Advances in Engineering Software 78 (2014) 1–15

Contents lists available at ScienceDirect

Advances in Engineering Software

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

An optimized GPU implementation of a 2D free surface simulation model on unstructured meshes

A. Lacasta *, M. Morales-Hernández, J. Murillo, P. García-Navarro

LIFTEC-EINA, CSIC-Universidad Zaragoza, Spain

ARTICLE INFO

Article history:
Received 18 December 2013
Received in revised form 3 June 2014
Accepted 17 August 2014
Available online 14 September 2014

Keywords:
GPU
Finite volume methods
Unsteady flow
Unstructured meshes
Dry/wet boundaries
CUDA
High performance computing

ABSTRACT

This work is related with the implementation of a finite volume method to solve the 2D Shallow Water Equations on Graphic Processing Units (GPU). The strategy is fully oriented to work efficiently with unstructured meshes which are widely used in many fields of Engineering. Due to the design of the GPU cards, structured meshes are better suited to work with than unstructured meshes. In order to overcome this situation, some strategies are proposed and analyzed in terms of computational gain, by means of introducing certain ordering on the unstructured meshes. The necessity of performing the simulations using unstructured instead of structured meshes is also justified by means of some test cases with analytical solution.

1. Introduction

Physically based simulations of complex systems usually require large computational facilities to be completed in a reasonable time. Moreover when the simulated phenomenon is unsteady and based on a dynamical estimation of the updating time step, the computational performance is an important topic to be taken into account. One of the most widespread strategies to reduce the computational cost is the use of parallel techniques, involving a suitable number of processors. Since CPU frequencies seem to be reaching their maximum capacity [1], nowadays Many-Core parallel techniques appear to be an interesting option.

In recent years, Graphic Processing Unit (GPU) has been used to accelerate the calculations because of its inherent vector-oriented designing. This paradigm is known as General-Purpose Computing on Graphics Processing Unit (GPGPU) and it is widely used for a very large range of applications in CFD such as [2–5] as well as other environmental applications such [6]. In the present work, special attention is paid to the application of these GPUs to unsteady flows of interest in hydraulics. Shallow Water models in particular are widely used to simulate surface geophysical flows. These situations usually involve large size domains and long time scales. Practical applications require a compromise between spatial accuracy and computational efficiency. In order to achieve the necessary spatial resolution, rather fine grids become necessary in many cases requiring more data storage, increasing proportionally the number of operations and reducing the allowable time step size for explicit calculations. When, at the same time, a reasonable computational time is desired, the use of GPU codes is one of the options for computing large space and temporal domain problems.

The idea of accelerating the calculations in unsteady hydraulic simulation using multiple CPU was recently reported in [7,8] or [9] as well as using GPU in [10–12] or [5]. Although a very good compromise between number of CPUs used and performance is offered by the former option, the cost of using multiple CPU is significant due to the hardware investment and associated use. Alternatively, the GPU technology offers the performance of smaller clusters with less disbursement [13]. The main difficulty, and apparent drawback, when porting codes from CPU to GPU, is the cell order required by the GPU to process data efficiently. This drawback is not present when dealing with structured meshes due to the inherent order and a simple and efficient implementation is relatively easy to be obtained.

Despite the wide use of structured meshes, complex geometries for internal or external boundaries are problematic to be represented if not using unstructured meshes. Moreover, when dealing with topographic representation some recent works [14] have shown the benefit of using unstructured meshes in unsteady hydraulic simulations over irregular topography. The quality of

* Corresponding author.
E-mail addresses: alacasta@unizar.es (A. Lacasta), mmorales@unizar.es (M. Morales-Hernández), Javier.murillo@unizar.es (J. Murillo), pigar@unizar.es (P. García-Navarro).

http://dx.doi.org/10.1016/j.advengsoft.2014.08.007
0965-9978/

the numerical results is sensitive to the grid resolution. Hence grid refinement is clearly an option to modify the whole resolution. In that sense, adaptive grid refinement, readily available when using triangular unstructured meshes [15], designed to follow local bed variations or irregular boundaries can be very useful. The present work is motivated by the implementation in GPU of a code able to perform unsteady hydraulic simulations on variable density triangular unstructured meshes.

The performance of GPU based calculations with Double Precision (`double`) is lower than those that use Single Precision (`float`) [16,5]. In the particular case of the 2D Shallow Water Equations with source terms [17,18], the use of `float` is not always desirable. In fact, when simulating complex topography cases, wave propagation over dry beds represents a numerical challenge. The past experience with the dynamical stability control of such transient flows involving wet/dry fronts indicates that `double` precision is always required. All the performance analysis presented will deal with that kind of data.

In the first part of the text, the governing equations are outlined. They are followed by a description of the finite volume updating scheme used. Then, the most relevant general aspects of the implementation in GPU are identified. The particular difficulties encountered when dealing with triangular unstructured meshes and some improvements to overcome them are detailed in the following section. Finally, they are applied to two test cases in order to prove their behaviour when using unstructured meshes.

2. Mathematical model/governing equations

The two-dimensional Shallow Water Equations (SWE), which represent depth averaged mass and momentum conservation, can be written as follows:

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla}\mathbf{E} = \mathbf{H} \tag{1}$$

where

$$\mathbf{U} = (h, q_x, q_y)^T \tag{2}$$

are the conserved variables with h representing the water depth, $q_x = hu$, $q_y = hv$ and $\mathbf{u} = (u, v)$ the depth averaged velocity vector along the (x, y) coordinates respectively. The fluxes of these variables are $\mathbf{E} = (\mathbf{F}, \mathbf{G})$ given by:

$$\mathbf{F} = \left(q_x, \frac{q_x^2}{h} + \frac{1}{2}gh^2, \frac{q_x q_y}{h}\right)^T, \quad \mathbf{G} = \left(q_y, \frac{q_x q_y}{h}, \frac{q_y^2}{h} + \frac{1}{2}gh^2\right)^T \tag{3}$$

where g is the acceleration due to the gravity.

The source terms of the system are the bed slope and the friction terms:

$$\mathbf{H} = \left(0, -gh\frac{\partial z}{\partial x} - \frac{\tau_{b,x}}{\rho_w}, -gh\frac{\partial z}{\partial y} - \frac{\tau_{b,y}}{\rho_w}\right)^T \tag{4}$$

where $\tau_{b,x}$ and $\tau_{b,y}$ are the components of the bed friction stress and ρ_w is the water density. These friction losses in both (x, y) axis are written in terms of the Manning's roughness coefficient n:

$$\frac{\tau_{b,x}}{\rho_w} = gh\frac{n^2 u\sqrt{u^2+v^2}}{h^{4/3}}, \quad \frac{\tau_{b,y}}{\rho_w} = gh\frac{n^2 v\sqrt{u^2+v^2}}{h^{4/3}} \tag{5}$$

3. Numerical scheme

The numerical resolution of system (1) can be obtained by means of the first order upwind finite volume scheme. Integrating in a volume or grid cell Ω the numerical scheme can be expressed compactly:

$$\frac{\partial}{\partial t}\int_\Omega \mathbf{U}\,d\Omega + \sum_{k=1}^{N_E}(\delta\mathbf{E} - \mathbf{T})_k \cdot \mathbf{n_k} l_k = 0 \tag{6}$$

It is possible to define a Jacobian matrix $\widetilde{\mathbf{J}}_k$ of the normal flux at each edge as a result of the local linearization

$$\delta(\mathbf{E}\cdot\mathbf{n})_k = \widetilde{\mathbf{J}}_k \delta\mathbf{U}_k \tag{7}$$

and to diagonalize it in terms of matrices $\widetilde{\mathbf{P}}$ and $\widetilde{\mathbf{\Lambda}}$, formed by its eigenvalues $\tilde{\lambda}_m$ and eigenvectors $\tilde{\mathbf{e}}_m$ respectively:

$$\widetilde{\mathbf{P}} = \begin{pmatrix} 1 & 0 & 1 \\ \tilde{u} - \tilde{c}n_x & -\tilde{c}n_y & \tilde{u} + \tilde{c}n_x \\ \tilde{v} - \tilde{c}n_y & \tilde{c}n_x & v + \tilde{c}n_y \end{pmatrix}, \quad \widetilde{\mathbf{\Lambda}} = \begin{pmatrix} \tilde{\lambda}_1 & 0 & 0 \\ 0 & \tilde{\lambda}_2 & 0 \\ 0 & 0 & \tilde{\lambda}_3 \end{pmatrix},$$
$$\tilde{\mathbf{e}}_1 = \begin{pmatrix} 1 \\ \tilde{u} - \tilde{c}n_x \\ \tilde{v} - \tilde{c}n_y \end{pmatrix}, \quad \tilde{\mathbf{e}}_2 = \begin{pmatrix} 0 \\ -\tilde{c}n_y \\ \tilde{c}n_x \end{pmatrix}, \quad \tilde{\mathbf{e}}_3 = \begin{pmatrix} 1 \\ \tilde{u} + \tilde{c}n_x \\ \tilde{v} + \tilde{c}n_y \end{pmatrix}, \tag{8}$$
$$\tilde{\lambda}_1 = \tilde{\mathbf{u}}\cdot\mathbf{n} - \tilde{c}, \quad \tilde{\lambda}_2 = \tilde{\mathbf{u}}\cdot\mathbf{n}, \quad \tilde{\lambda}_3 = \tilde{\mathbf{u}}\cdot\mathbf{n} + \tilde{c}$$

where $\tilde{\mathbf{u}}\cdot\mathbf{n} = \tilde{u}n_x + \tilde{v}n_y$. The definition of the averaged variables is as follows [19]:

$$\tilde{u}_k = \frac{u_i\sqrt{h_i} + u_j\sqrt{h_j}}{\sqrt{h_i} + \sqrt{h_j}}, \quad \tilde{v}_k = \frac{v_i\sqrt{h_i} + v_j\sqrt{h_j}}{\sqrt{h_i} + \sqrt{h_j}}, \quad \tilde{c}_k = \sqrt{g\frac{h_i + h_j}{2}} \tag{9}$$

The difference across the edge k can be projected onto the eigenvectors basis [18]:

$$\delta\mathbf{U}_k = \mathbf{U}_j - \mathbf{U}_i = \widetilde{\mathbf{P}}_k\widetilde{\mathbf{A}}_k \tag{10}$$

where $\widetilde{\mathbf{A}}_k = (\tilde{\alpha}_1, \tilde{\alpha}_2, \tilde{\alpha}_3)_k^T$ contains the set of wave strengths. Following the same procedure with the source terms [18]

$$(\widetilde{\mathbf{T}}\mathbf{n})_k = \widetilde{\mathbf{P}}_k\widetilde{\mathbf{B}}_k \tag{11}$$

where $\widetilde{\mathbf{B}}_k = (\tilde{\beta}_1, \tilde{\beta}_2, \tilde{\beta}_3)_k^T$ contains the source strengths.

More information about the values of the wave and the source strengths as well as the entropy fix can be found in [18]. The contributions due to the fluxes and the source terms are combined in a compact expression

$$(\tilde{\gamma}_m)_k = \left(\tilde{\lambda}_m\tilde{\alpha}_m - \tilde{\beta}_m\right)_k \tag{12}$$

The 2D numerical upwind explicit scheme is formulated using only the contributions that arrive to the cell:

$$\tilde{\gamma}_k^- = \frac{1}{2}\left[1 - \mathrm{sign}\left(\tilde{\lambda}_k\right)\right]\tilde{\gamma}_k \tag{13}$$

so that the finite volume approach for the updating of a single cell of area Ω_i is [20]:

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t}{\Omega_i}\sum_{k=1}^{N_E}\sum_{m=1}^{3}(\tilde{\gamma}_m^-\tilde{\mathbf{e}}_m l)_k^n \tag{14}$$

Considering that in the explicit scheme (14) each k cell edge is used to deliver information between a pair of neighbouring cells of different size, the time step size compatible with numerical stability is limited by

$$\Delta t \leqslant \Delta t^{\tilde{\lambda}} \quad \Delta t^{\tilde{\lambda}} = \frac{\min(\chi_i, \chi_j)}{\max_{m=1,2,3}|\tilde{\lambda}^m|} \tag{15}$$

so that the following dimensionless quantity is defined

$$CFL = \frac{\Delta t}{\Delta t^{\tilde{\lambda}}} \leqslant 1 \tag{16}$$

to control the numerical stability of the method.