INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2015; **77**:381–399
Published online 18 December 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/fld.3988

Solving the Savage–Hutter equations for granular avalanche flows with a second-order Godunov type method on GPU

Jian Zhai[1], Li Yuan[1,*,†], Wei Liu[1] and Xinting Zhang[2]

[1]*LSEC & NCMIS, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China*
[2]*École Centrale de Pékin, Beihang University, Beijing 100191, China*

SUMMARY

In this article, we apply Davis's second-order predictor-corrector Godunov type method to numerical solution of the Savage–Hutter equations for modeling granular avalanche flows. The method uses monotone upstream-centered schemes for conservation laws (MUSCL) reconstruction for conservative variables and Harten–Lax–van Leer contact (HLLC) scheme for numerical fluxes. Static resistance conditions and stopping criteria are incorporated into the algorithm. The computation is implemented on graphics processing unit (GPU) by using compute unified device architecture programming model. A practice of allocating memory for two-dimensional array in GPU is given and computational efficiency of two-dimensional memory allocation is compared with one-dimensional memory allocation. The effectiveness of the present simulation model is verified through several typical numerical examples. Numerical tests show that significant speedups of the GPU program over the CPU serial version can be obtained, and Davis's method in conjunction with MUSCL and HLLC schemes is accurate and robust for simulating granular avalanche flows with shock waves. As an application example, a case with a teardrop-shaped hydraulic jump in Johnson and Gray's granular jet experiment is reproduced by using specific friction coefficients given in the literature.

Received 8 April 2014; Revised 6 November 2014; Accepted 20 November 2014

KEY WORDS: Savage–Hutter equation; stopping criterion; HLLC scheme; MUSCL reconstruction; GPU computing

1. INTRODUCTION

Landslides, rock avalanches, and debris flows are natural phenomena, which frequently occur in mountainous areas. They may cause great loss of properties and lives and have been extensively studied by many researchers in various disciplines. However, these phenomena are multiphase, polydisperse, multiscale, erosive, and rheologically complicated, entailing high complexity to scientific studies. It is well known that the basic composition materials in the aforementioned geomorphological phenomena are granular materials. Granular materials are a collection of a large number of discrete solid particles with interstices filled with one or more fluids [1]. Motions of a plenty of granular materials make up granular flows, which are a continuum treatment used to describe real avalanches and debris flows. The granular materials driven by the gravity slide and accelerate down the slope, forming granular avalanche flows that generally have small thicknesses but can travel very long distances.

At the present stage, the main models for granular avalanche flows are typified by shallow granular flow models such as the Savage–Hutter (SH) model [2], the Iverson–Denlinger two-phase

*Correspondence to: Li Yuan, Academy of Mathematics and Systems Science, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences, Beijing 100190, China.
†E-mail: lyuan@lsec.cc.ac.cn

mixture model [3, 4], and the two-phase thin layer model proposed by Pitman [5], to name a few. The derivation of shallow flow models utilizes the depth average approach and their canonical equations are often written in a Cartesian coordinate system with the z axis normal (or approximately normal) to the basal surface. The canonical formulations [2, 3, 5] are not suitable for numerical calculation of avalanche flows over general topography. To simulate granular avalanche flows on general three-dimensional (3D) terrains, the SH model can be formulated in a general curvilinear coordinate system with x, y axes tangential to and z axis normal to the basal surface, but such forms contain many non-conservative metric terms and may lead to computational complexity. Bouchut and Westdickenberg [6] derived a shallow water models in which the coordinate system is a global Cartesian system, but the solution variables are the flow depth in the direction normal to the topography and the velocity tangent to the topography. However, the pressure term can not be written in conservative form for variable bed slope angles. Denlinger and Iverson [7] reformulated the depth-averaged governing equations in a global Cartesian coordinate system with z vertical and accounted explicitly for the effect of non-zero vertical accelerations on depth-averaged mass and momentum fluxes and stress states. Their equations provide a hope of using global Cartesian formulation to describe shallow granular flows over general 3D terrains. In this paper, we focus on numerical method for the SH model formulated in a special global curvilinear coordinate system that is straight in the transverse direction [8, 9]. This formalism has the same conservative terms as in the Cartesian formulation. We also consider the special difficulty in simulating granular avalanche flows, that is, the yielding/stopping properties of granular materials [10–13]. Appropriate static resistance condition and stopping criterion must be taken account into the solution procedure in order to model the yielding/stopping properties.

As depth averaged shallow flow models are in most situations hyperbolic equations that allow for discontinuous solutions, some well-known Godunov type methods such as Harten–Lax–van Leer (HLL), HLL contact (HLLC) (c.f. [14]), and wave propagation schemes have been used in the literature [3, 7, 15]. Other classes of high-resolution shock-capturing methods such as non-oscillatory central schemes [16] were also employed [8, 9]. Wang *et al*. [8] demonstrated that the non-oscillatory central scheme with Minmod limiter or van Leer limiter was very suitable to handle the problem of avalanche dynamics. Patra *et al*. [15] employed Davis's second-order predictor-corrector Godunov type method [17] with van Leer's monotone upstream-centered schemes for conservation laws (MUSCL) reconstruction (c.f. [14]) and HLL scheme to solve the Denlinger–Iverson formulation [4, 7]. However, the static resistance condition and the stopping criterion are missing in their numerical procedure. In this paper, we add the static resistance condition and the stopping criterion to Davis's Godunov type method to simulate the yielding/stopping properties of granular avalanche flows. Furthermore, we implement graphics processing unit (GPU) computing and test the correctness and efficiency of the resulting code in several numerical examples.

The SH model is 2D and its numerical solution time is generally not long. However, for engineering design and hazard risk analysis, a large number of simulations with high grid resolutions must be completed in short times, thus necessitating parallel computing. With the availability and development of NVIDIA's compute unified device architecture (CUDA, for introduction, see [18]) as a programming model, GPU computing has become one of the most popular choices for parallel computing because of GPU's massive parallel processing power and low energy consumption per Gflops. CUDA is based on C language, thus making nowadays general users learn easily. Owing to its character of 'many-core' and faster memory access, a GPU can speedup massively data-parallel codes many times relative to a central processing unit (CPU) of the same generation. Furthermore, performance benefits can scale with multiple GPUs using CUDA 4.0 above, OpenMP, or Pthread on a shared-memory computer installed with several GPUs. There are numerous work on applying GPU computing for computational fluid dynamics. Here, we only listed some work related to shallow flow models [19–25]. Kuo *et al.* [19] presented application of the split HLL scheme for both the Euler and shallow water equations to Tesla C1060 GPU (first generation general purpose GPU) using CUDA and reported over 67 times speedup over an Intel Xeon X5472 CPU core (3.0 GHz). Brodtkorb *et al.* [20] implemented the second-order Kurganov-Petrova's central scheme for the shallow water equations on NVIDIA GeForce GTX 480 graphics card and combined the simulation with OpenGL visualization on the same card. Sætra and Brodtkorb *et al.* [21, 22]

DOI: 10.1002/fld

presented detailed multiple-GPU implementation of a block-based AMR method for the same central scheme and a series of performance and accuracy tests. In Ref. [21], they used the row decomposition of computational domain to put the transferred data in continuous memory and the control thread-work thread strategy to enable multi-GPU computations in a single node and performed traditional CUDA block decomposition within each GPU for further parallelism. Their implementation shows near perfect weak and strong scaling and enables simulation of up to 235 million cells at a rate of over 1.2 gigacells per second using four Fermi GPUs (second generation). Vinas *et al.* [23] also presented multi-GPU implementations of a Roe type finite volume method for shallow water equations with contamination transport using MPI where the fastest example reached a speedup of 78× using four M2050 GPUs on an Infiniband network with respect to a parallel execution on a six-core CPU. Smith and Liang [24] implemented a Godunov-type MUSCL–Hancock scheme for shallow water equations on Fermi M2075 GPU and significantly expedited the simulations when compared to a traditional CPU approach. They also pointed out that there are significant localized errors introduced by 32-bit floating-point precision. Vacondio *et al.* [25] parallelized a well-balanced positive depth reconstruction finite volume explicit discretization technique on GTX 580 and M2070 GPUs for fast flood simulations. In that work, they adopted a novel and efficient block deactivation optimization procedure to increase the efficiency of numerical solution in the presence of wetting–drying fronts.

While many researchers have applied GPU computing to the shallow water equations for modeling flood events and obtained good performance as mentioned earlier, there are few similar applications to granular avalanche flows. Ref. [22] remarked that many techniques used in GPU implementation for the shallow water equations are transferable to other hyperbolic conservation laws and numerical schemes. Therefore, it is worthwhile to apply GPU computing to the SH equations for simulating granular avalanche flows. In this work, we use CUDA and a single GPU to implement the numerical algorithm for the SH model.

The paper is organized as follows. In Section 2, the SH model is formulated. In Section 3, Davis's version of the Godunov type method together with the MUSCL reconstruction and HLLC scheme is given, and the static resistance condition and stopping criterion are added to the numerical procedure. In Section 4, a short introduction of CUDA implementation used is presented followed by a practice on allocating global memory on GPU for 2D array for ease of code porting. In Section 5, three numerical examples are used to verify the correctness of the numerical model and test the performance of the GPU and CPU programs, and one granular jet example is used to demonstrate the application of the developed code. The last section gives the conclusion.

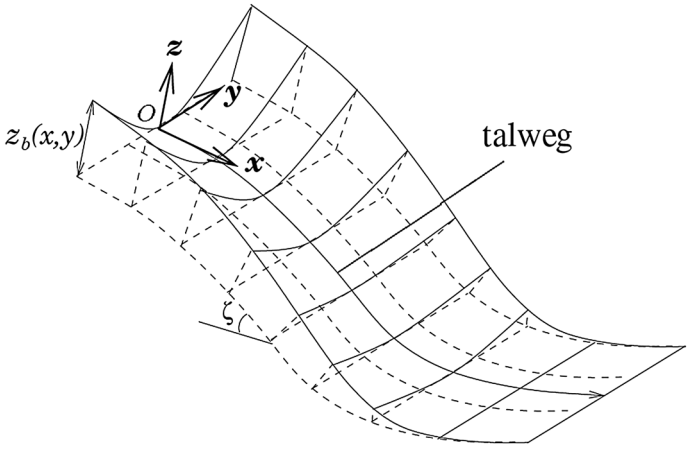


Figure 1. The curvilinear coordinates $Oxyz$ is defined on the reference surface (dashed lines) where the downslope inclination angle of the reference surface ζ is a function of the downslope coordinate x (reproduced from [8]).

DOI: 10.1002/fld

2. GOVERNING EQUATIONS

In 1989, Savage and Hutter first derived the SH equations [2] from the incompressible Navier–Stokes equations by using the depth-averaged approach. In the SH model, the granular material is regarded as cohesionless Coulomb material, that is, the shear stress τ only depends on the normal stress σ: $\tau = \sigma \tan\phi$, where ϕ is the internal friction angle of the material. The system of the equations is of hyperbolic type akin to the shallow water equations. There are various forms of SH equations according to different coordinate systems chosen. Here, we use the same special global curvilinear coordinate system given in Ref. [8] as shown in Figure 1, where there is no transverse variation for the inclination angle ζ in the y direction. The corresponding non-dimensional SH equations are

$$\frac{\partial h}{\partial t} + \frac{\partial}{\partial x}(hu) + \frac{\partial}{\partial y}(hv) = 0, \tag{1}$$

$$\frac{\partial}{\partial t}(hu) + \frac{\partial}{\partial x}\left(hu^2 + \frac{\beta_x h^2}{2}\right) + \frac{\partial}{\partial y}(huv) = hs_x, \tag{2}$$

$$\frac{\partial}{\partial t}(hv) + \frac{\partial}{\partial x}(huv) + \frac{\partial}{\partial y}\left(hv^2 + \frac{\beta_y h^2}{2}\right) = hs_y, \tag{3}$$

where h is the flow thickness in the z direction, and u and v are the depth-averaged velocity components in the x and y directions, respectively. β_x and β_y are defined as

$$\beta_x = \varepsilon \cos\zeta K_x, \quad \beta_y = \varepsilon \cos\zeta K_y, \tag{4}$$

where K_x and K_y are the earth pressure coefficients in the x and y directions according to the Mohr–Coulomb yield criterion, and ε is aspect ratio of the characteristic thickness to the characteristic downslope extent. The terms s_x and s_y represent the net driving accelerations in the x and y directions, respectively,

$$s_x = \sin\zeta - \frac{u}{|\mathbf{v}|}\tan\delta \cdot \max\left(0, \left(\cos\zeta + \lambda\kappa u^2\right)\right) - \varepsilon\cos\zeta\frac{\partial z_b}{\partial x}, \tag{5}$$

$$s_y = -\frac{v}{|\mathbf{v}|}\tan\delta \cdot \max\left(0, \left(\cos\zeta + \lambda\kappa u^2\right)\right) - \varepsilon\cos\zeta\frac{\partial z_b}{\partial y}, \tag{6}$$

where $|\mathbf{v}| = \sqrt{u^2 + v^2}$, δ is the basal Coulomb friction angle, $\kappa = -\dfrac{\partial\zeta}{\partial x}$ is the local curvature of the reference surface, and $\lambda\kappa$ is the local stretching of the curvature. The shallow basal topography is defined by its elevation $z = z_b(x, y)$ above the curvilinear reference surface as shown in Figure 1. The earth pressure coefficients are given according to [26]:

$$K_{x_{\text{act/pass}}} = 2\left(1 \mp \sqrt{1 - \frac{\cos^2\phi}{\cos^2\delta}}\right)\sec^2\phi - 1, \tag{7}$$

$$K_{y_{\text{act/pass}}} = \frac{1}{2}\left(K_x + 1 \mp \sqrt{(K_x - 1)^2 + 4\tan^2\delta}\right), \tag{8}$$

DOI: 10.1002/fld

where ϕ and δ are the internal and bed friction angles respectively. The subscripts 'act' and 'pass' denote active ('−') and passive ('+') stress states:

$$K_x = \begin{cases} K_{x_{\text{act}}}, & \dfrac{\partial u}{\partial x} \geqslant 0 \\ K_{x_{\text{pass}}}, & \dfrac{\partial u}{\partial x} < 0 \end{cases}, \tag{9}$$

$$K_y = \begin{cases} K_{y_{\text{act}}}, & \dfrac{\partial v}{\partial y} \geqslant 0 \\ K_{y_{\text{pass}}}, & \dfrac{\partial v}{\partial y} < 0 \end{cases}. \tag{10}$$

3. NUMERICAL METHODS

3.1. Second-order Godunov type method

Similar to Ref. [15], a second-order predictor-corrector Godunov type method due to Davis [17] with the MUSCL reconstruction for the conservative variables $\mathbf{U} = (h, hu, hv)^T$ and HLLC numerical flux [14] is used. Define the cell average:

$$\mathbf{U}_{i,j} = \frac{1}{\Delta x \Delta y} \iint_{\Omega_{ij}} \mathbf{U} \mathrm{d}x \mathrm{d}y. \tag{11}$$

The method achieves second-order accuracy in time and space by using second-order predictor-corrector scheme and MUSCL reconstruction. To proceed, rewrite the system (1–3) as

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A}\frac{\partial \mathbf{U}}{\partial x} + \mathbf{B}\frac{\partial \mathbf{U}}{\partial y} = \mathbf{S}(\mathbf{U}), \tag{12}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the Jacobian matrices of the physical fluxes $\mathbf{F}$ and $\mathbf{G}$, respectively:

$$\mathbf{A} = \frac{\partial \mathbf{F}}{\partial \mathbf{U}} = \begin{bmatrix} 0 & 1 & 0 \\ -u^2 + \beta_x h & 2u & 0 \\ -uv & v & u \end{bmatrix}, \quad \mathbf{B} = \frac{\partial \mathbf{G}}{\partial \mathbf{U}} = \begin{bmatrix} 0 & 0 & 1 \\ -uv & v & u \\ -v^2 + \beta_y h & 0 & 2v \end{bmatrix}. \tag{13}$$

The procedure of the predictor-corrector Godunov type method goes as follows.

Step 1: Predictor step. Compute the solution variables at the half time level $n + \frac{1}{2}$ by using Taylor expansion and Equation (12) (Lax–Wendroff approach):

$$\mathbf{U}_{i,j}^{n+\frac{1}{2}} = \mathbf{U}_{i,j}^{n} - \frac{\Delta t}{2}\left[\mathbf{A}_{i,j}^{n}\left(\boldsymbol{\sigma}_{i,j}^{x}\right)^{n} + \mathbf{B}_{i,j}^{n}\left(\boldsymbol{\sigma}_{i,j}^{y}\right)^{n} - \mathbf{S}_{i,j}^{n}\right], \tag{14}$$

where $\boldsymbol{\sigma}_{i,j}^{x}$ and $\boldsymbol{\sigma}_{i,j}^{y}$ are the limited slopes of $\mathbf{U}$ in the x and y directions, respectively. The limiter here could use the Minmod limiter

$$\boldsymbol{\sigma}_{i,j}^{x} = \frac{\boldsymbol{\phi}_{i,j}^{x}}{\Delta x}, \qquad \boldsymbol{\phi}_{i,j}^{x} = \text{Minmod}(\mathbf{U}_{i,j} - \mathbf{U}_{i-1,j}, \mathbf{U}_{i+1,j} - \mathbf{U}_{i,j}). \tag{15}$$

Step 2: Data reconstruction. The left state $\mathbf{U}_{i+1/2}^{L}$ and the right state of $\mathbf{U}_{i+1/2}^{R}$ at the cell interface $i + 1/2$ in the x direction are constructed using the piece-wise linear MUSCL reconstruction for the conservative variables obtained in Step 1:

$$\mathbf{U}_{i+\frac{1}{2}}^{L} = \mathbf{U}_{i,j}^{n+\frac{1}{2}} + \frac{\Delta x}{2}\left(\boldsymbol{\sigma}_{i,j}^{x}\right)^{n+\frac{1}{2}}, \quad \mathbf{U}_{i+\frac{1}{2}}^{R} = \mathbf{U}_{i+1,j}^{n+\frac{1}{2}} - \frac{\Delta x}{2}\left(\boldsymbol{\sigma}_{i+1,j}^{x}\right)^{n+\frac{1}{2}}. \tag{16}$$

Similarly, $\mathbf{U}_{j+1/2}^{L,R}$ are reconstructed in the y direction.

DOI: 10.1002/fld

Step 3: Corrector step. Godunov type method is used to update the solution,

$$\mathbf{U}_{i,j}^{n+1} = \mathbf{U}_{i,j}^{n} - \frac{\Delta t}{\Delta x}\left(\hat{\mathbf{F}}_{i+\frac{1}{2},j} - \hat{\mathbf{F}}_{i-\frac{1}{2},j}\right) - \frac{\Delta t}{\Delta y}\left(\hat{\mathbf{G}}_{i,j+\frac{1}{2}} - \hat{\mathbf{G}}_{i,j-\frac{1}{2}}\right) + \Delta t \mathbf{S}_{i,j}^{n+1/2}, \quad (17)$$

where $\widehat{\mathbf{F}}_{i+\frac{1}{2}} = \widehat{F}\left(\mathbf{U}_{i+\frac{1}{2}}^{L}, \mathbf{U}_{i+\frac{1}{2}}^{R}\right)$ is the numerical flux. We use the HLLC numerical flux

$$\widehat{F}^{\mathrm{HLLC}}\left(\mathbf{U}_{i+\frac{1}{2}}^{L}, \mathbf{U}_{i+\frac{1}{2}}^{R}\right) = \begin{cases} \mathbf{F}\left(\mathbf{U}_{i+\frac{1}{2}}^{L}\right), & S_L \geqslant 0 \\ \mathbf{F}\left(\mathbf{U}_{i+\frac{1}{2}}^{L}\right) + S_L\left(\mathbf{U}_{i+\frac{1}{2}}^{*L} - \mathbf{U}_{i+\frac{1}{2}}^{L}\right), & S_L \leqslant 0 \leqslant S_* \\ \mathbf{F}\left(\mathbf{U}_{i+\frac{1}{2}}^{R}\right) + S_R\left(\mathbf{U}_{i+\frac{1}{2}}^{*R} - \mathbf{U}_{i+\frac{1}{2}}^{R}\right), & S_* \leqslant 0 \leqslant S_R \\ \mathbf{F}\left(\mathbf{U}_{i+\frac{1}{2}}^{R}\right), & S_R \leqslant 0 \end{cases}. \quad (18)$$

The intermediate variables in the start region of the HLLC approximate Riemann solver are

$$\mathbf{U}^{*K} = h_K \frac{S_K - u_K}{S_K - S^*}\begin{bmatrix} 1 \\ S_* \\ v_K \end{bmatrix}, \quad K = L, R, \quad (19)$$

in the x direction, and

$$\mathbf{U}^{*K} = h_K \frac{S_K - v_K}{S_K - S_*}\begin{bmatrix} 1 \\ u_K \\ S_* \end{bmatrix}, \quad K = L, R, \quad (20)$$

in the y direction. The estimates for the left, middle [4], and right waves are

$$S_L = \min\left(u_L - \sqrt{\beta_L h_L}, u_R - \sqrt{\beta_R h_R}\right), \quad (21)$$

$$S_* = \frac{1}{2}(u_L + u_R) + \left(\sqrt{\beta_L h_L} - \sqrt{\beta_R h_R}\right), \quad (22)$$

$$S_R = \max\left(u_L + \sqrt{\beta_L h_L}, u_R + \sqrt{\beta_R h_R}\right). \quad (23)$$

This Godunov type method is stable under the CFL condition

$$\mathrm{CFL} = \Delta t \max_{\forall i,j}\left(\frac{|u_{i,j}| + \sqrt{(\beta_x h)_{i,j}}}{\Delta x} + \frac{|v_{i,j}| + \sqrt{(\beta_y h)_{i,j}}}{\Delta y}\right) < 1. \quad (24)$$

In this work, CFL number is taken to be 0.4.

3.2. Static resistance condition

In some studies [8, 15], the Coulomb friction law contained in Eqs (5)–(6) was used to determine the friction force everywhere any time. This is not correct when the granular mass is at static condition: $\mathbf{v} = 0$ or when the mass has little 'net driving force' [10, 11]. For the case $\mathbf{v} = 0$, Ref. [12] gave a static friction calculation method for one-dimensional (1D) cases. Here, we extend the same treatment to 2D cases.

DOI: 10.1002/fld

Theoretically, when the velocity is non-zero, that is, $\mathbf{v} \neq 0$, the friction force is computed with the Coulomb friction law. When $\mathbf{v} = 0$, the static resistance begins to work. Under this condition, the momentum Eqs (2)–(3) reduce to

$$\begin{bmatrix} h\dfrac{\partial u}{\partial t} \\ h\dfrac{\partial v}{\partial t} \end{bmatrix} = \begin{bmatrix} D_x \\ D_y \end{bmatrix} + \begin{bmatrix} f_x \\ f_y \end{bmatrix}, \tag{25}$$

where the combined gravity–topography–pressure terms are $D_x = h\sin\zeta - h\varepsilon\cos\zeta\dfrac{\partial z_b}{\partial x} - \beta_x h\dfrac{\partial h}{\partial x}$, and $D_y = -h\varepsilon\cos\zeta\dfrac{\partial z_b}{\partial y} - \beta_y h\dfrac{\partial h}{\partial y}$. Here, we set $K_x = K_y = 1$ in β_x and β_y for static cases as the literature. As the maximum friction that the materials can sustain is $\tau_{\max} = h\tan\delta$, the static resistance $\mathbf{f} = (f_x, f_y)$ can be deduced from Equation (25) as follows:

$$\begin{aligned} &\text{when } \mathbf{v} = 0 \text{ and } \sqrt{D_x^2 + D_y^2} < \tau_{\max}, \text{ set } \boldsymbol{f} = -\mathbf{D}, \\ &\text{when } \mathbf{v} = 0 \text{ and } \sqrt{D_x^2 + D_y^2} \geqslant \tau_{\max}, \text{ set } \boldsymbol{f} = -\frac{\mathbf{D}}{|\mathbf{D}|}\tau_{\max}. \end{aligned} \tag{26}$$

3.3. Stopping criteria

Stopping criteria for moving granular materials are introduced based on a Coulomb threshold in a way similar to Refs. [10, 11, 13]. The Coulomb threshold is $\tau_{\max} = h_i^{n+1}\tan\delta$. After obtaining a trial solution $\left(h_i^{n+1}, (\widetilde{hu})_i^{n+1}, (\widetilde{hv})_i^{n+1}\right)$ by using Equation (17) without the source term, we set an admissible basal shear stress at cell i and time $n+1$ as

$$\begin{bmatrix} \tilde{T}_x \\ \tilde{T}_y \end{bmatrix}_i^{n+1} = \begin{bmatrix} \widetilde{hu} \\ \widetilde{hv} \end{bmatrix}_i^{n+1} + \Delta t \begin{bmatrix} \sin\zeta - h\varepsilon\cos\zeta\frac{\partial z_b}{\partial x} \\ -h\varepsilon\cos\zeta\frac{\partial z_b}{\partial y} \end{bmatrix}_i^{n+1}. \tag{27}$$

The stopping criteria are

- When $\left|\tilde{\mathbf{T}}_i^{n+1}\right| < \tau_{\max}\Delta t$, that is, the admissible basal shear stress is less than the Coulomb threshold value, and when $|\nabla(h + z_b)| < \tan\phi$, the flow velocity $(u, v)_i^{n+1}$ is set to zero.
- Otherwise, solve for momentums $(hu)_i^{n+1}$ and $(hv)_i^{n+1}$ using Equation (17) with the full source term that uses the Coulomb basal friction law.

3.4. Flow front and half wet cell

In this article, we treat the flow front as Refs. [4, 15] did. Take the x direction as example. Suppose there is a flow front of which the right side is where $h = 0$, then a left-going rarefaction wave emanates from the front, and corresponding Riemann invariant is

$$I_L = u|_L + 2c_L = u|_0 + 2c_0. \tag{28}$$

Near the flow front, c_0 approaches zero as $h \to 0$, so the wet/dry front corresponds to the tail of the rarefaction and has exact propagating speed $u|_0 = u|_L + 2c_L$. This speed is taken as the estimated right-going wave speed S_R in the HLLC scheme.

The existence of dry beds is a particular feature that must be properly addressed when designing numerical schemes for shallow water flows. In the initial conditions, we set the granular thickness to zero where the debris flow has not arrived. In order to ensure numerical stability and avoid spurious

DOI: 10.1002/fld

diffusion of the flow front in updating the problematic partial-wet cells, we define a cell with $0 < h \leqslant \epsilon = 10^{-6}$ as the partial-wet cell. The momentum equations in the partial-wet cell are not solved and instead the velocities are extrapolated from the neighboring cell with the largest h in a way similar to Ref. [27]:

$$\mathbf{v} = \begin{cases} 0, & \text{if } h \leqslant 10^{-6} \ \forall \text{ neighbor cells} \\ \mathbf{v}|_{h_{\max}}, & \text{else} \end{cases} . \tag{29}$$

4. GPU PARALLEL COMPUTING

The most widely used programming model to implement GPU computing is NVIDIA's CUDA [18]. A CUDA program includes two kinds of codes, the serial codes and the parallel codes [28, 29]. The serial codes that run on the host (CPU) side are responsible for variables declaration, initialization, data transmission, and kernel invocation. The parallel codes (called 'kernel functions') running on the device (GPU) side are executed in parallel by massive threads organized to match the GPU hardware feature and to allow for mapping typical data structures (arrays, matrices). A number of threads makes up a block, and many blocks make up a grid, which is the counterpart of a kernel function. The block may be organized into 1D, 2D, or 3D array of threads, while the grid may be in 1D, 2D, or 3D array of blocks. 2D block and grid are illustrated in Figure 2.

Efficient GPU computing requires careful consideration of parallelism organization, memory management, and kernel scheduling [24]. In the present explicit predictor-corrector Godunov type method, all the calculations including slope limiter, reconstruction, flux and source evaluation, as well as solution update could be run on GPU in parallel. They are executed in several different kernel functions in order to keep each kernel small enough to use the fast registers and shared memories efficiently. Kernel *__global__ void getLRUD* is used to calculate the predictor step and data reconstruction step, *__global__ void RKresult* is used to compute the corrector step including flux and source evaluation, and *__global__ void Boundary* is used to apply boundary conditions. Furthermore, a single timestep Δt in Equation (24) is used for all cells, which is determined from the smallest permissible timestep in the whole domain. It can be computed on GPU in a reduction kernel (*__global__ void getDeltat*), which utilizes the shared memory and round query strategy [30]. This process is shown in Figure 3. As convention, we map each mesh cell to a thread, and each thread controls one or several mesh cells depending on whether the total number of computational cells

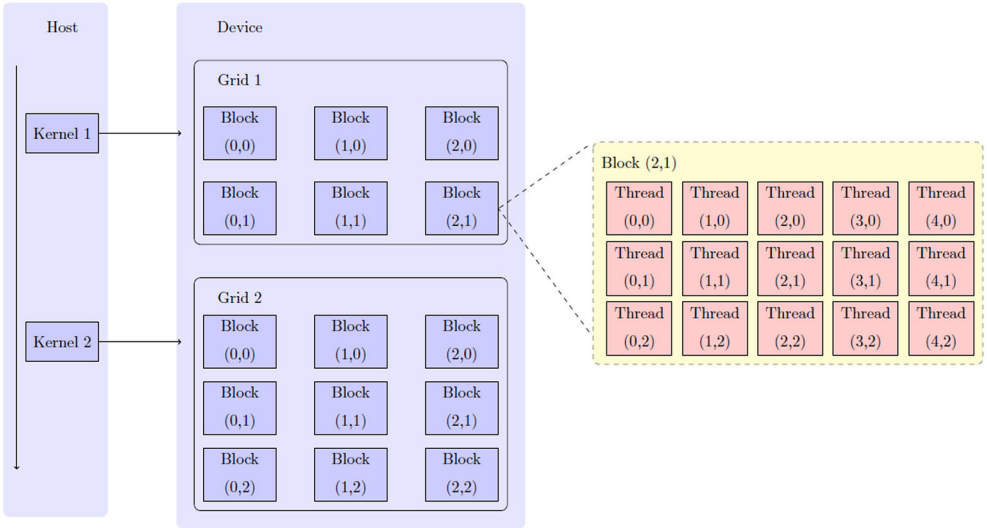


Figure 2. Sketch of organization of a compute unified device architecture (CUDA) program.

DOI: 10.1002/fld

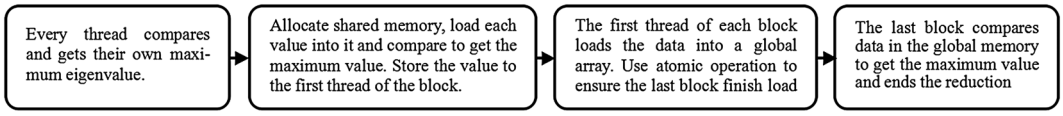


Figure 3. Timestep reduction process in GPU.

```
double **h;
h = new double*[Nx];    //Nx,Ny are the row number and column number,respectively
for(int i=0;i<Nx;i++)
    h[i] = new double [Ny];
```

Figure 4. Allocation of a second-rank pointer in CPU.

```
double **d_h,**dev_h;    //dev_h is an auxiliary variable
dev_h = new double *[Nx];
cudaMalloc((void**)&d_h,sizeof(double*)*Nx);
for(int i=0;i<Nx;i++)
    cudaMalloc((void**)&dev_h[i],sizeof(double)*Ny);
cudaMemcpy(d_h,dev_h,sizeof(double*)*Nx,cudaMemcpyHostToDevice);
```

Figure 5. Allocation of a second-rank pointer *d_h* with the help of auxiliary variable *dev_h* on GPU.

```
for(int i=0;i<Nx;i++)
    cudaMemcpy(h[i],dev_h[i],sizeof(double)*Ny,cudaMemcpyDeviceToHost);
```

Figure 6. Copy data from GPU to CPU.

exceeds the maximum number of threads allowed by the GPU capacity. In the first box in Figure 3, every thread computes the maximum eigenvalue of its own. In the second box, every block allocates a shared memory, all threads of a block load their values into the shared memory, then carry out binary comparison, and the resulting maximum value is stored at the first thread of the block. In the third box, the first thread of each block loads the block's max-value into a global array, during which atom addition operator is used to record the number of blocks that have done the data load and mark the finishing of data load. In the fourth box, all threads in the last block make binary comparison in the data in the global array and gets the maximum value of all cells. We then use this value to compute the time step from Equation (24).

A lot of GPU programmers will use 1D memory allocation, which can make addressing and memory copy between CPU and GPU fast. In this work, in order to make the addressing direct in the GPU program, we use 2D memory allocating for solution variables. In C++, the 2D memory allocation of a second-rank pointer h on CPU is via *new double* (Figure 4).

Now, the CPU memory can be used directly for the pointer h. But we could not use *cudaMalloc* to allocate a second-rank pointer on GPU in the way similar to *new double* in Figure 4, because *cudaMalloc* can only allocate a first-rank pointer. In this work, in order to use a second-rank pointer d_h on GPU, we allocate memories for the second class (pointing to pointer) of d_h and for the first class (pointing to double number) of a temporary second-rank pointer dev_h on GPU (line 3 and 5 in Figure 5, respectively), while the second class of dev_h is allocated on CPU (line 2). We then copy the memory of the second class of dev_h from CPU to that of d_h on GPU, as shown in the last line of Figure 5. In this way, the first-class pointer of d_h is associated with that of dev_h on GPU.

Now, we can use the second-rank pointer d_h as a 2D array directly in GPU, but when we want to copy the data of d_h back to CPU for output, we must copy each $dev\_h[i]$, which points to N_y double numbers allocated on GPU, instead of $d\_h[i]$, which is not explicitly allocated on GPU, back to $h[i]$ (Figure 6).

While this method of allocating 2D GPU memory is convenient for porting the serial program to the GPU program, the speed of accessing GPU memory is expected to be slower than that of the 1D memory allocation. Table I shows comparison of the computational times between 1D and 2D

DOI: 10.1002/fld

Table I. Comparison of the running times for matrix multiplication $C = A \times B$.

n	1D allocation	2D allocation	2D(serial code)
128	1.1 ms	8.2 ms	10 ms
256	3.1 ms	23.4 ms	140 ms
512	14.2 ms	109.2 ms	1340 ms
1024	90.7 ms	716.2 ms	12040 ms

memory allocation methods to compute $C = A \times B$, where A, B and C are $n \times n$ matrices. The results were obtained in double precision using an NVIDIA Tesla C2075 GPU with 6 GB memory and an Intel Xeon Westmere 5675 3.06 GHz 6-core CPU with 24 GB memory. Only global memory is used in the code. It can be seen from Table I that 2D memory allocation method is much slower than the 1D counterpart, although it achieves considerable speedup relative to 2D allocation serial code for large n. This indicates that 1D memory allocation is desirable for efficient GPU computing. However, the difference may not be so large for a practical application code because other bottlenecks occur even using 1D memory allocation, as will be seen from Table IV. The 2D memory allocation technique presented here may be helpful in porting a serial code to a GPU code.

5. NUMERICAL EXAMPLES

We use three examples to verify the accuracy of the numerical model and compare the efficiency of both CPU and GPU programs. Meanwhile, we compare the efficiency of present GPU implementation with earlier GPU codes for modeling shallow water flows in the literature. Finally, we simulate a granular jet example to demonstrate the application of the developed flow solver. The computer we used is a server with an Intel Xeon Westmere 5675 (3.06 GHz) CPU and an NVIDIA Tesla C2075 GPU. All computations use double precision.

5.1. Dam break problem

For the dam break problem [31], the computational domain is set to $[-12.8, 12.8] \times [-1.6, 1.6]$ m^2, and the initial conditions are

$$\mathbf{v} = 0,\ h = \begin{cases} h_0 & x < 0 \\ 0 & x \geqslant 0 \end{cases}, \tag{30}$$

where $h_0 = 10$ m, $\zeta = 40^{\circ}$, and $\delta = 24.5^{\circ}$. The original example is a 1D problem, but we calculate it with 2D code and cut a slice from the plane $y = 0$ to compare with the exact solution.

In order to get the exact solution, we rewrite the SH equations in dimensionalized form as in Ref. [31] (we correct the sign errors in [31]):

$$\begin{aligned} &h_t + uh_x + hu_x = 0, \\ &u_t + g\cos\zeta h_x + uu_x = -g\cos\zeta\,(\tan\delta - \tan\zeta)\,. \end{aligned} \tag{31}$$

Notice that the x direction is downward along the slope. With the change of variables

$$\chi = x + \frac{1}{2}g\cos\zeta(\tan\delta - \tan\zeta)t^2, \quad \mathcal{U} = u + g\cos\zeta(\tan\delta - \tan\zeta)t, \tag{32}$$

Equation (31) reduces to a homogeneous system of equations; then by utilizing the Ritter solution of a frictionless dam break, we can obtain the analytical solution of a granular dam break problem [31]:

DOI: 10.1002/fld

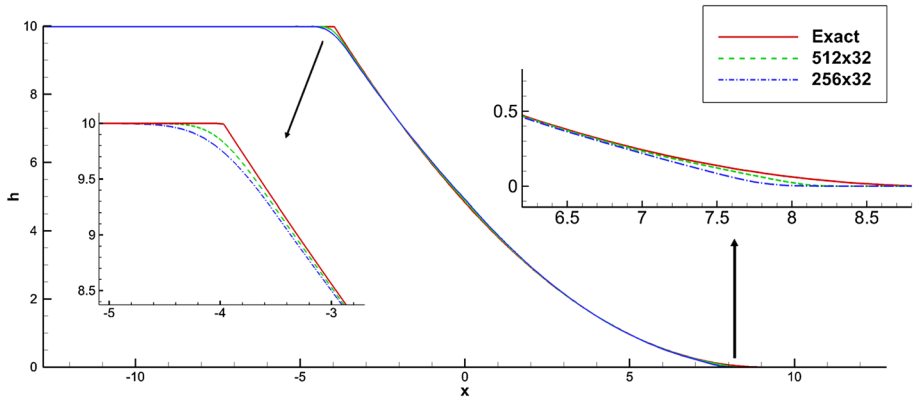


Figure 7. Exact and numerical solutions on two grid numbers for the dam break problem after 0.5 s.

Table II. Errors of the computed depth and orders of accuracy for the dam break problem.

Grid	L_2 error	order	L_∞ error	Order
64×8	0.130163	—	0.452701	—
128×16	0.073894	0.8168	0.345509	0.3898
256×32	0.0409712	0.8508	0.252918	0.4501
512×64	0.0223083	0.8770	0.177114	0.5140

$$(h,\mathcal{U}) = \begin{cases} (h_0,0), & \chi < -c_0 t \\ \left(\dfrac{h_0}{9}\left(2-\dfrac{\chi}{c_0 t}\right)^2, \dfrac{2}{3}\left(\dfrac{\chi}{t}+c_0\right)\right), & -c_0 t \leqslant \chi \leqslant 2c_0 t \\ (0,0), & \chi > 2c_0 t \end{cases}. \tag{33}$$

where $c_0 = \sqrt{g h_0 \cos\zeta}$.

Figure 7 shows the computed depth in comparison with the exact solution after 0.5 s. We can observe some differences only at the right flow front and the left tail of the rarefaction wave. However, on the finer grid 512×32, the difference is further reduced. In Table II, we give numerical errors of the computed depth h relative to the exact solution. The errors are reduced with increasing grid sizes, but the numerical order is not second order. It may be related to the initial data jump, the limiter, the numerical flux, the treatment of the partial wet cells, and so on. However, the numerical order is compatible with other general convergence rates for similar test models (e.g., [10]).

5.2. *Quiescent equilibrium and start/stop flow conditions*

This example is used to illustrate the role of the stopping criteria and static resistance in simulating quiescent equilibrium and start/stop flow conditions. According to Ref. [31], we give the bed level z_b and the initial free surface level $H = z_b + h$ as

$$z_b = \frac{\sqrt{x^2+y^2}}{2}, \quad H = z_b + h = 0.2 - z_b. \tag{34}$$

Because h is non-negative, the initial granular mass only covers the region where $z_b(x,y) \leqslant 0.1$. The computational domain is $[-0.3, 0.3]\times[-0.3, 0.3]$ m^2, and the mesh used is 256×256. In the first case, we set $\delta = \phi = 40^o$. As the free surface slope angle of the mass is less than the internal friction angle, which decides the maximum permissible value of the pile slope, the initial pile will keep its static state all the time. In Table III, we give numerical errors of the computed depth relative to the

DOI: 10.1002/fld

Table III. Errors of the computed depth and orders of accuracy for the static pile problem at $\delta = 40^{o}$, $t = 1$ s.

Mesh	L_2 error	Order	L_∞ error	Order
64×64	0.00348176	—	0.0337154	—
128×128	0.00160774	1.1148	0.0225977	0.5772
256×256	0.000719244	1.1605	0.0148297	0.6077
512×512	0.00032317	1.1542	0.00963436	0.6222

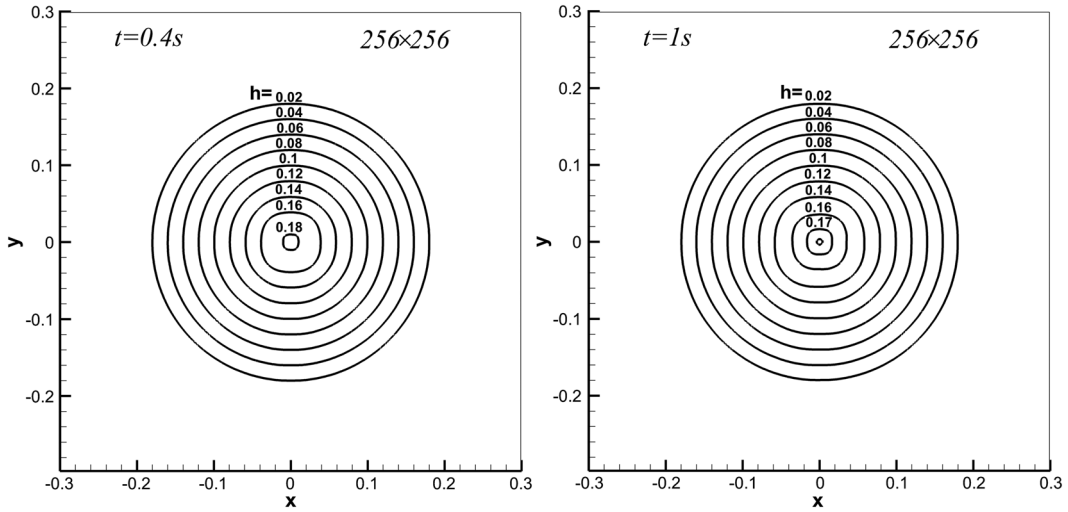


Figure 8. The counter lines of depth with $\delta = 40^{o}$ and $t = 0.4$ s (left), $t = 1$ s (right).

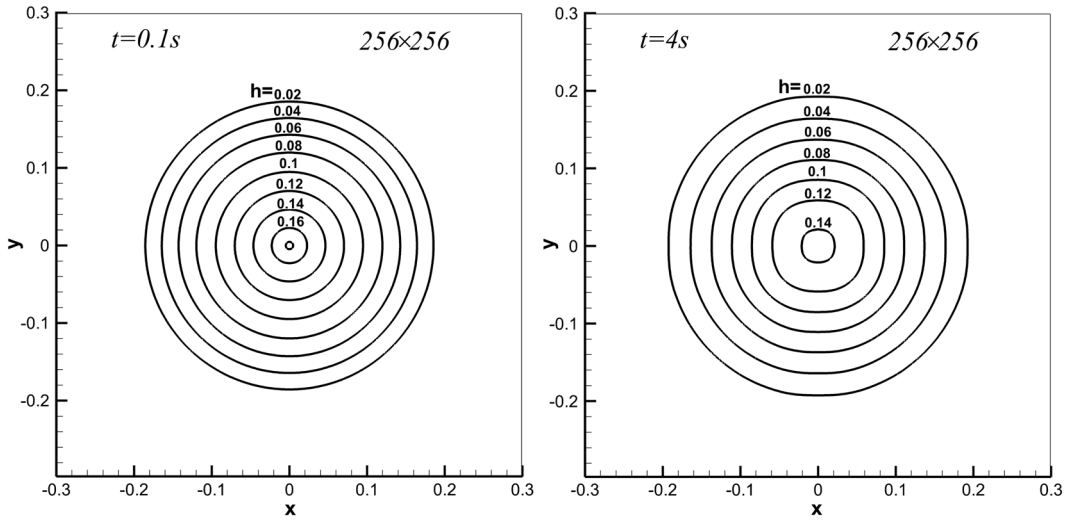


Figure 9. The counter lines of depth with $\delta = 20^{o}$ at $t = 0.1$ s and $t = 4$ s.

exact solution on various mesh sizes. The numerical orders of accuracy are improved compared with Table II. Figure 8 shows the computed counter lines of the depth at $t = 0.4$ s and $t = 1$ s, we can observe the mass at $t = 1.0$ s is basically the same as at $t = 0.4$ s, with a little smearing in the center.

In the second case, we set $\delta = \phi = 20^{o}$. As the friction angle is smaller than the initial free surface slope angle, the mass begins to flow, and Figure 9 shows the counter line distributions at $t = 0.1$ s and $t = 4$ s. The distribution after $t = 4$ s will remain unchanged as the flow has reached static balance.

DOI: 10.1002/fld

5.3. *Inclined plane merging continuously into a horizontal plane*

In this subsection, we present a simulation example of finite granular mass sliding down an inclined plane and merging continuously into a horizontal plane [8]. The computational domain is $x \in [0, 30]$, $y \in [-7, 7]$ (dimensionless). The ratio ε and the stretching parameter λ in Equation (5) are both set to 1 in this work. The inclined region lies $x \in [0, 17.5]$, and the horizontal region lies $x \geqslant 21.5$. The transition zone between them is smooth and the inclination angle is

$$\zeta(x) = \begin{cases} \zeta_0, & 0 \leqslant x \leqslant 17.5, \\ \zeta_0 \left(1 - \dfrac{x - 17.5}{4}\right), & 17.5 < x < 21.5, \\ 0^{\mathrm{o}}, & x \geqslant 21.5, \end{cases} \tag{35}$$

where $\zeta_0 = 35^{\mathrm{o}}$. We set $\delta = \phi = 30^{\mathrm{o}}$. An hemispherical shell with radius of $r_0 = 1.85$ centered at $(x_0, y_0) = (4, 0)$ is released suddenly at $t = 0$.

Figure 10 shows the thickness contours of the fluid at nine different times as computed on 480×224 meshes without using the stopping criteria so as to compare with the results in Ref. [8]. The results at $t = 6, 15$, and 24 are comparable. The quantitative difference between present and Wang's results may attribute to different schemes, different mesh sizes, and different ε used.

We may compare the difference between with and without the stopping criteria for later stage of the avalanche. Figure 11 shows results at $t = 40$ on three gradually refined meshes. The pile becomes wider as the mesh is refined. For this particular example, the difference between with and without the stopping criteria is small.

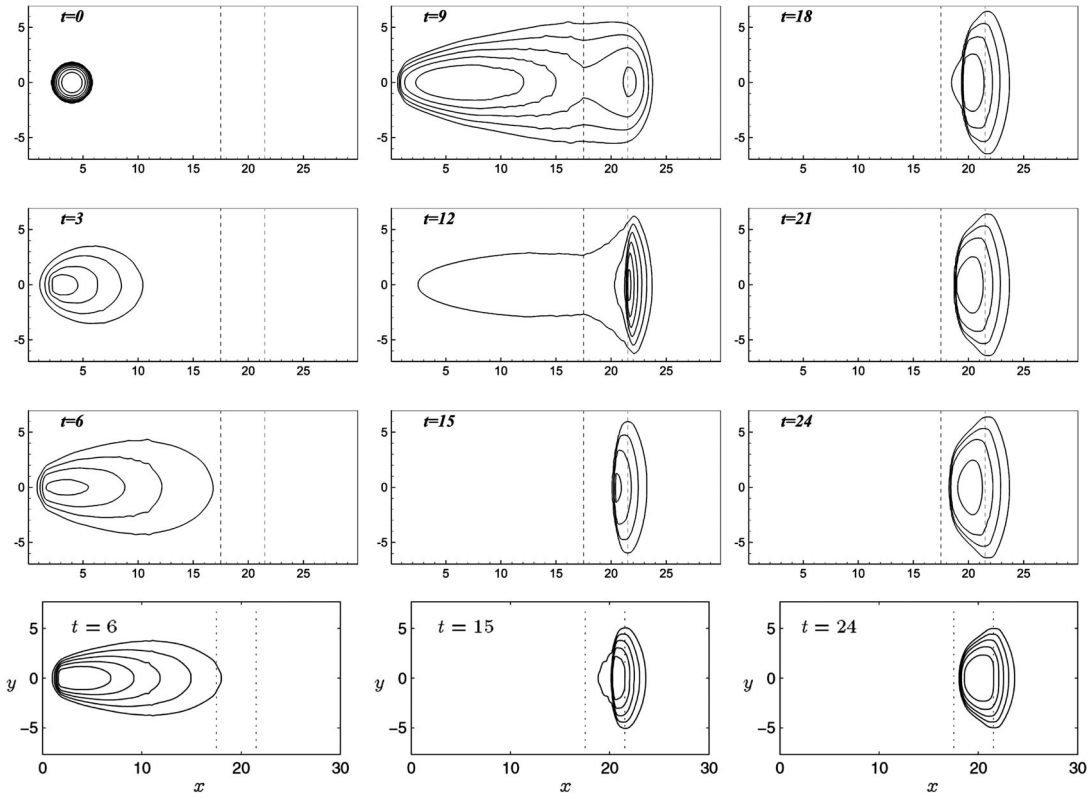


Figure 10. Thickness contours of the flow at nine times as computed on 480×224 meshes without the stopping criteria. The last row is Wang's results [8]. The quantitative difference may be caused by different schemes and mesh sizes and possibly different ε used (Ref. [8] did not give mesh numbers and ε).

DOI: 10.1002/fld

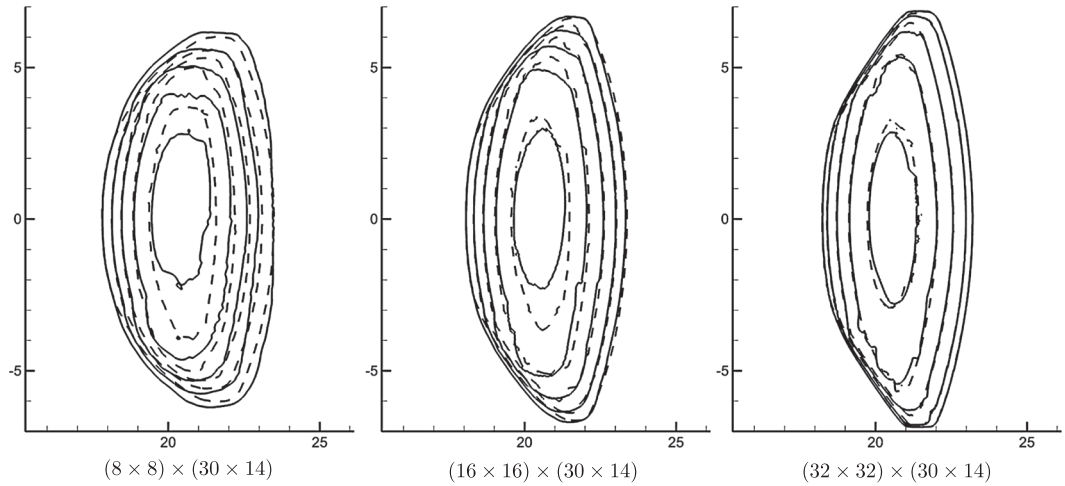


Figure 11. Comparison of calculated results between with and without the stopping criteria at $t = 40$ using three consecutively refined meshes. $(8 \times 8) \times (30 \times 14)$ means there are 8×30 meshes in the x direction and 8×14 meshes in the y direction. Solid lines denote results with the stopping criteria, while dashed lines denote without.

Table IV. CPU and GPU times for run from $t = 0$ to $t = 24$ with four meshes, where $M_0 N_0 = 30 \times 14$.

Mesh	CPU serial time	GPU time (2D)	S_p (2D)	GPU time (1D)	S_p(1D)
$4 \times 4 \times (M_0 N_0)$	13220 ms	758.8 ms	17.4	538.4 ms	24.56
$8 \times 8 \times (M_0 N_0)$	100480 ms	3466 ms	29.0	2452 ms	41.0
$16 \times 16 \times (M_0 N_0)$	793250 ms	21672.7 ms	36.6	16169.1 ms	49.1
$32 \times 32 \times (M_0 N_0)$	6420670 ms	156224.8 ms	41.1	122621.7 ms	52.4

1D, one-dimensional memory allocation ; 2D, two-dimensional memory allocation.

We have also compared both CPU and GPU programs in this example. The running times of both CPU and GPU programs are recorded in order to compare their efficiencies. Define the GPU/CPU speedup ratio as

$$S_p = \frac{\text{time of CPU serial program}}{\text{time of GPU program}}. \tag{36}$$

Timings for run from $t = 0$ to $t = 24$ with four different meshes are given in Table IV. It can be seen that with the increase of the mesh number, the speedup ratio increases. This can be explained as follows. In the GPU program, there are some parts that need to be serialized. Hence, the running time is composed of the serial time and the parallel time. With the increase of the mesh number, the parallel computation work running on GPU will increase and finally saturate because of GPU's parallel capacity, while the serial work running on CPU and the memory copy work will not increase much. Therefore, the speedup ratio will increase until a maximum value is attained. Table IV also compares the 2D and 1D memory allocations. Unlike in previous matrix multiplication case, the 1D GPU memory allocation code is only a bit faster than the 2D memory allocation code for this case.

As to performance comparison with previous work, the best specific time (i.e., time per cell per time step) of the present implementation is approximately 5.2×10^{-8} s per cell per time step (double precision) with 1D memory allocation on Tesla C2075 for this 2D example (Table V). This rate is a bit slower than but comparable to other CUDA–GPU implementations for the shallow water equations, for example, 1×10^{-8} s per cell per time step (in double precision) on Tesla M2075 in Ref. [24], and $1.9 \times 10^{-8} \sim 2.2 \times 10^{-9}$ s per cell per time step on Tesla M2070 in Ref. [25]. Because the SH equations have extra calculations for K_x, K_y, and stopping criteria compared with

DOI: 10.1002/fld

Table V. Specific times for two-dimensional and one-dimensional memory allocations with four meshes, where $M_0N_0 = 30 \times 14$.

Mesh	2D specific time(s/cell/step)	1D specific time(s/cell/step)
$4 \times 4 \times (M_0N_0)$	1.7×10^{-7}	1.1×10^{-7}
$8 \times 8 \times (M_0N_0)$	1.1×10^{-7}	6.8×10^{-8}
$16 \times 16 \times (M_0N_0)$	9.0×10^{-8}	5.7×10^{-8}
$32 \times 32 \times (M_0N_0)$	8.3×10^{-8}	5.2×10^{-8}

1D, one-dimensional memory allocation; 2D, two-dimensional memory allocation.

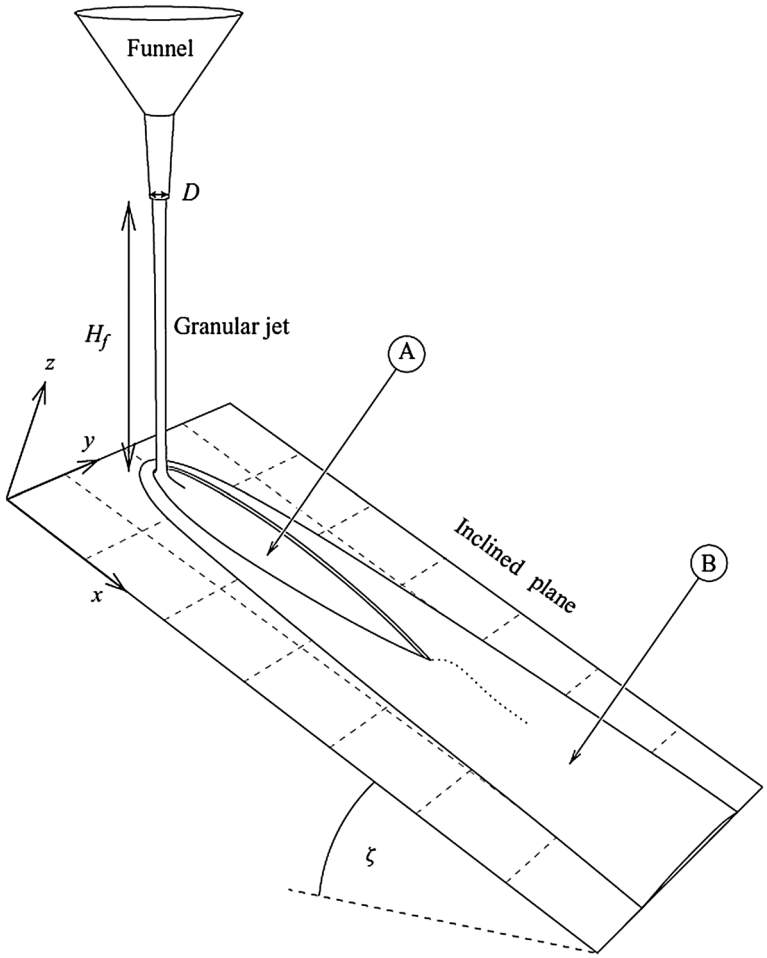


Figure 12. Experimental apparatus [32]. The granular jet impinges on the inclined plane, spreads into a region of thin, fast flow (A), and then passes through a jump, becoming thicker and slower flow (B).

the shallow water equations, and different papers had used different numerical schemes and different treatment of dry-wet fronts, a precise comparison of computational efficiency is out of the scope of the present paper.

5.4. Granular jet

Johnson and Gray [32] conducted a series of experiments of granular jet impinging on an inclined plane. According to their results, there are steady-state teardrop shock, blunted shock, and unstable time-dependent flows that are dependent on the inclined angle ζ and the height of the release position of the jet relative to the plane H_f, as sketched in Figure 12.

DOI: 10.1002/fld

We simulate this problem with the 2D memory allocation GPU program by giving the initial conditions in the circular region centered at the stagnation point on the plane as Ref. [32]. In the circular region with a chosen radius $R_{imp} = 2R$, the analytical solution [32]

$$\mathbf{v} = \left(u_{jet}\cos\theta, u_{jet}\sin\theta\right), \tag{37}$$

$$h = \frac{R^2\cos^3\zeta}{2r(1-\sin\zeta\cos\theta)^2}, \tag{38}$$

is valid, where R is the radius of the cylindrical jet, (r,θ) is the polar coordinates centered at the stagnation point, and u_{jet} is the jet velocity. We use the same parameters $\zeta = 26.7^{\circ}$, $u_{jet} = 0.99$ m/s, and $R \approx 0.5D = 7.5$ mm as in Ref. [32] for the steady case with a teardrop-shaped shock.

If using a constant basal friction coefficient μ, one can only obtain one steady flow of uniform thickness at a single slope angle of $\zeta = \tan^{-1}\mu$. But experiments [33] have observed steady uniform flows over a range of slope angles over a roughened bed. This motivated Johnson and Gray [32] to use a variable basal friction law to model the experiments. In 1999, Pouliquen *et al.* [34] demonstrated that a steady flowing layer with uniform thickness can exist on a slope inclined at an angle ζ with a minimum depth $\tilde{h}_{stop}(\zeta)$, below which steady flow is not observed, here $\tilde{h}$ denotes dimensional flow depth. They also found an empirical dependence of the ratio of flow depth $\tilde{h}$ to $\tilde{h}_{stop}(\zeta)$ on the Froude number,

$$\mathrm{Fr} \equiv \frac{|\mathbf{v}|}{\sqrt{h\varepsilon\cos\zeta}} = \beta\frac{\tilde{h}}{\tilde{h}_{stop}(\zeta)}, \tag{39}$$

where $\beta = 0.136$ is a measured constant for glass beads. At steady flowing states, $\mu = \tan\zeta$ holds, and the inverse function of $\tilde{h}_{stop}(\zeta)$ together with Equation (39) can lead to an equation for the friction coefficient $\mu_{stop} = \tan[\zeta_{stop}(\tilde{h}\beta/\mathrm{Fr})]$, implying that $\mu_{stop} = \mu_{stop}(h')$. Similar to Ref. [32], we use the form for the function $\mu_{stop}(h')$ as given Ref. [34] and the form for the function $\mu_{start}(h')$ as given in Ref. [33], respectively, with experimentally measured friction angles,

$$\mu_{stop}(h') = \tan\zeta_1 + (\tan\zeta_2 - \tan\zeta_1)\,e^{-\frac{h'}{\varphi}}, \tag{40}$$

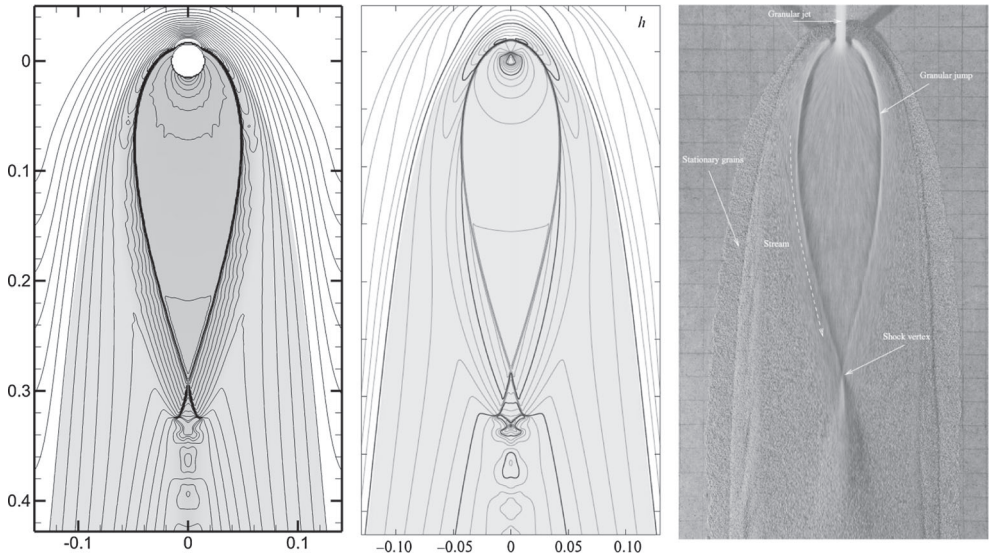
$$\mu_{start}(h') = \tan\zeta_3 + (\tan\zeta_2 - \tan\zeta_1)\frac{1}{1+\frac{h'}{\varphi}}, \tag{41}$$

where $\zeta_1 = 21^{\circ}, \zeta_2 = 30.7^{\circ}$, and $\zeta_3 = 22.2^{\circ}$ are measured friction angles [33]. The function $\mu_{stop}(h')$ is a fit to the experimental measurements of $\tilde{h}_{stop}(\zeta)$, and it takes the aforementioned form with a transition between two friction angles ζ_1 and ζ_2. The function $\mu_{start}(h')$ is a static friction coefficient and is calculated by measuring the maximum inclination angle at which a uniform layer of stationary material with depth h' starts to move. The free parameter φ has the dimension of length and depends on the granular material and surface properties of the plane and characterizes the depth of flow over, which a transition between the two friction angles ζ_1 and ζ_2 occurs. In order to make the steady state fit the experiment better, we take the free parameter $\varphi = 6$ mm in this paper. The functions μ_{stop} and μ_{start} are used in the following way.

- When $\mathrm{Fr} > \beta$, that is, for the steady regime where $\tilde{h} > \tilde{h}_{stop}$, the friction law (40) is used as

$$\mu = \mu_{stop}\left(\frac{\tilde{h}\beta}{\mathrm{Fr}}\right). \tag{42}$$

DOI: 10.1002/fld



(a) present result (b) simulation [32] (c) experiment with sand [32]

Figure 13. Comparison of (a) present result with 512×256, where the contours are of flow depth and shading indicates the flowing region of material, (b) simulation by Johnson and Gray [32], where the contours are of flow depth at intervals of 1 mm, with dark contours at intervals of 5 mm and shading indicates the flowing region of material, and (c) experimental results. Unit in x, y ticks in (a) and (b) is meter, while grid squares in (c) are 2-cm intervals. Different from (b), in (a), we do not plot contours inside the circular region with radius R_{imp} representing the granular jet, where the analytical solutions (37)-(38) are imposed.

- When $0 < \mathrm{Fr} < \beta$, we follow the method [33] in interpolating between the static and steady-flow friction coefficients with a power function

$$\mu = \left(\frac{\mathrm{Fr}}{\beta}\right)^{\gamma} \left(\mu_{stop}(\tilde{h}) - \mu_{start}(\tilde{h})\right) + \mu_{start}(\tilde{h}), \tag{43}$$

  where $\gamma = 10^{-3}$.

The earth pressure coefficients are set to unity in this example as done in Ref. [32]. Figure 13 shows the steady-state depth contours for a teardrop-shaped shock flow obtained with the present GPU code on 512×256 mesh cells in comparison with the numerical simulation and experimental visualization in Ref. [32]. We can clearly observe that the contour lines, the shape and position of the shock, and the stationary region are comparable to the literature. However, present shock is slightly longer than that in [32], which may be caused by different values of the free parameter φ used.

6. CONCLUSION

In this work, we have presented an improved numerical method for the SH model for modeling granular avalanche flows by taking into account the static resistance condition and the stopping criteria. We apply Davis's second-order predictor-corrector Godunov method in conjunction with the MUSCL reconstruction and HLLC scheme to GPU computing. The accuracy of the resulting numerical algorithm is verified through several typical numerical examples and the computational speedup of the GPU code relative to the CPU serial code is shown to be evident. Comparisons between using 1D and 2D memory allocations and with previous GPU codes for the shallow water equations are also represented. As an application example, one case with teardrop-shaped shock in the granular jet experiments of Johnson and Gray is simulated, and the computed results are comparable to experimental and numerical results in the reference.

DOI: 10.1002/fld

ACKNOWLEDGEMENTS

The authors thank the support of state key program for developing basic sciences (2010CB731505) and Natural Science Foundation of China (11321061, 11261160486).

REFERENCES

1. Hutter K, Rajagopal KR. On flows of granular materials. *Continuum Mechanics and Thermodynamics* 1994; **6**(2): 81–139.
2. Savage SB, Hutter K. The motion of a finite mass of granular material down a rough incline. *Journal of Fluid Mechanics* 1989; **199**(1):177–215.
3. Iverson RM, Denlinger RP. Flow of variably fluidized granular masses across three-dimensional terrain: 1. Coulomb mixture theory. *Journal of Geophysical Research* 2001; **106**(B1):537–552.
4. Denlinger RP, Iverson RM. Flow of variably fluidized granular masses across three-dimensional terrain: 2. Numerical predictions and experimental tests. *Journal of Geophysical Research* 2001; **106**(B1):553–566.
5. Pitman EB, Le L. A two-fluid model for avalanche and debris flows. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 2005; **363**(1832):1573–1601.
6. Bouchut F, Westdickenberg M. Gravity driven shallow water models for arbitrary topography. *Communications in Mathematical Sciences* 2004; **2**(3):359–389.
7. Denlinger RP, Iverson RM. Granular avalanches across irregular three-dimensional terrain: 1. Theory and computation. *Journal of Geophysical Research* 2004; **109**:14. DOI: 10.1029/2003JF000085.
8. Wang YQ, Hutter K, Pudasaini SP. The Savage-Hutter theory: a system of partial differential equations for avalanche flows of snow, debris, and mud. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik* 2004; **84**(8):507–527.
9. Tai Y-C, Noelle S, Gray JMNT, Hutter K. Shock-capturing and front-tracking methods for granular avalanches. *Journal of Computational Physics* 2002; **175**(1):269–301.
10. Mangeney-Castelnau A, Vilotte JP, Bristeau MO, Perthame B, Bouchut F, Simeoni C, Yerneni S. Numerical modeling of avalanches based on Saint Venant equations using a kinetic scheme. *Journal of Geophysical Research* 2003; **108**(B11):17. DOI: 10.1029/2002JB002024.
11. Mangeney-Castelnau A, Bouchut F, Vilotte JP, Lajeneusse E, Aubertin A, Pirulli M. On the use of Saint Venant equations to simulate the spreading of a granular mass. *Journal of Geophysical Research* 2005; **110**:17. DOI: 10.1029/2004JB003161.
12. Fan YY, Wang SJ, Wang EZ. Characteristics of static and dynamic resistance of one-dimensional debris flow and its numerical simulation. *Journal of Engineering Geology* 2010; **18**(6):857–861. (In Chinese).
13. Maeno F, Hogg AJ, Sparks R, Matson GP. Unconfined slumping of a granular mass on a slope. *Physics of Fluids* 2013; **25**:023302. DOI: 10.1063/1.4792707.
14. Toro EF. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. Springer: Dordrecht Heidelberg London, New York, 2009.
15. Patra AK, Bauer AC, Nichita CC, Pitman EB, Sheridan MF, Bursik M, Rupp Bc, Webber A, Stinton AJ, Namikawa LM, Renschler CS. Parallel adaptive numerical simulation of dry avalanches over natural terrain. *Journal of Volcanology and Geothermal Research* 2005; **139**(1):1–21.
16. Kurganov A, Tadmor E. New high-resolution central schemes for nonlinear conservation laws and convection–diffusion equations. *Journal of Computational Physics* 2000; **160**:214–282.
17. Davis SF. Simplified second order Godunov-type methods. *Society of Industrial and Applied Mathematics (SIAM) Journal on Scientific and Statistical Computing* 1998; **9**:445–473.
18. CUDA: reference manual, version 4.0.: Nvidia, Santa Clara, CA, 2011.
19. Kuo F-A, Smith M, Hsieh C-W, Chou C-Y, Wu J-S. GPU acceleration for general conservation equations and its application to several engineering problems. *Computers & Fluids* 2011; **45**:147–154.
20. Brodtkorb A, Sætra M, Altinakar M. Efficient shallow water simulations on GPUs: implementation, visualization, verification, and validation. *Computers & Fluids* 2012; **55**:1–12.
21. Sætra M, Brodtkorb A. Shallow water simulations on multiple GPUs. In *Applied Parallel and Scientific Computing*, Jónasson K (ed.)., Vol. 7134 of Lecture Notes in Computer Science. Springer: Berlin Heidelberg, 2012; 56–66.
22. Sætra M, Brodtkorb A, Lie K-A. Efficient GPU-implementation of adaptive mesh refinement for the shallow-water equations. *Journal of Scientific Computing* 2014; **60**:1–26. DOI: 10.1007/s10915-014-9883-4.
23. Garcia-Rodriguez J, Vinas J, Lobeiras J, Fraguela B, Arenaz M, Amor M, Castro M, Doallo R. A multi-GPU shallow-water simulation with transport of contaminants. *Concurrency and Computation: Practice and Experience* 2013; **25**(8):1153–1169.
24. Smith LS, Liang QH. Towards a generalised GPU/CPU shallow-flow modelling tool. *Computers & Fluids* 2013: 334–343.
25. Vacondio R, Palù A, Mignosa P. GPU-enhanced finite volume shallow water solver for fast flood simulations. *Environmental Modelling & Software* 2014; **57**:60–75.
26. Hutter K, Siegel M, Savage SB, Nohguchi Y. Two-dimensional spreading of a granular avalanche down an inclined plane part I. Theory. *Acta Mechanica* 1993; **100**(1–2):37–68.
27. Bradford SF, Sanders BF. Finite-volume model for shallow-water flooding of arbitrary topography. *Journal of Hydraulic Engineering* 2002; **128**(3):289–298.

28. Zhang S, Chu YL. *GPU high performance computing with CUDA*, 2009. Chinese Hydrology Pub. Hsouse, (In Chinese).
29. Sanders J, Kandrot E. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional: Boston, USA, 2010.
30. Liang S, Liu W, Yuan L. Solving seven-equation model for compressible two-phase flow using multiple GPUs. *Computers & Fluids* 2014; **99**:156–171.
31. Juez C, Murillo J, García-Navarro P. 2D simulation of granular flow over irregular steep slopes using global and local coordinates. *Journal of Computational Physics* 2013; **255**:166–204.
32. Johnson CG, Gray JMNT. Granular jets and hydraulic jumps on an inclined plane. *Journal of Fluid Mechanics* 2011; **675**(1):87–116.
33. Pouliquen O, Forterre Y. Friction law for dense granular flows: application to the motion of a mass down a rough inclined plane. *Journal of Fluid Mechanics* 2002; **453**:133–151.
34. Pouliquen O. Scaling laws in granular flows down rough inclined planes. *Physics of Fluids* 1999; **11**:542–548.

DOI: 10.1002/fld