Computers and Fluids 161 (2018) 107–120

Contents lists available at ScienceDirect

Computers and Fluids

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

GPU driven finite difference WENO scheme for real time solution of the shallow water equations

P. Parna*, K. Meyer, R. Falconer

School of Design and Informatics, Abertay University, Bell Street, Dundee, Scotland DD1 1HG, United Kingdom

ARTICLE INFO

Article history:
Received 15 May 2017
Revised 24 October 2017
Accepted 19 November 2017
Available online 21 November 2017

Keywords:
Shallow water
Picard integral
WENO
Finite difference
GPGPU

ABSTRACT

The shallow water equations are applicable to many common engineering problems involving modelling of waves dominated by motions in the horizontal directions (e.g. tsunami propagation, dam breaks). As such events pose substantial economic costs, as well as potential loss of life, accurate real-time simulation and visualization methods are of great importance. For this purpose, we propose a new finite difference scheme for the 2D shallow water equations that is specifically formulated to take advantage of modern GPUs. The new scheme is based on the so-called Picard integral formulation of conservation laws combined with Weighted Essentially Non-Oscillatory reconstruction. The emphasis of the work is on third order in space and second order in time solutions (in both single and double precision). Further, the scheme is well-balanced for bathymetry functions that are not surface piercing and can handle wetting and drying in a GPU-friendly manner without resorting to long and specific case-by-case procedures. We also present a fast single kernel GPU implementation with a novel boundary condition application technique that allows for simultaneous real-time visualization and single precision simulations even on large ($>2000\times2000$) grids on consumer-level hardware - the full kernel source codes are also provided online at https://github.com/pparna/swe_pifweno3.

1. Introduction

The shallow water equations (SWE) are a set of hyperbolic partial differential equations that arise from the more general inviscid Navier–Stokes equations (also referred to as the Euler equations) under the assumption that the vertical water depth h_0 is much smaller than the horizontal length scale L of the waves, i.e. $h_0 \ll L$, and hence the vertical acceleration is considered negligible [1, p.89,91]. Such a simplification is especially beneficial from a computation point of view as the arising equations result in dimensional reduction from $\mathbb{R}^3$ to $\mathbb{R}^2$, while still describing the evolution of a three dimensional fluid surface. As a result, the equations are often used for real-time flood prediction [2], simulations of tsunami propagation and inundation [3], modelling of dam breaks [4] and even computer graphics animations of water [5].

The shallow water equations in 2D conservation form are given as:

$$\frac{\partial \boldsymbol{U}(x,y,t)}{\partial t} + \frac{\partial \boldsymbol{F}(\boldsymbol{U}(x,y,t))}{\partial x} + \frac{\partial \boldsymbol{G}(\boldsymbol{U}(x,y,t))}{\partial y} = \boldsymbol{S}(b(x,y)) \qquad (1)$$

where $\boldsymbol{U}$ is the vector of conserved variables (mass and momentum), $\boldsymbol{F}$ and $\boldsymbol{G}$ the x and y directional fluxes, respectively; $\boldsymbol{S}$ is the source term due to topography underneath the water surface (also referred to as the bathymetry). In this work, we are interested in modelling the time-independent source term (i.e. only static bathymetry functions are considered). The vectors themselves are given as:

$$\boldsymbol{U} = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix};\ \boldsymbol{F} = \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{pmatrix};\ \boldsymbol{G} = \begin{pmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{pmatrix};\ \boldsymbol{S} = \begin{pmatrix} 0 \\ -gh\frac{\partial b}{\partial x} \\ -gh\frac{\partial b}{\partial y} \end{pmatrix} \qquad (2)$$

where h is the water height, u and v are the horizontal velocities, b is the underlying topography function and g the gravitational constant. It will also be useful to consider the total surface elevation $\eta = b + h$ as illustrated in Fig. 1. The conservation law form of the shallow water equations lends itself to many well-known numerical methods - the equations have been solved by several authors using classic finite difference schemes, such as the MacCormack method [3,6,7], alongside specifically designed finite volume schemes such as the central-upwind scheme by Kurganov and Petrova [4,8,9]. These schemes generally have a fixed order of

* Corresponding author.
E-mail address: p.parna@phys-gfx.net (P. Parna).
URL: http://www.phys-gfx.net (P. Parna)

https://doi.org/10.1016/j.compfluid.2017.11.012
0045-7930/

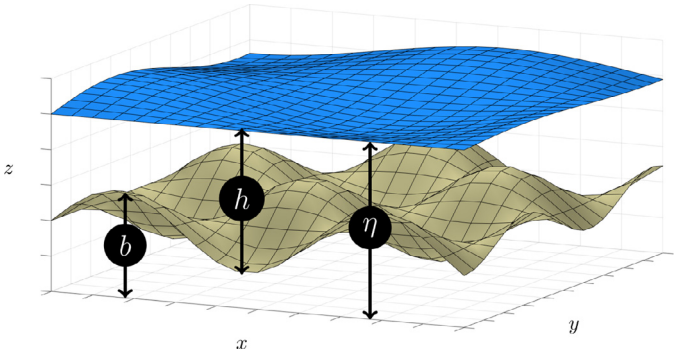

Fig. 1. Spatial setup for the shallow water equations.

accuracy - often the spatial order of accuracy is formulated to second order, with no straight-forward way of increasing it. Furthermore, these schemes commonly use the method of lines (MOL) approach[1] with Runge–Kutta integration for timestepping, leading to multi-step implementations requiring complex flux evaluations at every stage. One of the most common arbitrary-order finite difference methods is based on the Weighted Essentially Non-Oscillatory (WENO) reconstruction procedures [10]. Recently, the application of WENO to conservation laws was further modified to incorporate time-averaged flux functions by Seal et al. [11] (called the Picard Integral Formulation of WENO (PIFWENO) schemes), who also successfully applied the idea to the compressible Euler equations [12]. The compact nature of the PIFWENO formulation (specifically the Taylor timestepping variation) makes it a particularly interesting candidate for high-accuracy real-time simulations and as such the following paper provides a complete derivation and description of a PIFWENO-type scheme for the 2D shallow water equations that is third order accurate in space and second order accurate in time, well-balanced (the flux terms balance the source term [9]) and is capable of retaining the positivity of the water depth, thus allowing for simulations with dry zones. Further, we present an optimized, single pass GPU implementation of the scheme capable of achieving real-time performance on various grid sizes using either single or double precision floating point arithmetic.

The rest of the paper is organized as follows: in Section 2 a detailed mathematical description of the numerical scheme is presented, followed by an overview of the practical implementation describing the employed optimization strategies in Section 3. In Section 4 the numerical accuracy and the capability of the scheme to model complex flows with moving shorelines are investigated, alongside verification of the well-balanced property and grid convergence rates. Furthermore, the performance of the GPU implementation for real-time computation and rendering is assessed using both single and double precision arithmetic. Finally, increasing the scheme's spatial and temporal orders are discussed, followed by conclusions of the undertaken research in Section 5.

2. Numerical method

2.1. Picard integral formulation for SWE

The Picard integral formulation (PIF) of the SWE can be defined by integrating Eq. (1) over the interval $t \in [t^n, t^{n+1}]$ [11] (subscripts denote derivatives while superscripts denote the time level):

$$\boldsymbol{U}^{n+1} = \boldsymbol{U}^n - \Delta t \widetilde{\boldsymbol{F}}^n_x - \Delta t \widetilde{\boldsymbol{G}}^n_y + \Delta t \boldsymbol{S}^n \tag{3}$$

[1] A partial differential equation is transformed into multiple ordinary differential equations via initial semi-discretization in space - this results in n ordinary differential equations in time where n is the total number of grid cells.

where $\widetilde{\boldsymbol{F}}^n$ and $\widetilde{\boldsymbol{G}}^n$ are the time-averaged fluxes defined as:

$$\begin{aligned} \widetilde{\boldsymbol{F}}^n &= \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \boldsymbol{F}^n(\boldsymbol{U})\, dt \\ \widetilde{\boldsymbol{G}}^n &= \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \boldsymbol{G}^n(\boldsymbol{U})\, dt. \end{aligned} \tag{4}$$

The conservative finite difference discretization of Eq. (3) can be written as:

$$\begin{aligned} \boldsymbol{U}^{n+1}_{i,j} = \boldsymbol{U}^n_{i,j} &- \frac{\Delta t}{\Delta x}\left(\widehat{\boldsymbol{F}}^n_{i+1/2,j} - \widehat{\boldsymbol{F}}^n_{i-1/2,j}\right) \\ &- \frac{\Delta t}{\Delta y}\left(\widehat{\boldsymbol{G}}^n_{i,j+1/2} - \widehat{\boldsymbol{G}}^n_{i,j-1/2}\right) + \Delta t \boldsymbol{S}^n_{i,j} \end{aligned} \tag{5}$$

where the values of $\widehat{\boldsymbol{F}}^n$ and $\widehat{\boldsymbol{G}}^n$ at the cell edges are given by the WENO reconstruction procedure of the time-averaged fluxes $\widetilde{\boldsymbol{F}}^n$ and $\widetilde{\boldsymbol{G}}^n$, respectively.

The time averaged fluxes can be approximated via Taylor expansion of the fluxes centred at $t = t^n$ and then integrating the result with respect to t [11] (henceforward, time level n dropped for convenience):

$$\begin{aligned} \widetilde{\boldsymbol{F}} &= \boldsymbol{F}(\boldsymbol{U}) + \frac{\Delta t}{2}\frac{d\boldsymbol{F}(\boldsymbol{U})}{dt} + \mathcal{O}(\Delta t^2) \\ \widetilde{\boldsymbol{G}} &= \boldsymbol{G}(\boldsymbol{U}) + \frac{\Delta t}{2}\frac{d\boldsymbol{G}(\boldsymbol{U})}{dt} + \mathcal{O}(\Delta t^2). \end{aligned} \tag{6}$$

Higher order approximations can be achieved by including more terms in the Taylor expansions. However, these require evaluations of Hessians and other higher order tensors which grow exponentially in size [11] - we found second order to be sufficient for our purposes. Note that the Hessian tensors of the flux functions for the SWE involve a scalar multiplier $1/h$ which further complicates simulations involving dry zones ($h = 0$). The temporal derivatives appearing in Eq. (6) can be expanded as:

$$\begin{aligned} \frac{d\boldsymbol{F}(\boldsymbol{U})}{dt} &= \frac{\partial \boldsymbol{F}(\boldsymbol{U})}{\partial \boldsymbol{U}}\frac{\partial \boldsymbol{U}}{\partial t} \\ \frac{d\boldsymbol{G}(\boldsymbol{U})}{dt} &= \frac{\partial \boldsymbol{G}(\boldsymbol{U})}{\partial \boldsymbol{U}}\frac{\partial \boldsymbol{U}}{\partial t} \end{aligned} \tag{7}$$

where $\partial \boldsymbol{F}/\partial \boldsymbol{U}$ and $\partial \boldsymbol{G}/\partial \boldsymbol{U}$ are the flux Jacobians. For the SWE these are [13, p.429]:

$$\frac{\partial \mathbf{F}}{\partial \mathbf{U}} = \begin{pmatrix} 0 & 1 & 0 \\ -u^2 + gh & 2u & 0 \\ -uv & v & u \end{pmatrix} \quad \frac{\partial \mathbf{G}}{\partial \mathbf{U}} = \begin{pmatrix} 0 & 0 & 1 \\ -uv & v & u \\ -v^2 + gh & 0 & 2v \end{pmatrix}. \tag{8}$$

Combining Eq. (7) with the Cauchy–Kowalewski procedure (using the original PDE in Eq. (1)) and plugging the results into Eq. (6) gives the final form of the time-averaged fluxes as:

$$\begin{aligned} \widetilde{\boldsymbol{F}}^n &= \boldsymbol{F}(\boldsymbol{U}) + \frac{\Delta t}{2}\frac{\partial \boldsymbol{F}(\boldsymbol{U})}{\partial \boldsymbol{U}}(\boldsymbol{S} - \boldsymbol{F}(\boldsymbol{U})_x - \boldsymbol{G}(\boldsymbol{U})_y) \\ \widetilde{\boldsymbol{G}}^n &= \boldsymbol{G}(\boldsymbol{U}) + \frac{\Delta t}{2}\frac{\partial \boldsymbol{G}(\boldsymbol{U})}{\partial \boldsymbol{U}}(\boldsymbol{S} - \boldsymbol{F}(\boldsymbol{U})_x - \boldsymbol{G}(\boldsymbol{U})_y). \end{aligned} \tag{9}$$

Any derivatives that appear in Eq. (9) are evaluated using simple central finite difference equations of order $k - 1$ where k is the order of the WENO reconstruction. Due to the extra Δt term, this approximation is sufficiently accurate (for any higher order time-averaged flux approximations, every higher order term can be evaluated with a consequently lower order approximation stencil, e.g. see [11] for an example of third order approximation).

2.2. WENO reconstruction

The core idea of the essentially non-oscillatory (ENO) reconstruction procedure [14] is to choose an approximation to the function to be reconstructed such that it is as smooth as possible in the candidate stencil used for the approximation. Weighted ENO