Computers & Geosciences 78 (2015) 63–72

Contents lists available at ScienceDirect

Computers & Geosciences

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

An efficient solution for hazardous geophysical flows simulation using GPUs

A. Lacasta*, C. Juez, J. Murillo, P. García-Navarro

LIFTEC-EINA, CSIC-Universidad de Zaragoza, Spain

ARTICLE INFO

Article history:
Received 14 July 2014
Received in revised form
9 January 2015
Accepted 13 February 2015
Available online 16 February 2015

Keywords:
CUDA
GPU
Landslides
Numerical modeling
Shallow flow
Coulomb forces

ABSTRACT

The movement of poorly sorted material over steep areas constitutes a hazardous environmental problem. Computational tools help in the understanding and predictions of such landslides. The main drawback is the high computational effort required for obtaining accurate numerical solutions due to the high number of cells involved in the calculus. In order to overcome this problem, this work proposes the use of GPUs for decreasing significantly the CPU simulation time. The numerical scheme implemented in GPU is based on a finite volume scheme and it was validated in previous work with exact solutions and experimental data. The computational cost time obtained with the Graphical Hardware technology, GPU, is compared against Single-Core (sequential) and Multi-Core (parallel) CPU implementations. The GPU implementation allows to reduce the computational cost time in two orders of magnitude.

1. Introduction

Landslides play an important role on the evolution of landscape and constitute an important environmental risk. They can be responsible for dramatic civilian damages and that is the reason why the building of defenses and barriers is required. The computational tools are a suitable partner for developing a careful design of such elements. Over recent years, reliable predictions of the spreading of granular material have been obtained (Pirulli et al., 2007; Pirulli and Mangeney, 2008; Moretti et al., 2012; Bouchut et al., 2008) and numerical results have been validated against series of experiments based on granular dry flows (Savage and Hutter, 1989; Iverson and Denlinger, 2001; Pouliquen and Forterre, 2002; Lajeunesse et al., 2004; Mangeney et al., 2010). In particular, Murillo and García-Navarro (2012) and Juez et al. (2013) have recently presented a robust finite volume upwind scheme which includes the presence of steep slopes leading to obtain promising results.

Once the forecasting capability of the computational tool has been achieved another important concern is the improvement of the efficiency in terms of the computational cost. This type of geophysical flow involves the study of huge domains, such as catchments, mountains or gorges, where an accurate digital terrain model is required in order to mimic the complex topography of the terrain. For this reason, implementations based on the most recent GPU hardware (Graphics Processing Unit) emerges as a promising strategy for handling this environmental and up to date problem. This hardware consists of thousands of simple arithmetic processing units originally designed to pixel-based computations. In the recent years, this technology has been successfully extended and applied to more general problems in order to accelerate them. This concept is also known as General Purpose Computing on Graphics Processing Units (GPGPU) (Owens et al., 2007).

In terms of scientific computation, the last four decades have followed Moore's (2003) law, i.e. the number of transistors on a chip increases exponentially. This integration allowed to obtain faster and faster applications, by recompiling the code for these new processors. Unfortunately, power has become the primary design constraint for chip designers, where both energy and power dissipation create a technological barrier for the integration capacity (Dreslinski et al., 2010). Nevertheless, Multi-Core micro-architecture together with an adequate programming model (OpenMP is one of the most extended) brings a chance to exploit the parallelism of some parts of the code (Sharma and Gupta, 2013). Moreover, the Multi-Core paradigm has a large power consumption rate when performing small tasks and, for these purposes, Many-Core systems appear to be a very interesting option (Borkar, 2007). Many-Core architectures are those composed of smaller and not so complex cores that usually have special purposes. Industrial implementation of this solution has been

* Corresponding author.
E-mail addresses: alacasta@unizar.es (A. Lacasta), carmelo@unizar.es (C. Juez), javier.murillo@unizar.es (J. Murillo), pigar@unizar.es (P. García-Navarro).

http://dx.doi.org/10.1016/j.cageo.2015.02.010
0098-3004/© 2015 Elsevier Ltd. All rights reserved.

obtained in the field of Graphical Processing, where several efforts have been devoted to make more powerful devices. Indeed, this technology has been historically oriented to the very particular task of performing shading operations when rendering graphics. Following Lacasta et al. (2014), the purpose of the present work is to apply this hardware to the simulation of hazardous and very high time consuming geophysical flows.

The outline of this work is as follows: Section 2 is devoted to explain the mathematical model and numerical scheme used for modeling and solving the landslides behavior. In Section 3, the implementation on GPU is described. Section 4 gathers several experimental and realistic cases considered for testing the GPU performance. The differences on the computational cost time between the sequential, parallel and GPU strategies is discussed in Section 5. Finally in Section 6 the conclusions are summarized.

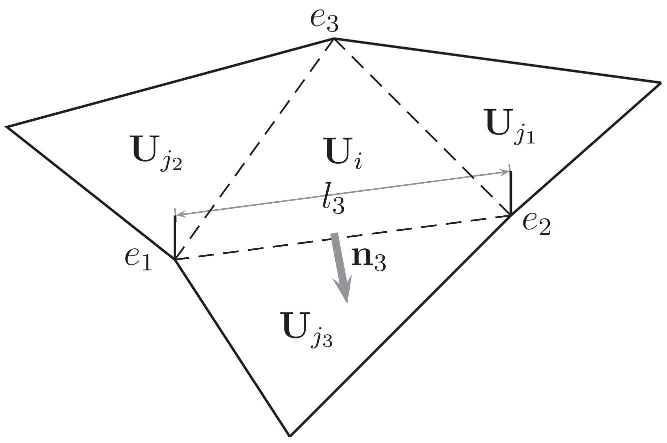

Fig. 1. Cell parameters.

2. Mathematical model and numerical scheme

2.1. Mathematical model

The mathematical model considered for reproducing the landslides phenomenon is based on the shallow flow equations, where the general three-dimensional conservation laws are depth averaged. The pressure distribution is considered hydrostatic and as frictional terms, only Coulomb type friction forces are assumed (Juez et al., 2013). Bearing in mind these hypothesis, the 2D equations are written in global coordinates as follows:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} + \frac{\partial \mathbf{G}(\mathbf{U})}{\partial y} = \mathbf{S}_\tau + \mathbf{S}_b \tag{1}$$

where

$$\mathbf{U} = (h, hu, hv)^T \tag{2}$$

are the conserved variables with h representing granular material depth parallel to the z coordinate and (u,v) the depth averaged components of the velocity vector. The fluxes are given by

$$\mathbf{F} = \left(hu,\ hu^2 + \tfrac{1}{2}g_\psi h^2,\ huv\right)^T$$
$$\mathbf{G} = \left(hv,\ huv,\ hv^2 + \tfrac{1}{2}g_\psi h^2\right)^T \tag{3}$$

with $g_\psi = g\cos^2\psi$ and ψ the direction cosine of the bed normal with respect to the vertical axis. The physical basis of this gravity projection is explained in detail in Juez et al. (2013) and it is of utmost importance for keeping accurate numerical predictions when the simulation involves the presence of steep slopes and consequently, the direction cosines become relevant.

The term $\mathbf{S}_\tau$ notes the frictional effects in the bed, and is defined as

$$\mathbf{S}_\tau = \left(0,\ -\frac{\tau_{b,x}}{\rho},\ -\frac{\tau_{b,y}}{\rho}\right)^T \tag{4}$$

with $\tau_{b,x}$, $\tau_{b,y}$ the bed shear stress in the x and y directions respectively and ρ the density of the granular mass. In this work, only dense granular flows are considered, therefore, the main rheological properties are governed by the frictional forces. These interactions between the sand grains are computed by means of the Coulomb law. This formula is based on the internal friction angle of the material, θ_b.

On the other hand, the term $\mathbf{S}_b$ is defined for gathering the information relative to the pressure force exerted over the bottom.

Thanks to the hyperbolic character of (1) it is possible to obtain a Jacobian matrix, $\mathbf{J}_\mathbf{n}$, which is built by means of the flux normal to a direction given by the unit vector $\mathbf{n}$, $\mathbf{E}_\mathbf{n} = \mathbf{F}n_x + \mathbf{G}n_y$,

$$\mathbf{J}_\mathbf{n} = \frac{\partial \mathbf{E}_\mathbf{n}}{\partial \mathbf{U}} = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}n_x + \frac{\partial \mathbf{G}}{\partial \mathbf{U}}n_y \tag{5}$$

whose components are

$$\mathbf{J}_n = \begin{pmatrix} 0 & n_x & n_y \\ (g_\psi h - u^2)n_x - uvn_y & vn_y + 2un_x & un_y \\ (g_\psi h - v^2)n_y - uvn_x & vn_x & un_x + 2vn_y \end{pmatrix} \tag{6}$$

The eigenvalues of this Jacobian matrix constitute the basis of the upwind technique which is detailed in the next subsection.

2.2. Numerical scheme

System in (1) is integrated in a grid cell Ω_i and the Gauss theorem is applied, being the normal vector $\mathbf{n}$ outward to the cell Ω_i, as displayed in Fig. 1

$$\frac{\partial}{\partial t}\int_{\Omega_i} \mathbf{U}\, d\Omega + \oint_{\partial\Omega_i} \mathbf{E}_\mathbf{n}\, dl = \int_{\Omega_i} (\mathbf{S}_\tau + \mathbf{S}_b)\, d\Omega \tag{7}$$

The second integral in (7) can be explicitly expressed as a sum over the cell edges:

$$\frac{\partial}{\partial t}\int_{\Omega_i} \mathbf{U}\, d\Omega + \sum_{k=1}^{NE} (\mathbf{E}_n)_k l_k = \sum_{k=1}^{NE} \mathbf{S}_{\mathbf{n}\tau} l_k + \sum_{k=1}^{NE} \mathbf{S}_{\mathbf{n}b} l_k \tag{8}$$

where l_k is the corresponding edge length and $\mathbf{S}_{\mathbf{n}b}$ and $\mathbf{S}_{\mathbf{n}_\tau}$ are suitable integrals of the bed slope and friction source terms (Juez et al., 2013):

$$\mathbf{S}_{\mathbf{n}b} = \left(0,\ -g_\psi h\frac{\partial z}{\partial x}n_x,\ -g_\psi h\frac{\partial z}{\partial y}n_y\right)^T \tag{9}$$

$$\mathbf{S}_{\mathbf{n}\tau} = \left(0,\ \rho g_\psi h\tan\theta_b n_x,\ \rho g_\psi h\tan\theta_b n_y\right)^T \tag{10}$$

where n_x and n_y are the components of the unit vector $\mathbf{n}$ of each edge of each computational cell.

The numerical scheme is constructed by defining a local linearization in terms of an approximate Jacobian matrix $\tilde{\mathbf{J}}$ at each k edge between neighboring cells defined through the normal flux $\mathbf{E}_\mathbf{n}$

$$(\delta\mathbf{E}_\mathbf{n})_k = \tilde{\mathbf{J}}_{\mathbf{n},k}\delta\mathbf{U}_k \tag{11}$$

with $\delta(\mathbf{E}_\mathbf{n})_k = (\mathbf{E}_j - \mathbf{E}_i)_{\mathbf{n}_k}$, $\delta\mathbf{U}_k = \mathbf{U}_j - \mathbf{U}_i$, and $\mathbf{U}_i$ and $\mathbf{U}_j$ the initial values at cells i and j sharing edge k.