Computers and Fluids 136 (2016) 371–383

Contents lists available at ScienceDirect

Computers and Fluids

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

A GPU accelerated adjoint-based optimizer for inverse modeling of the two-dimensional shallow water equations

A. Lacasta*, P. García-Navarro

LIFTEC,CSIC-Universidad Zaragoza, Maria de Luna 3, 50018 Zaragoza, Spain

ARTICLE INFO

Article history:
Received 28 April 2016
Revised 23 June 2016
Accepted 27 June 2016
Available online 28 June 2016

Keywords:
Adjoint method
Shallow water equations
Free-surface flow
Quasi-Newton methods
Checkpointing
GPU computing

ABSTRACT

The increasing computational capacity of current computing technologies makes feasible the application of predictive high-resolution mathematical models for studying physical phenomena. Additionally, this can be enhanced if those methods are incorporated to some optimization method in order to perform inverse modeling. In this paper, an optimization procedure based on the adjoint equations is used for the reconstruction of information in a 2D Shallow Water model previously developed and proved to be fast, robust and accurate. The continuous adjoint approach used for the evaluation of the gradient that is introduced in a gradient-based optimizer. Furthermore, the computation of both physical and adjoint systems is accelerated by the use of GPU programming. Even though notable speed-ups are achieved with this technique they are only possible in small to medium size grids due to memory limitations. A novel checkpointing strategy is proposed to allow data handling in these devices hence offering the possibility to overcome that limitation.

1. Introduction

Nowadays, the computational resources available for simulation purposes open new possibilities that previously were prohibitive. The increasingly efficient implementations of numerical tools allow the application of accurate mathematical models for prediction of physical problems in many engineering disciplines. Moreover, this increasing predictive capability allows their application for detailed response studies. A special field of application is computational fluid dynamics where, using different approaches, the flow field is characterized providing useful information [1].

Another perspective of predictive tools is their application for inverse modeling. In fact, there are many developments relative to inverse design based on high-fidelity simulations with applications on noise reduction in car aerodynamics [2], design optimization of cavitating flow with vapor minimization [3] and the classical study of drag and lift improvements by means of shape optimization [4]. Each inverse problem is concerned with the minimization of some functional. In other words, the optimization problem can be formulated as a minimization problem stated as

$$\begin{array}{lll} \underset{\mathbf{U}}{\text{minimize}} & \mathcal{J}(\mathbf{U},\mathbf{U}^*,\omega) & \\ \text{subject to} & \mathcal{W}(\mathbf{U},\omega)=0 & \text{in}\,\Omega\times[0,T] \end{array} \tag{1}$$

where the performance function is frequently written as a squared deviation in terms of an objective quantity, $\mathbf{U}^*$ as

$$\mathcal{J}(\mathbf{U},\mathbf{U}^*,\omega) = \frac{1}{2}\int_0^T\int_\Omega \mathcal{E} dSdt = \frac{1}{2}\int_0^T\int_\Omega (\mathbf{U}(\omega)-\mathbf{U}^*)^2 dSdt \tag{2}$$

It establishes the error in time $[0, T]$ and space Ω produced by using a set of parameters, geometry or particular conditions ω in a model $\mathcal{W}$ which produces the physical variables $\mathbf{U}$. Inverse design is then the process to find those optimal parameters ω^* that allow to obtain the closest solution $\mathbf{U}$ to $\mathbf{U}^*$ by modifying such variable conditions to be controlled. Difficulties appear when the relationship between the functional (2) and the controlled parameters is not known. In particular, when the controlled variable is related to flow conditions, the optimal solution may not be easily obtained. Usually, the mathematical model $\mathcal{W}$ that governs the physical phenomena, is a set of non-linear PDEs. A solution is the development of the adjoint equations of the mathematical model to obtain the sensitivities of the functional $\mathcal{J}$ to the variables that are controlled [5]. In the present work the governing equations are the Shallow Water Equations (SWE), widely used for the modelization of free surface flows when horizontal scales are larger than vertical. The range of problems where they can be applied covers simulation of flooding events [6–8], open-channel flow [9–11] and even the interaction with sedimentary processes [12,13].

* Corresponding author.
E-mail address: alacasta@unizar.es (A. Lacasta).

http://dx.doi.org/10.1016/j.compfluid.2016.06.024
0045-7930/© 2016 Elsevier Ltd. All rights reserved.

There are several techniques that can be applied for the numerical resolution of these type of equations such as [14–19]. The stability, well-balancing and accuracy of the solution provided by these methods usually implies an enormous computational effort that may result a drawback when dealing with practical applications. In order to achieve a trade-off between numerical quality and computational requirements, a well-balanced first order Upwind (FOU) scheme such as [20] is used in this work for solving both, the physical problem and the adjoint equations. With independence of the numerical solver chosen, a large amount of arithmetical operations are required to reach the solution.

A recent advance in computing is the design of hardware accelerators to perform faster mathematical operations following the SIMT paradigm. The most common approach is the adaptation of the Graphical Processing Units (GPU) for dealing not only with graphics but with general purpose computing [21,22]. This new computational scheme is very recent and it was named General Purpose Graphical Procesing Units (GPGPU) computing. It allows developers to exploit the computational capabilities of these many-core devices by developing massively parallel algorithms. In the case of the numerical solvers in general, and their application to 2D free surface modeling in particular, explicit finite volume schemes are well suited to work efficiently on these processors. Some implementations such as [23–27] deal with them. While originally these devices reached their maximum performance on structured grids, recent advances extend their efficient application to unstructured meshes [28–30]. The adaptation of the GPU computation to the combined problem of 2D physical equations and 2D adjoint equations is one of the original parts in the present work.

The unsteady character of the physical equations leads to time-dependent information for the adjoint equations which, in fact, have the same unsteady character. Consequently, as the adjoint solver requires information of the flow variables previously computed, data storage issues may appear when dealing with large problems. The basic option is to store all the information provided by the physical model and then use it to compute the adjoint equations. However, this option has the inconvenience of the enormous memory requirement. In addition, and when dealing with GPU computing, the memory is fixed and not expandable. Therefore, the amount of memory used should be less or equal to that available in the GPU. At this moment, this size usually goes from 6 to 24 GB. So, in the case of large physical and/or temporal domains, it represents an important restriction. In this work, a checkpointing strategy is used and deeply detailed, making feasible the application of the adjoint method to calculate the 2D SWE sensitivities on GPUs. This represents a novelty in this field.

This work is structured as follows: first, in Section 2, the set of physical equations and the numerical solver used for their integration is presented. Next, details regarding the adjoint equations used in this work as well as the numerical method used for their integration are explained. Section 4 deals with how the optimization problem is stated and a synthetic test case is proposed for illustrating the technique. In the same section a GPU implementation is detailed and compared with a CPU implementation. As a result, the memory appears to be a limitation that is next solved providing details of the implementation of a checkpoint strategy. Section 5 contains an application which aims at recovering a previously computed water depth by obtaining the inlet hydrograph used to compute it. Section 6 outlines the conclusions obtained in the work.

2. Physical model: shallow water equations

Two dimensional free surface flows are commonly well-described by means of the two-dimensional version of the St. Venant equations, also known as Shallow Water Equations (SWE). They are obtained depth-integrating the Navier-Stokes equations using the hydrostatic approximation under the assumption of horizontal length scale larger than the vertical length scale. The system of equations $\mathcal{W} = \{\mathcal{W}_h, \mathcal{W}_{q_x}, \mathcal{W}_{q_y}\}$ represents mass and momentum conservation

$$\begin{aligned} \mathcal{W}_h &: \frac{\partial h}{\partial t} + \frac{\partial (q_x)}{\partial x} + \frac{\partial (q_y)}{\partial y} = i_e \\ \mathcal{W}_{q_x} &: \frac{\partial (q_x)}{\partial t} + \frac{\partial}{\partial x}\left(hu^2 + \frac{1}{2}gh^2\right) + \frac{\partial (huv)}{\partial y} = gh(S_{0x} - S_{fx}) \\ \mathcal{W}_{q_y} &: \frac{\partial (q_y)}{\partial t} + \frac{\partial (huv)}{\partial x} + \frac{\partial}{\partial y}\left(hv^2 + \frac{1}{2}gh^2\right) = gh(S_{0y} - S_{fy}) \end{aligned} \quad (3)$$

where h is water depth, (hu, hv) are unitary discharges in x and y axis respectively, (u, v) are depth averaged velocity vector components and g is the gravitational acceleration aligned with z axis.

The right hand side in (3) includes source terms. For the mass equation i_e is an effective mass contribution rate (i.e. rain, evaporation or source). The momentum equations include the bed slopes (S_{0x}, S_{0y}) in terms of the bed level z_b as

$$S_{0x} = -\frac{\partial z_b}{\partial x} \qquad S_{0y} = -\frac{\partial z_b}{\partial y} \quad (4)$$

and friction losses, where S_{fx} and S_{fy} are the (dimensionless) components of the friction slope written in terms of the Gauckler-Manning's roughness coefficient n:

$$S_{fx} = \frac{n^2 u\sqrt{u^2+v^2}}{h^{4/3}}, \qquad S_{fy} = \frac{n^2 v\sqrt{u^2+v^2}}{h^{4/3}} \quad (5)$$

2.1. Numerical solver

For simplicity, system (3) is compactly written as:

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \mathbf{E} = \mathbf{R} \quad (6)$$

being $\mathbf{U} = (h, q_x, q_y)^T$ the conserved variables, $\mathbf{E} = (\mathbf{F}, \mathbf{G})$ with

$$\mathbf{F} = \left(q_x,\ \frac{q_x^2}{h} + \frac{1}{2}gh^2,\ \frac{q_x q_y}{h}\right)^T, \qquad \mathbf{G} = \left(q_y,\ \frac{q_x q_y}{h},\ \frac{q_y^2}{h} + \frac{1}{2}gh^2\right)^T \quad (7)$$

and the source term $\mathbf{R}$ defined as follows

$$\mathbf{R} = \left(i_e,\ gh(S_{0x} - S_{fx}),\ gh(S_{0y} - S_{fy})\right)^T \quad (8)$$

In order to obtain a numerical solution from a finite volume approach, Eq. (6) is integrated over each computational cell Ω_i with area A_i:

$$\frac{\partial}{\partial t}\int_{\Omega_i} \mathbf{U}\, d\Omega + \int_{\Omega_i} (\vec{\nabla} \cdot \mathbf{E})\, d\Omega = \int_{\Omega_i} \mathbf{R}\, d\Omega \quad (9)$$

It is possible to define a Jacobian matrix $\mathbf{J_n}$ of the normal flux $\mathbf{E \cdot n}$ and to diagonalise it in terms of matrices $\mathbf{\Lambda}$ and $\mathbf{P}$ where the eigenvalues and eigenvectors play an important role:

$$\mathbf{J_n} = \frac{\partial(\mathbf{E \cdot n})}{\partial \mathbf{U}} = \mathbf{P}\,\mathbf{\Lambda}\,\mathbf{P}^{-1} \quad (10)$$

where $\mathbf{n} = (n_x, n_y)$ is the outward unit normal vector to Ω_i. Roe's linearization [31] allows us to define locally an approximate matrix $\tilde{\mathbf{J}}_\mathbf{n}$ at each cell edge k whose eigenvalues $\tilde{\lambda}^m$ and eigenvectors $\tilde{\mathbf{e}}^m$ can be used to express the differences in vector $\mathbf{U}$ [32]:

$$\delta \mathbf{U}_k = \mathbf{U}_i - \mathbf{U}_j = \sum_{m=1}^{3} (\tilde{\alpha}\ \tilde{\mathbf{e}})_k^m \quad (11)$$

where i and j are the indexes of the cells sharing edge k and $\tilde{\alpha}_k^m$ is the wave strength. This enables:

$$\delta(\mathbf{E \cdot n})_k = \tilde{\mathbf{J}}_\mathbf{n} \delta \mathbf{U}_k = \tilde{\mathbf{J}}_\mathbf{n} \sum_{m=1}^{3} (\tilde{\alpha}\ \tilde{\mathbf{e}})_k^m \quad (12)$$