. In this paper, the new formulation of the classical two-dimensional shallow water equations proposed by Song et al. (2011) is adopted as the governing equations, as well as the rainfall and infiltration is incorporated in the source term:

$$\frac{\partial \mathbf{U}}{\partial t}+\frac{\partial \mathbf{E}}{\partial x}+\frac{\partial \mathbf{G}}{\partial y}=\mathbf{S} \tag{1}$$

in which, $\mathbf{U}=\begin{bmatrix} h \\ hu \\ hv \end{bmatrix}$, $\mathbf{E}=\begin{bmatrix} hu \\ hu^2+g(h^2-b^2)/2 \\ huv \end{bmatrix}$, $\mathbf{G}=\begin{bmatrix} hv \\ huv \\ hv^2+g(h^2-b^2)/2 \end{bmatrix}$, $\mathbf{S}=\mathbf{S}_0+\mathbf{S}_f=\begin{bmatrix} r-f \\ g(h+b)S_{0x} \\ g(h+b)S_{0y} \end{bmatrix}+\begin{bmatrix} 0 \\ -ghS_{fx} \\ -ghS_{fy} \end{bmatrix}$ where h is the water depth; u and v are the velocity components in the x and y directions, respectively; b is the bed elevation; r and f are the source and sink terms due to rainfall and infiltration; S_{0x} and S_{0y} are bed slopes in the x and y directions, respectively, assume the bed is fixed, i.e., $b=b(x,y)$, then $S_{0x}=-\partial b/\partial x$ and $S_{0y}=-\partial b/\partial y$; g is the gravity acceleration; S_{fx} and S_{fy} are the friction terms in the x and y directions, respectively. The friction terms are estimated by Manning formulae:

$$S_{\mathrm{fx}}=\frac{n^2u\sqrt{u^2+v^2}}{h^{4/3}},\ S_{\mathrm{fy}}=\frac{n^2v\sqrt{u^2+v^2}}{h^{4/3}} \tag{2}$$

It should be noted that the new formulation can essentially eliminate the need for momentum flux corrections, providing a robust alternative to the classical form for preserving the C-property. Besides, the new formulation is mass conservative in the framework of sloping bottom model, which represents numerically irregular terrain with second-order accuracy (Song et al. 2011).

3 Numerical scheme

In order to simulate the rainfall−runoff process related to the rapid mountain watersheds response, a robust finite volume scheme (Song et al. 2011) is adopted for numerical resolution of the governing equations. Triangular grid is used to facilitate grid generation and localized refinement when modeling the natural waterways in mountain watershed. The slope-limited linear reconstruction method is used to achieve second-order accuracy in space where solutions are smooth, and to preserve first-order accuracy but high-resolution where great gradient or wet/dry front exists. The model adopts the Harten–Lax–van Leer contact wave (HLLC) approximate Riemann solver (Toro 2001) to compute inner interface fluxes, and the Hancock's predictor-corrector scheme for efficient time stepping as well as second-order temporal accuracy.

Briefly, the process of time stepping is as follows:

Step 1 Calculate the limited gradient by central values for each cell.

In this paper, the unlimited gradient of a cell is computed by the three neighbor central values, and the limiting approach of Hubbard (Hubbard 1999) is used to get the limited gradient.

Step 2 Calculate the predictor results for each cell.

The predictor results are given by

$$\begin{aligned}
\eta_i^{k+1/2} &= \eta_i^k - \frac{\Delta t}{2}\left(h\bar{\partial}_x u + h\bar{\partial}_y v + u\bar{\partial}_x h + v\bar{\partial}_y h\right)\Big|_i^k \\
u_i^{k+1/2} &= \frac{1}{1 + gn^2 h^{-4/3}\sqrt{u^2+v^2}\Delta t/2}\left[u - \frac{\Delta t}{2}\left(u\bar{\partial}_x u + g\bar{\partial}_x \eta + v\bar{\partial}_y u\right)\right]\Big|_i^k \\
v_i^{k+1/2} &= \frac{1}{1 + gn^2 h^{-4/3}\sqrt{u^2+v^2}\Delta t/2}\left[v - \frac{\Delta t}{2}\left(u\bar{\partial}_x v + g\bar{\partial}_y \eta + v\bar{\partial}_y v\right)\right]\Big|_i^k
\end{aligned} \tag{3}$$

in which η is water level; Δt is computational time step controlled by the CFL condition; $\bar{\partial}_x$ and $\bar{\partial}_y$ denote the limited gradient in the x and y directions, respectively; the superscripts k and $k + 1/2$ denote the current time step and the predictor step, respectively.

Step 3 Calculate the reconstructed values for each edge.

The reconstructed values at time step $n + 1/2$ are computed from the predictor results and limited gradients at the basis time step, which is given by

$$p_{\mathrm{L}} = p_i^{n+1/2} + \bar{\nabla}_i p^n \cdot \mathbf{r}_i \, p_{\mathrm{R}} = p_j^{n+1/2} + \bar{\nabla}_j p^n \cdot \mathbf{r}_j \tag{4}$$

where the subscripts L and R denote the left state and right state of the edge, respectively, which are used for constructing local Riemann problem and numerical flux computation; the subscripts i, j denote the left side and right side cell of the edge, respectively; p denotes η, u or v; $\mathbf{r}$ denote the position vector of middle point of the edge relative to the cell center; $\bar{\nabla} p$ denotes the limited gradient of p.

Step 4 Calculate the numerical fluxes for each edge.

The HLLC approximate Riemann solver is adopted to compute face fluxes, which is given by

$$\mathbf{F}(\mathbf{U}_{\mathrm{L}}, \mathbf{U}_{\mathrm{R}}) \cdot \mathbf{n} = \begin{cases} \mathbf{F}(\mathbf{U}_{\mathrm{L}}) \cdot \mathbf{n} & \text{if } s_1 \geq 0 \\ \mathbf{F}_{*\mathrm{L}} & \text{if } s_1 < 0 \leq s_2 \\ \mathbf{F}_{*\mathrm{R}} & \text{if } s_2 < 0 < s_3 \\ \mathbf{F}(\mathbf{U}_{\mathrm{R}}) \cdot \mathbf{n} & \text{if } s_3 \leq 0 \end{cases} \tag{5}$$

where $\mathbf{U}_{\mathrm{L}}$ and $\mathbf{U}_{\mathrm{R}}$ denote the reconstructed values; s_1, s_2, s_3 are wave speeds. More details about the HLLC solver may refer to Song et al. (2011).

Step 5 Calculate the source terms approximation for each cell.

Source terms include rainfall and infiltration intensity, bed slope term, and friction term. The rainfall and infiltration intensity, as well as bed slope term, are approximated explicitly, while the friction term is computed by a semi-implicit scheme. In general, the rainfall intensity is provided by design values or field measurements, while the infiltration

rate can be determined using the popular Green–Ampt approximation or an assumed constant value.

Step 6 Calculate the solutions at the next time step for each cell.

The solutions at the next time step $k+1$ are given as

$$\mathbf{U}_i^{k+1} = \mathbf{U}_i^k - \frac{1}{\Omega_i}\sum_{m=1}^{3}\mathbf{F}_m(\mathbf{U}_{\mathrm{L}}^{k+1/2}, \mathbf{U}_{\mathrm{R}}^{k+1/2})\cdot\mathbf{n}_m L_m + \frac{1}{\Omega_i}\mathbf{S}_i^k \tag{6}$$

where m and L are the edge number and length, respectively; $\mathbf{F}$ is the numerical flux; Ω is the area; $\mathbf{S}$ is the source approximation. The superscripts $k+1/2$ denote that the reconstructed variables are computed on the predictor results.

After the water depth is updated, the water level at the next time step is updated by the VFRs proposed by Begnudelli and Sanders (2006), and then, cells can be classified as fully submerged, partially submerged, and dry by using the horizontal free surface classification method proposed by Begnudelli and Sanders (2007).

4 GPU computing

From the process of time stepping described above, it can be concluded that there are many computationally intensive loop structures without data dependence, i.e., the calculations for a cell or edge are uncorrelated with other cells or edges in the current time step, since the finite volume scheme adopted to simulate flash flood is explicit. So the calculation loops are natural parallelizable and can be executed in GPU-based parallel model. On the basis of serial FORTRAN codes, an incremental developing way with minimal recoding work can be achieved by using OpenACC directives and clauses, so the PGI compiler (version 2017) which supports OpenACC FORTRAN is adopted to realize GPU-based high-performance computing in this work.

4.1 Design of calculation flow

OpenACC is a high-level programming model that parallelizes loops automatically by using directives and clauses. In the OpenACC abstract accelerator, both computation and data are offloaded from a host device (CPU) to an accelerator device (GPU). So the parallelization of the flash flood model involves three steps in the OpenACC parallel model: data transportation from CPU to GPU, numerical calculation in GPU, and data transportation from GPU to CPU. Since the PCIe (peripheral component interface express)-based data transportation between CPU and GPU is time-consuming, all numerical calculations should be executed in GPU to avoid unnecessary data transportation. Figure 1 presents the computational flow diagram on CPU–GPU heterogeneous platform. When data including mesh information, initial and boundary condition, and computation parameters are initialized in CPU, the corresponding GPU device space is allocated by CPU and those initial data are transported from CPU to GPU. Then, all numerical calculation subroutines are successively called by CPU but parallel executed by numerous computation cores of GPU with three levels of parallelism: gang, worker, and vector. At the same time, CPU waits for the completion of GPU computation and transports data from GPU to CPU when it is time to output results. As the calculation flow described above, the parallel model is CPU-directed execution with an attached GPU device. All

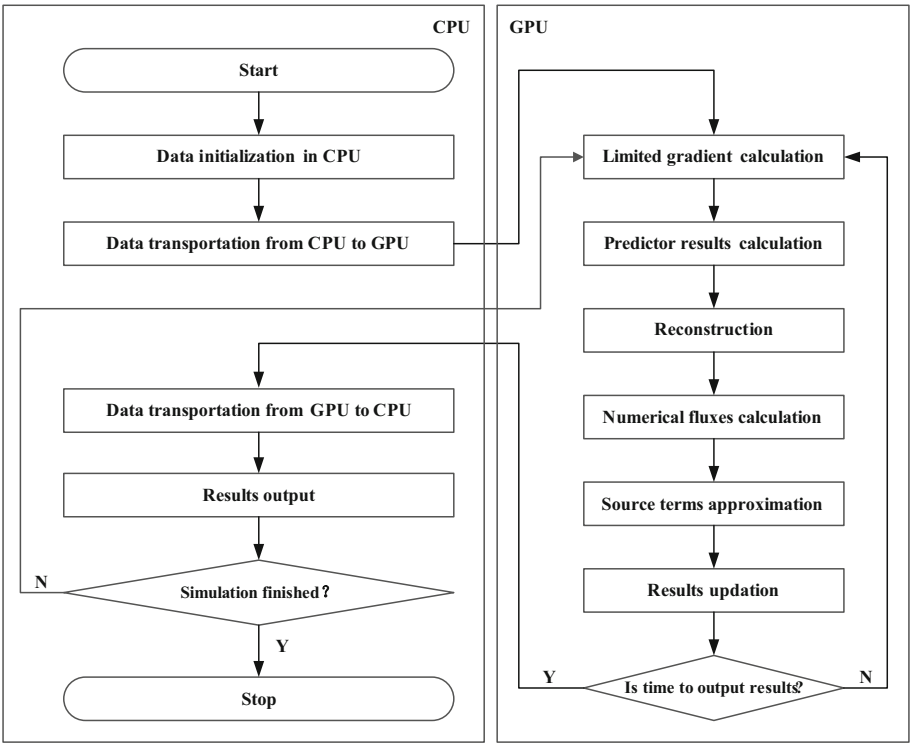

Fig. 1 Computing flow diagram

computationally intensive loop structures and the corresponding data are parallel handled in the GPU device, and the CPU device is only used to control the calculation process with data inputs and outputs.

By using OpenACC, the thread structure of GPU device is defined by three levels of parallelism: gang, worker, and vector. Vector parallelism has the finest granularity, gang parallelism is coarse-grained parallelism, and worker parallelism sits between vector and gang levels. A gang consists of 1 or more workers, each of which operates on a particular vector length. Vector operations are performed with the length, indicating how many data elements may be operated on with the same instruction. It should be noted that using these three levels of parallelism as well as sequential, we can map the parallelism in the code to GPU. Fortunately, this mapping can be implemented implicitly by the compiler using what it knows about the target device, i.e., massive amounts of data can be handled in the accelerator device (GPU) with auto-optimal parameters of gangs-number, workers-number, and vector-length to efficiently use thousands of computing cores. This makes OpenACC parallel model highly portable, since the same code may be mapped to all types of GPU devices.

4.2 Data management

OpenACC exposes separate memory through the device data environment of CPU and GPU. In this paper, the unstructured data management method is adopted by using the

OpenACC clause "*!$acc declare create (data array),*" and then, the program creates a copy of the GPU device for the first time after declaring data array in the CPU device space. Any declared global variables in the parallel loops region or kernels will be regard as GPU device data, but the declared global variables would be regard as CPU device data when used in the CPU-computed loops region. So the OpenACC clause "*!$acc update host (data array)*" should be used for data transportation from GPU to CPU, while the clause "*!$acc update device (data array)*" should be used for data transportation from CPU to GPU. Since the GPU device space is allocated once in the initial process, and the number of times of data transportation is limited, the data management approach can improve the performance of the parallel algorithm.

4.3 Parallelization

Since the unstructured triangular meshes are adopted for finite volume discretization, most of the numerical calculations are single-layer loops with one-dimensional arrays. So the OpenACC clause of "*!$acc parallel loop present(data array)*" is adopted for parallelization. In the view of data dependence, numerical calculations can be classified into two categories: The first is computational loops without reduction and atomic operation, and the second is computational loops with reduction or atomic operation. Figure 2 presents the corresponding two methods for parallelization, from which it can be seen that an incremental developing way with minimal recoding work can be achieved by using OpenACC directives and clauses.

5 Results and discussion

In this section, the numerical accuracy of the proposed model to represent flash flood routine process is demonstrated by firstly considering two idealized rainfall−runoff cases over a sloping bottom with analytical solutions and a laboratory experiment. To further

```
Subroutine Comp1
    Use Global, only : Flo2D
    Implicit none
    Integer*4          :: i
    !$acc parallel loop present (Flo2D)
    Do i =1, edgesNum
        …
        …
        …
        …
    End do
    !$acc end parallel loop
End subroutine
```

(a) without reduction and atomic operation

```
Subroutine Comp2
    Use Global, only : Flo2D
    Implicit none
    Integer*4          :: i
    Real*8             :: speed
    Speed = 0.0d0
    !$acc parallel loop present (Flo2D)
    !$acc reduction (max: Speed)
    Do i =1, edgesNum
      Speed=max(Speed, Flo2D%Speed(i))
    End do
    !$acc end parallel loop
End subroutine
```

(b) with reduction operation

Fig. 2 Examples for parallelizing loop

analyze the performance of the model, we considered a real flash flooding-prone area in China and three grid division schemes with different resolution. A NVIDIA Tesla K20c card is used in this work for parallel computation. This card has a Kepler GK110 GPU and 2496 NVIDIA CUDA cores, meaning the proposed model can be executed efficiently with the OpenACC application. The time step is governed by the CFL number of 0.85 for all computation cases. It should be noted that both the serial model and parallel model are used for testing in this work, and results show that the GPU-based simulations are the same as for the serial model, indicating the veracity of the parallel model.

5.1 Overland flow over a sloping bottom

The first test problem is chosen to verify that the present model indeed accurately simulates the overland flow resulted from rainfall over a sloping bottom, which has analytical solutions. This example involves two test cases of one-dimensional overland flow, which were also used by Singh et al. (2015) for model verification. In the first case proposed by Giammarco et al. (1996), the computational domain is a 1000 m $\times$ 800 m rectangular channel with a slope of 0.05 in the y direction. A steady, uniform excess rainfall of 10.8 mm/h for the first 1.5 h is applied over the channel. This case was simulated using uniform triangular meshes ($\xi = 10$ m). The Manning's n is set to 0.015. Solid slip conditions are imposed at the upstream and lateral walls of the channel, while the downstream boundary is a free outlet. The bed is initially dry. We run the simulation until $t = 12{,}000$ s. Figure 3 shows the comparison of simulated hydrograph and the analytical solution at the outlet: As it can be seen, a very good agreement is observed. Note that both the analytical solution and numerical result of Singh et al. (2015) are digitally extracted from Singh et al. (2015).

The second case proposed by Singh (1996) is the overland flow generated over a sloping plane. This test case was used by other researchers for model verification (Tsai and Yang

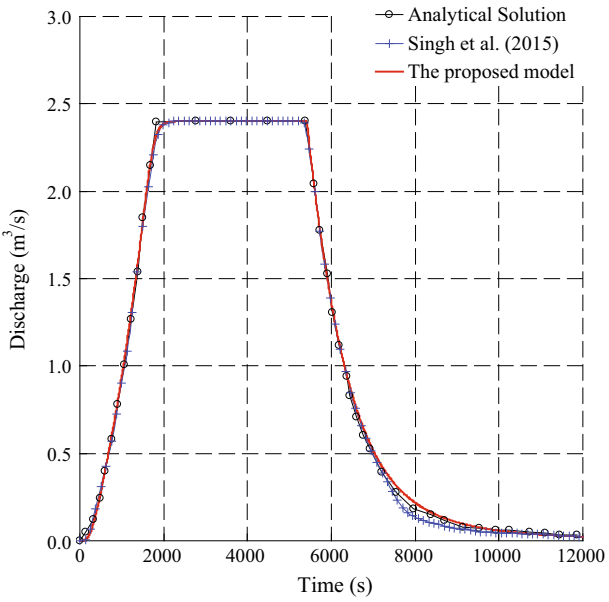

Fig. 3 Comparison of the analytical solution and simulated discharge for Case 1

2005; Singh et al. 2015). The computation is a sloping plane with a length of 500 m and slope of 0.01. The boundary and initial conditions are the same as the former test case. A steady, uniform excess rainfall intensity of 300 mm/h for duration of the first 1600 s is applied over the plane. This case was simulated using uniform triangular meshes ($\xi = 1$ m). The Manning's n is set to 0.02. We run the simulation until $t = 6000$ s. Figure 4, which is bounded to clearly show the results in the peak period, presents the comparison of the simulated and analytical solutions for water depth at the outlet. Similarly, both the analytical solution and numerical result of Singh et al. (2015) are digitally extracted from Singh et al. (2015). It can be seen that the numerical result computed by the proposed model matches well with both the analytical solution and numerical result of Singh et al. (2015), verifying the accuracy of the proposed model.

5.2 Rainfall−runoff process over a three-plane cascade surface

This example proposed by Iwagaki (1955) is used to validate the appropriateness of the present model for overland flow simulation by comparison with experimental data. In this experiment, a series of artificial rainfall was applied on an aluminum laboratory flume with a three-plane cascade surface. The flume is 24 m in length and is equally divided into three planes with constant slopes of 0.020, 0.015, and 0.010, respectively, from upstream to downstream. The constant excess rainfall intensities of 389, 230, and 288 cm/h were applied to the up-, middle-, and downstream of the flume, respectively, for the durations of 10, 20, and 30 s in three experiments.

The three cases are simulated using 1126 triangular meshes ($\xi = 0.5$ m). The Manning's n is set to 0.007 (Huang 2006). Solid slip conditions are imposed at the upstream and lateral walls of the flume, while the downstream boundary is a free outlet. The bed is initially dry. Figures 5, 6, and 7 present the comparison of the computed hydrograph at the outlet and experimental data for the three cases, respectively. It can be seen that the

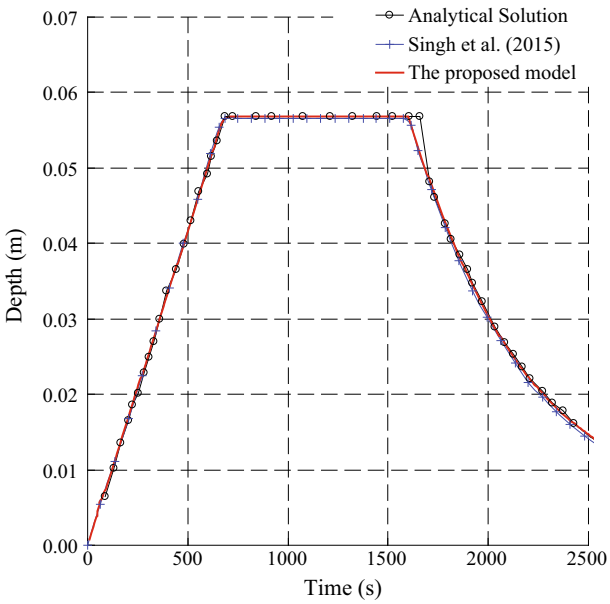

Fig. 4 Comparison of analytical solution and simulated flow depth at outlet for Case 2

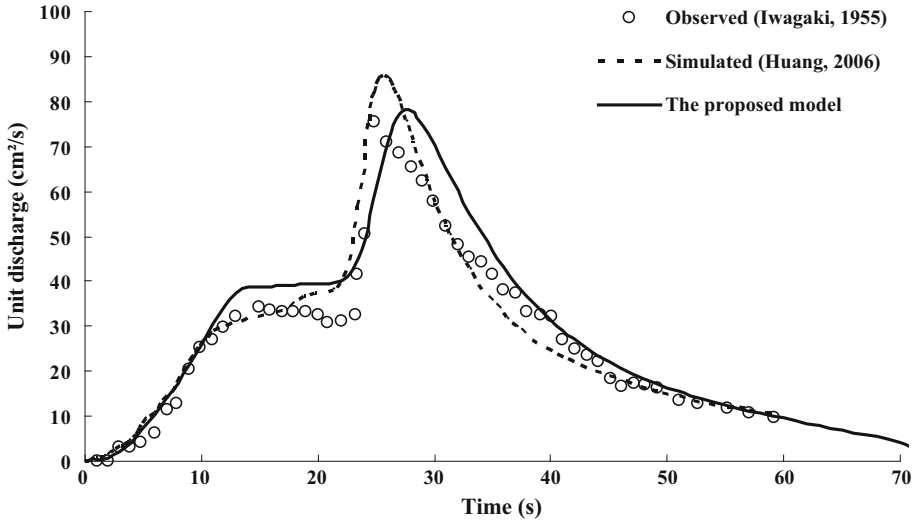

Fig. 5 Computed and observed discharge hydrograph for the three-plane cascade with rainfall duration of 10 s

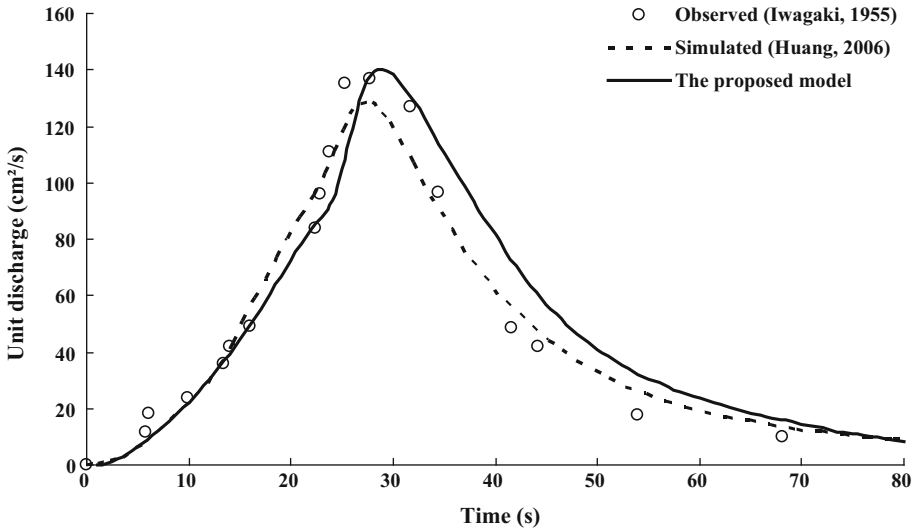

Fig. 6 Computed and observed discharge hydrograph for the three-plane cascade with rainfall duration of 20 s

numerical results computed by the proposed model match well with the experimental data, verifying the accuracy of the proposed model. Furthermore, numerical result of short rainfall duration (10 s) shows that kinematic shock wave generated by the changes in bed slope is simulated successfully without numerical instabilities and oscillations. This shock capture capability is very important for numerical models to simulate the overland flow with wet/dry fronts on highly irregular topography.

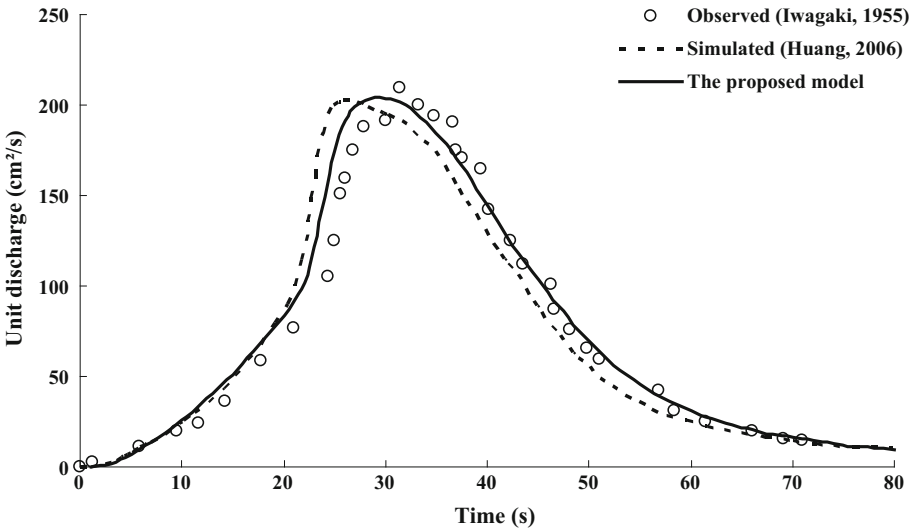

Fig. 7 Computed and observed discharge hydrograph for the three-plane cascade with rainfall duration of 30 s

5.3 Case study and model performance testing

5.3.1 Study area

The mountain watershed is located in the Wangmo County, Guizhou Province, China. Rivers in the mountain watershed belong to the Beipan River and Hongshui River system of the Pearl River basin. The length of the mountain watershed is 14 km from east to west and 24 km from north to south, giving a total drainage area of 198.6 km^2. Figure 8 presents the sketch map of 3D terrain, from which it can be seen that the topography is highly steep and irregular. The altitude varies approximately from 600 to 1600 m, and the average slope of the mainstream named Wangmo River is 2.73%. Besides, the average slopes of tributaries vary from 2.95 to 25.93%. As a result of complex topographical condition, the rainfall−runoff process related to the mountain watersheds response is rapid. Since the rivers are narrow, steep, and short in length, flash floods due to excessive rainfall are the most serious threatening environment hazard in this mountain watershed.

5.3.2 Computational grid

In this work, unstructured, triangular grid is adopted to facilitate grid generation and localized refinement for modeling natural waterways in mountain watershed. To ensure the precision of the calculations, the whole drainage area of the Wangmo River is triangulated to computational grids. Three grid division schemes with different resolution are considered in this paper. The first division is a base grid (see Fig. 9) with average side-lengths varying from 35 to 100 m. The other two divisions are refined in turn by averagely dividing one grid to four grids from the middle of the edges. So the corresponding total numbers of triangular mesh cells are 85504, 342016, and 1368064, respectively. For simplified

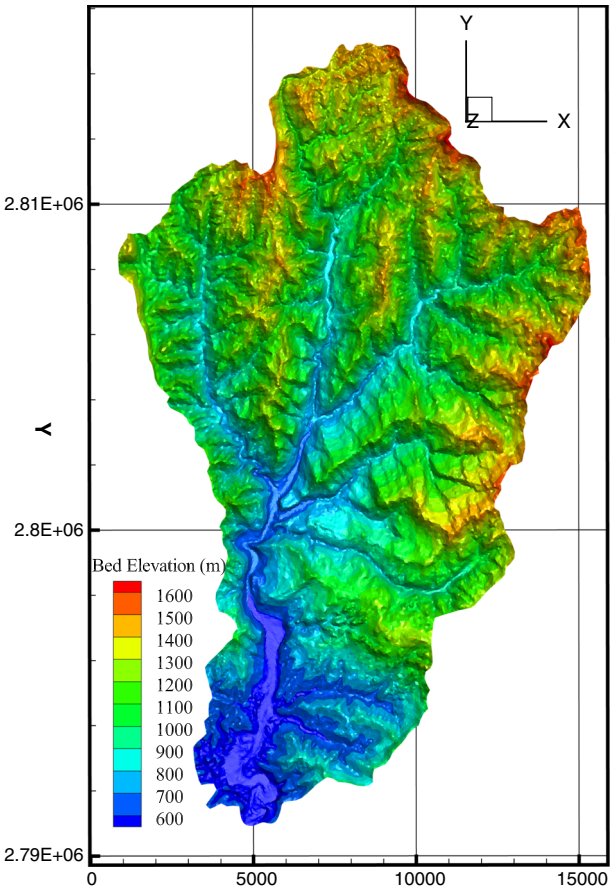

Fig. 8 The sketch map of 3D terrain

expression, the three grid divisions are denoted as mesh-A, mesh-B, and mesh-C, respectively.

5.3.3 Verification of mass conservation

Since the property of mass conservation is very important for flash floods simulation on highly irregular topography, a hypothetical constant excess rainfall intensity of 100 mm/h is considered in this work. The Manning's n is set to 0.05. Solid slip conditions are imposed at boundaries except that the downstream boundary of the Wangmo River is a free outlet. The bed is initially dry. As the rainfall intensity is constant, the outlet discharge would be gradually rose to a steady state with the exact value of 5517 m^3/s, which is calculated from 198.6 $km^2 \times 100$ mm/h. We run the simulation until $t = 10$ h. Figure 10 shows the computed discharge hydrographs at the outlet. It can be seen that the computed steady outlet discharges are 5514, 5516, and 5517 m^3/s for mesh-A, mesh-B, and mesh-C, respectively, which match well with the exact value. Besides, a faster runoff concentration in the flood rise period is simulated for finer grid resolution. This is partially due to the water detention effect of local depressions. As the topography is highly irregular and steep, finer grids have a smaller height deviation for cells as well as higher resolution in space for

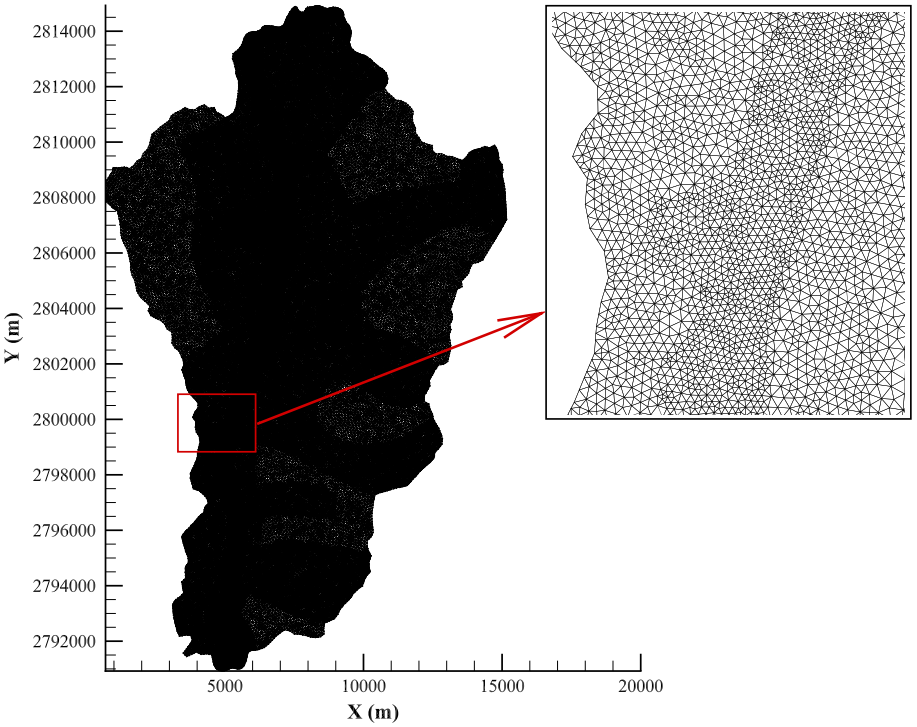

Fig. 9 The computational grid of mountain watershed

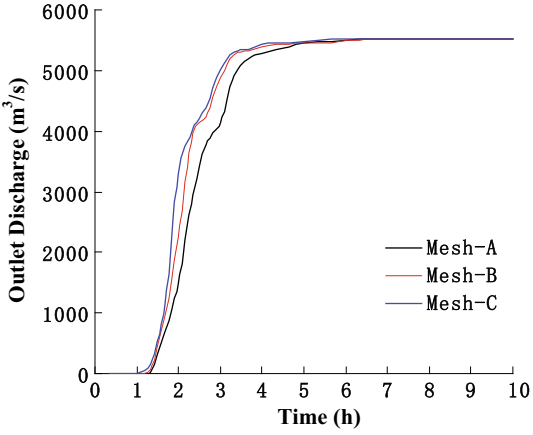

Fig. 10 The computed discharge hydrographs at the outlet

modeling narrow waterways. So water detention effect might be smaller for finer gird resolution (See Fig. 11), resulting in a faster runoff concentration. On the other hand, numerical accuracy is related to spatial discretization resolution, and finer grids would reduce the numerical dissipation. Nevertheless, the computed discharge hydrographs at the outlet are basically identical for the three grid divisions. Figure 12 shows the hydrographs of accumulated water volume: As it can be seen, the accumulated water volume from

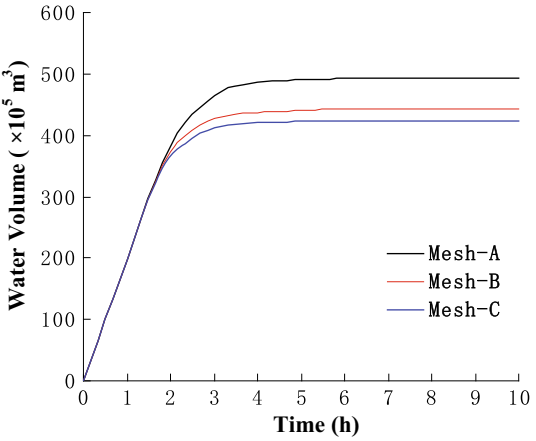

Fig. 11 The accumulated water volume in cells

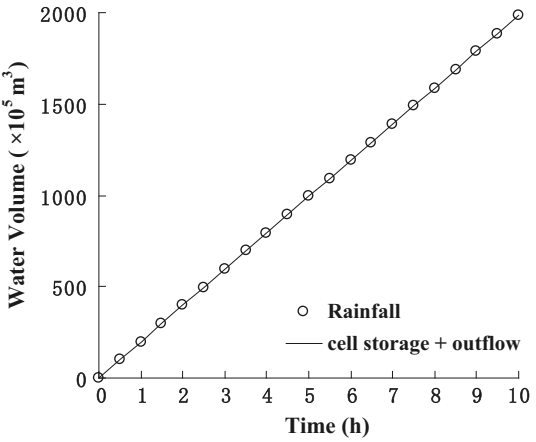

Fig. 12 The hydrographs of accumulated water volume of rainfall and simulated results

rainfall is equal to the sum of that of cell storage and outflow, i.e., mass conservation is preserved well. Note that only the result of mesh-A is drawn in Fig. 12 for clear expression, as other results of mesh-B and mesh-C are similar.

5.3.4 Real flash flood simulation

A heavy rain was occurred in June 5, 2011, and has caused huge economic losses to society. Figure 13 presents the rainfall process, from which we can see that the 3-h maximum rainfall is up to 257 mm. In general, reservoir in the mountain watershed should be modeled as inner boundary. The flood discharge of reservoir is controlled by the operation rules and should be computed based on the discharging curve. Besides, infiltration rate should be determined, for example, using the popular Green–Ampt equation. Furthermore, the underground river should be considered here because of the Karst topography. Together these factors make the fine simulation is very difficult, mostly due to the limitation of data. In this work, to focus on the speedup ratio of GPU computing, the effect of these factors is generalized by a coefficient of rainfall loss, which is calibrated to 0.62 based on the mesh-B. The comparisons of surveyed and computed peak discharges, as

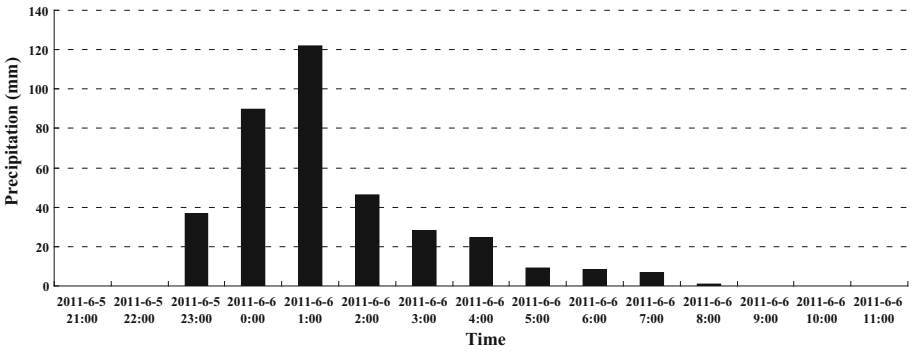

Fig. 13 The rainfall process

well as the maximum water levels at Wangmo city square, are presented in Table 1 and illustrate that model predictions compare well with surveyed data. So it could be concluded that the coefficient of rainfall loss is reasonable. Besides, result errors of mesh-A are much bigger than those of mesh-B, indicating a moderate grid resolution is necessary for flash flood simulation. Using coarse grids would result in illusive depressions and then decrease the magnitude of floods. To improve the prediction accuracy, high-resolution grids should be used at the catchment scale. This puts forward quite high demand for the computational efficiency. Figure 14 presents the distribution of maximum water depth computed on mesh-C, which is one of the major types of flood risk map.

5.3.5 Parallel performance analysis

In this work, we use Intel Xeon(R) CPU E5-2609 v2 @ 2.50 GHz for running the serial model, while additionally use a NVIDIA Tesla K20c card for parallel model. This card has a Kepler GK110 GPU and 2496 NVIDIA CUDA cores, and the clock frequency is 706 MHz, delivering up to 1.17 TFLOPS of peak double-precision floating point performance for intense compute workloads. The real flash flood simulation schemes presented in Sect. 5.3.4 is used for parallel performance analysis. Because the calculation time of serial model for the entire rainfall−runoff process is too long, we studied the first 9 h of rainfall (i.e., 2011/6/5 23:00 ∼ 2011/6/6 8:00) to compare model performance.

To analysis parallel performance, a speedup is defined as (Zhang et al. 2017):

Table 1 The comparison of surveyed and computed values at Wangmo city square

	Observed	Mesh-A	Mesh-B	Mesh-C
Discharge (m^3/s)	1340	1112	1337	1425
Discharge relative error (%)	–	− 17.0%	− 0.2%	6.3%
Water level (m)	546.44	546.13	546.43	546.60
Water level error (m)	–	− 0.31	− 0.01	0.16

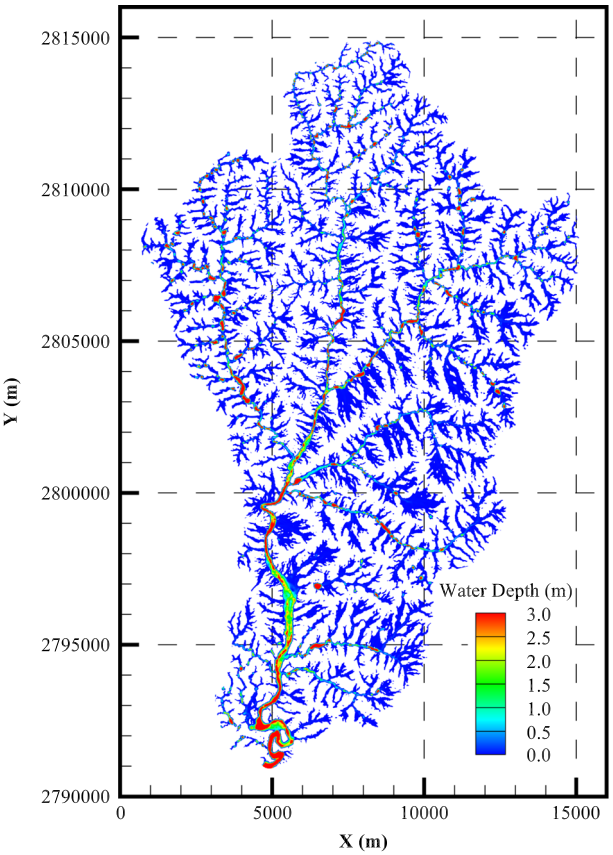

Fig. 14 The distribution of maximum water depth

$$S = T_{\mathrm{CPU}}/T_{\mathrm{GPU}} \tag{7}$$

where S denotes the speedup ratio; T_{CPU} and T_{GPU} denote the run time of the serial model and the parallel model, respectively.

The time saving ratio is defined as follows (Zhang et al. 2017):

$$TS = (T_{\mathrm{CPU}} - T_{\mathrm{GPU}})/T_{\mathrm{CPU}} \tag{8}$$

Results including speedup and time saving ratio of calculation on different mesh divisions are presented in Table 2. We can see that all three mesh divisions attained a

Table 2 Parallel results of different calculation schemes

	Grid number	Average grid area (m^2)	T_{CPU} (h)	T_{GPU} (h)	S	TS (%)
Mesh-A	85504	2323	4.45	0.31	14.4	93
Mesh-B	342016	581	24.37	1.03	23.7	96
Mesh-C	1368064	145	103.24	3.37	30.6	97

dramatic reduction in running time. For the mesh-A of 85504 grids with average side-lengths varying from 35 to 100 m, a 9-h rainfall−runoff process took approximately 4.45 h to calculate with the serial model, whereas only 0.31 h was required with a parallel model on the NVIDIA Tesla K20c card. For the mesh-C of 1368064 grids with average side-lengths varying from 9 m to 25 m, the running time requirement is approximately 103.24 h (4.3 days) and 3.37 h for the serial model and parallel model, respectively. The simulations using mesh-A, mesh-B, and mesh-C are nearly 29, 9, and 3 times faster than real time, respectively. Generally, mesh refinement should be used to improve prediction precision, while ineluctability is limited by the demand on calculation speed. So it can be concluded that the tradeoff between numerical accuracy and computational efficiency can be relaxed by using high-performance GPU computing. Furthermore, the speedup ratio increases with the number of grid cells. In this work, a speedup ratio of 14.4 was achieved with the mesh-A of 85504 grids, whereas a speedup ratio of 30.6 was achieved with the mesh-C of 1368064 grids. The speedup ratio could be further increased if multi-GPU cards are used for calculation.

6 Conclusions

A two-dimensional full hydrodynamic model is presented for the simulation of flash floods in mountain watersheds. The new formulation of the shallow water equations and sloping bottom model are adopted to construct a robust finite volume scheme, which can simulate the overland flow with wet/dry fronts on highly irregular topography. Since the finite volume scheme is explicit, there are many computationally intensive loop structures without data dependence, i.e., the calculations for a cell or edge are uncorrelated with other cells or edges in the current time step. So the calculation loops are natural parallelizable and can be executed in GPU-based parallel model.

The OpenACC programming model, which parallelizes loops automatically by using directives and clauses as well as maps the parallelism with auto-optimal parameters of gangs-number, workers-number, and vector-length, is adopted to realize high-performance GPU computing. So an incremental developing way with minimal recoding work is achieved, and the parallel model is highly portable with all types of GPU devices. To minimize the time-consuming data transportation, the parallelization of the flash flood model is designed to three steps: data transportation from CPU to GPU, numerical calculation in GPU, and data transportation from GPU to CPU. So all computationally intensive loop structures and the corresponding data are parallel handled in the GPU device, and the CPU device is only used to control the calculation process with data inputs and outputs.

Model accuracy is validated by benchmark cases with exact solutions and experimental data. To further analyze the performance of the model, the Wangmo River basin of 198.6 km^2 drainage area is triangulated to 85504, 342016, and 1368064 grids, respectively. A hypothetical constant excess rainfall intensity of 100 mm/h is used for verification of mass conservation. Results show that the computed steady outlet discharges match well with the exact value, and a faster runoff concentration in the flood rise period is simulated for finer grid resolution. This is partially due to the water detention effect of local depressions. As the topography is highly irregular and steep, finer grids have a smaller height deviation for cells as well as higher resolution in space for modeling narrow waterways. So water detention effect might be smaller for finer gird resolution.

Furthermore, a historical flash flood occurred in June 5, 2011, was reproduced by the proposed model, and model predictions compare well with surveyed data.

Parallel performance is analyzed by comparing the running time of serial model and parallel model. The Intel Xeon(R) CPU E5-2609 v2 @ 2.50 GHz is used for running the serial model, while additionally a NVIDIA Tesla K20c card is used for parallel model. A historical rainfall−runoff process of 9 h (i.e., 2011/6/5 23:00 ~ 2011/6/6 8:00) is used to analysis parallel performance. Results show that all three mesh divisions attained a dramatic reduction in running time by using parallel model. For the mesh-A of 85504 grids with average side-lengths varying from 35 to 100 m, a 9-h rainfall−runoff process took approximately 4.45 h to calculate with the serial model, whereas only 0.31 h was required with the parallel model on the NVIDIA Tesla K20c card. For the mesh-C of 1368064 grids with average side-lengths varying from 9 m to 25 m, the running time requirement is approximately 103.24 h (4.3 days) and 3.37 h for the serial model and parallel model, respectively. In other words, a speedup ratio of 14.4 was achieved with the 85504 grids, whereas a speedup ratio of 30.6 was achieved with the 1368064 grids, indicating that a higher speedup ratio can be achieved for finer grids resolution. So it can be concluded that the proposed parallel model can be used for real-time prediction of large-scale flash flood on high-resolution grids and thus has bright application prospects.

Acknowledgements This work was supported by a grant from the Natural Science Foundation of Guangdong Province, China (No. 2014A030310283), a grant from the National Key Research and Development Program of China (No. 2017YFC0405900), a grant from the Open Research Foundation of PRHRI (Project No. 2013KJ01), and a grant from the Special Research Foundation for the Public Welfare Industry of the Ministry of Water Resources (No. 201501030).

References

Begnudelli L, Sanders BF (2006) Unstructured grid finite-volume algorithm for shallow-water flow and scalar transport with wetting and drying. ASCE J Hydraul Eng 132(4):371–384

Begnudelli L, Sanders BF (2007) Conservative wetting and drying methodology for quadrilateral grid finite-volume models. ASCE J Hydraul Eng 133(3):312–322

Elfeki A, Masoud M, Niyazi B (2017) Integrated rainfall–runoff and flood inundation modeling for flash flood risk assessment under data scarcity in arid regions: Wadi Fatimah basin case study, Saudi Arabia. Nat Hazards 85:87–109

Giammarco PD, Tadini E, Lamberti P (1996) A conservative finite element approach to overland flow: the control volume finite element formulation. J Hydrol 175(1–4):267–291

Guinot V, Sanders BF, Schubert JE (2017) Dual integral porosity shallow water model for urban flood modeling. Adv Water Resour 103:16–31

Herdman JA, Gaudin WP, Mcintosh-Smith S, et al (2012) Accelerating hydrocodes with OpenACC, OpenCL and CUDA. In: SC companion: high performance computing, networking, storage and analysis. IEEE computer society, pp 465–471

Huang G (2006) Physics based, integrated modeling of hydrology and hydraulics at watershed scales. PhD thesis, The Pennsylvania State University

Hubbard ME (1999) Multidimensional slope limiters for MUSCL-type finite volume schemes on unstructured grids. J Comput Phys 155(1):54–74

Iwagaki Y (1955) Fundamental studies on the runoff by characteristics. Bull Disaster Prev Res Inst Kyoto Univ 10:1–25

Kim B, Sanders BF, Schubert JE et al (2014) Mesh type tradeoffs in 2D hydrodynamic modeling of flooding with a Godunov-based flow solver. Adv Water Resour 68:42–61

Kourgialas NN, Karatzas GP, Nikolaidis NP (2012) Development of a thresholds approach for real-time flash flood prediction in complex geomorphological river basins. Hydrol Process 26:1478–1494

Lai W, Khan AA (2016) A parallel two-dimensional discontinuous galerkin method for shallow-water flows using high-resolution unstructured meshes. J Comput Civil Eng 31(3):04016073

Lian J, Yang W, Xu K et al (2017) Flash flood vulnerability assessment for small catchments with a material flow approach. Nat Hazards 88:699–719
Liang Q, Borthwick AGL (2009) Adaptive quadtree simulation of shallow flows with wet-dry fronts over complex topography. Comput Fluids 38(2):221–234
Liang Q, Xia X, Hou J (2016) Catchment-scale high-resolution flash flood simulation using the GPU-based technology. Procedia Eng 154:975–981
Liu W, Chen W, Hsu M et al (2010) Dynamic routing modeling for flash flood forecast in river system. Nat Hazards 52:519–537
Pender G, Cao Z, Zhang S et al (2010) Hydrodynamic modelling in support of flash flood warning. Water Manag 163(7):327–340
Sanders BF, Schubert JE, Detwiler RL (2010) ParBreZo: a parallel, unstructured grid, Godunov-type, shallow-water code for high-resolution flood inundation modeling at the regional scale. Adv Water Resour 33(12):1456–1467
Singh VP (1996) Kinematic wave modeling in water resources: surface-water hydrology. Wiley, New York
Singh J, Altinakar MS, Ding Y (2015) Numerical modeling of rainfall-generated overland flow using nonlinear shallow-water equations. ASCE J Hydraul Eng. https://doi.org/10.1061/(ASCE)HE.1943-5584.0001124
Song L, Zhou J, Guo J et al (2011) A robust well-balanced finite volume model for shallow water flows with wetting and drying over irregular terrain. Adv Water Resour 34(7):915–932
Tao J, Barros AP (2013) Prospects for flash flood forecasting in mountainous regions-An investigation of Tropical Storm Fay in the Southern Appalachians. J Hydrol 506:69–89
Toro EF (2001) Shock-capturing methods for free-surface shallow flows. Wiley, Chichester. ISBN 0-471-98766-2
Tsai TL, Yang JC (2005) Kinematic wave modeling of overland flow using characteristics method with cubic-spline interpolation. Adv Water Resour 28(7):661–670
Wang X, Shangguan Y, Onodera N et al (2014) Direct numerical simulation and large eddy simulation on a turbulent wall-bounded flow using lattice Boltzmann method and multiple GPUs. Math Probl Eng 2014:742432
Yang L, Smith JA, Baeck ML et al (2016) Flash flooding in small urban watersheds: storm event hydrologic response. Water Resour Res 52:4571–4589
Zeng Z, Tang G, Long D et al (2016) A cascading flash flood guidance system: development and application in Yunnan Province, China. Nat Hazards 84:2071–2093
Zhang S, Yuan R, Wu Y et al (2016) Implementation and efficiency analysis of parallel computation using OpenACC: a case study using flow field simulations. Int J Comput Fluid Dyn 30(1):79–88
Zhang S, Yuan R, Wu Y et al (2017) Parallel computation of a dam-break flow model using OpenACC applications. J Hydraul Eng 143(1):04016070