Water Resour Manage (2016) 30:261–277
DOI 10.1007/s11269-015-1160-2

Explicit GPU Based Second-Order Finite-Difference Modeling on a High Resolution Surface, Feather River, California

Owen T. Ransom[1] · **Bassam A. Younis**[1]

Received: 9 February 2015 / Accepted: 12 October 2015 /
Published online: 26 October 2015

Abstract An explicit second-order finite-difference computer model was developed and optimized for solution of the Shallow Water Equations. The model was applied to the Feather River below the Oroville Dam and Thermalito Afterbay near Gridley, California. Two versions of the computer model were constructed to run on either Central Processing Units or Graphical Processing Units, utilizing Fortran, C, C++, and the NVIDIA Compute Unified Device Architecture (CUDA) parallel computing platform. The underlying algorithm utilizes a structured grid and is capable of handling the wetting and drying of cells. It was developed with view to maximizing stability while maintaining accuracy, and allowing for flexibility of the computational domain. Comparisons with analytical and observed results showed the proposed methodology to be robust, accurate, and efficient. The models were applied to a section of the Feather River where observations of flow depths and volumetric flow rates are available for multiple flood events. The domain surface was partially developed using high-resolution photogrammetric data obtained through use of unmanned aerial vehicles. Runtimes and results were compared to the United States Bureau of Reclamations' implicit finite-volume numerical method and with field observation with generally good correspondence.

Keywords GPU computing · Explicit · Shallow-water · Transcritical Flow · TVD

1 Introduction

Most computational models currently in use for large-scale flood simulation utilize an implicit scheme for time integration so that the time-step size is not limited by stability

✉ Owen T. Ransom
oransom@ucdavis.edu

[1] Department of Civil and Environmental Engineering, University of California, One Shields Ave., Davis, CA, 95616, California

criteria. These methods can be prone to significant diffusion and generation of artificial velocity around areas of complex flow dynamics which can lead to inaccurate prediction of water depths and velocities and thus produce erroneous wave arrival times (Ransom and Younis 2015). On the other hand, explicit methods, precisely because the time-step size is limited by stability considerations, allow accurate simulations even within areas of complex or rapidly varied flow. Thus, using an explicit finite-difference method on hyperbolic equations while satisfying the appropriate stability constraints can produce the proper solution of the weak conditions, and capture discontinuities which can take the form of shocks or hydraulic jumps (Garcia and Kahawita 1986). The downside to explicit methods is obvious: adherence to the Courant-Friedrichs-Lewy (CFL) criteria leads to excessively long run times when simulations are performed for large domains or over extended periods. Given the geographic and hydraulic complexity of riverine domains, and the resolution that is necessary to properly resolve small side streams and hydraulic structures, runtimes for traditional explicit models can often be measured in days. This is often unacceptable in engineering practice.

A number of numerical and computational adaptations have been advanced in order to minimize runtime but not all have been satisfactory (Néelz and Pender 2013). Simplification of the governing equations either by reducing the dimensionality of the problem or by oversimplification of important physical processes can lead to a decrease in physical realism, while techniques such as local grid refinement or dynamic grid adaptation can cause a loss of conservation of mass and momentum (Smith and Liang 2013; Garcia and Kahawita 1986). Even then, these techniques generally speed up the simulations by only 2-3 fold (Brodtkorb et al. 2012). Distributed computing has proven effective in reducing runtimes but this invariably requires a large investment in hardware and is not ideal due to communication latency for information shared between cells. More recently, Graphics Processing Units (GPU) have been utilized in the place of parallelized Central Processing Units (CPU) or distributed computing to minimize computational runtimes. Largely, GPU based solutions of the Shallow Water Equations (SWE) have relied on the finite-volume method (Vacondio et al. 2014; Sastra and Brodtkorb 2012; de la Asuncion et al. 2013).

In this paper we present a two-dimensional implementation of a second-order accurate shock-capturing finite-difference scheme designed to run efficiently on both CPU and GPU. The underlying algorithm combines the explicit, second-order MacCormack predictor-corrector scheme with a Total Variation Diminishing (TVD) limiter adapted from (Liang et al. 2007). The TVD method is beneficial for its ability to maintain stability at shock interfaces with little effect on overall accuracy or computational resources. The method is constructed to be accurate and robust, while remaining computationally efficient and easily adaptable to changing flow conditions. Previous applications of explicit finite-difference methods with TVD modifications in unsteady natural flow regimes are found in (Liang et al. 2006). Another reason for the choice of the MacCormack scheme is because it is suited for use in parallel computing operations, since each successive set of calculations is based only on data from the previous time step and the method utilizes a compact solution stencil (Brodtkorb et al. 2012).

To demonstrate the utility of the developed model, simulations were performed for a reach of the Feather River, California where adjacent agricultural land is being considered for the installation of photovoltaic farms. Recent field studies have been undertaken in the area of interest to provide data suitable for model calibration and verification. An essential requirement for the accurate simulation of this flow requires an accurate and detailed definition of the surface involved. Large scale topology modeling is typically done using USGS Digital Elevation Modeling (DEM) data, which is generally limited to 10 meter contours.

This level of precision omits fine topographic data that may result in drastically different flow than what is observed. The surface created for this model was based upon four data sources: LiDAR data, unmanned aerial vehicle (UAV) gathered photogrammetric data, Real-Time Kinematic Global Positioning Satellite (RTK-GPS) data, and echo sounder obtained transects of the river bathymetry. Details are presented in Section 4.2.

The code has been applied to both analytical and naturally occurring complex flow problems that included wetting and drying and RVF. Comparison of run times between CPU only and CPU/GPU model runs for various situations, including the natural Feather River domain show that the overall increase in speed for the GPU code places it along side implicit codes for run time, while maintaining shock-capturing second-order estimations of water depth and velocity. The method presented is robust and simple to implement, and should allow for spatially expansive models to be run on consumer grade computer equipment. The following sections describe the construction of the algorithm and domain, and present the outcome of tests carried out to verify its accuracy and runtime differences between CPU and GPU implementations.

2 Governing Equations

In the computation of flows in complex domains using shock-capturing methods, numerical instabilities are often observed (Brodtkorb et al. 2012). These arise from a number of different sources, some being associated with the irregular topography that characterizes flows in natural systems, others due to inconsistent discretization of the flux-gradient and source terms in the governing equations. Following (Liang et al. 2006; Rogers et al. 2003), in order to more consistently discretize these terms, the deviatoric method is employed, where deviation from the still-water level and discharge are taken as the dependent variables. This method is less prone to numerically-induced flow velocities during times of RVF or where complex bed topographies are encountered (Rogers et al. 2003; Tseng 1999). The equations governing the conservation of mass and momentum in two dimensions can be written in the general deviatoric form (Liang et al. 2006):

$$\frac{\partial X}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = E_x + E_y \qquad (1)$$

where:

$$\mathbf{X} = \begin{bmatrix} \eta \\ q_x \\ q_y \end{bmatrix} \qquad (2)$$

$$\mathbf{F} = \begin{bmatrix} q_x \\ \frac{q_x^2}{h+\eta} + \frac{g\eta^2}{2} + gh\eta \\ \frac{q_x q_y}{h+\eta} \end{bmatrix} \qquad \mathbf{G} = \begin{bmatrix} q_y \\ \frac{q_x q_y}{h+\eta} \\ \frac{q_y^2}{h+\eta} + \frac{g\eta^2}{2} + gh\eta \end{bmatrix} \qquad (3)$$

$$\mathbf{E_x} = \begin{bmatrix} 0 \\ g\eta\frac{\partial h}{\partial x} - \frac{g^2 q_x (q_x^2+q_y^2)^{1/2}}{(h+\eta)^{(2)}} \\ 0 \end{bmatrix} \qquad \mathbf{E_y} = \begin{bmatrix} 0 \\ 0 \\ g\eta\frac{\partial h}{\partial y} - \frac{gn^2 q_y (q_x^2+q_y^2)^{1/2}}{(h+\eta)^2} \end{bmatrix} \qquad (4)$$

t is time, η is the deviation of the water surface elevation with respect to the still-water level, q_x and q_y are the discharge per unit width, h is the depth below the still-water datum, g is the gravitational acceleration, and n is Manning's n.

2.1 MacCormack Method

The explicit, second-order, finite-difference MacCormack method was adapted for use within the current scheme. With addition of a Total Variation Diminishing (TVD) modification, this scheme can accurately and robustly predict transcritical flow without the use of artificial viscosity, typically necessary when dealing with flooding and levee failure (Liang et al. 2006; Liang et al. 2007; Fennema and Chaudhry 1989). Operator-splitting, used in the MacCormack method, divides the two-dimensional SWEs to sets of one-dimensional equations for each time step. Equation (1) can be decomposed into two one-dimensional equations:

$$\frac{\partial X}{\partial t} + \frac{\partial F}{\partial x} = E_x \tag{5}$$

$$\frac{\partial X}{\partial t} + \frac{\partial G}{\partial y} = E_y \tag{6}$$

The following is the forward step of the x-direction MacCormack method.

The x-direction predictor sequence:

$$\eta^p_{i,j} = \eta^o_{i,j} - \frac{\Delta t}{\Delta x}(q^o_{x_{i,j}} - q^o_{x_{i-1,j}}) \tag{7}$$

$$\begin{aligned} q^p_{x_{i,j}} &= q^o_{i,j} - \frac{\Delta t}{\Delta x}(F^o_{i,j} - F^o_{i-1,j}) \\ &+ g\Delta t \left(\frac{\eta^o_{i,j} - \eta^o_{i-1,j}}{2}\right)\left[-\left(\frac{z_{f_{i,j}} - z_{f_{i-1,j}}}{\Delta x}\right) - E^o_{x_{i,j}}\right] \\ &+ \epsilon \frac{\Delta t}{\Delta x^2}(q^o_{x_{i-1,j}} - 2q^o_{x_{i,j}} + q^o_{x_{i+1,j}}) \end{aligned} \tag{8}$$

$$q^p_{y_{i,j}} = q^o_{y_{i,j}} - \frac{\Delta t}{\Delta x}(G^o_{i,j} - G^o_{i-1,j}) + \epsilon \frac{\Delta t}{\Delta x^2}(q^o_{y_{i-1,j}} - 2q^o_{y_{i,j}} + q^o_{y_{i+1,j}}) \tag{9}$$

The x-direction corrector sequence:

$$\eta^c_{i,j} = \frac{1}{2}\left(\eta^o_{i,j} + \eta^p_{i,j} - \frac{\Delta t}{\Delta x}(q^p_{x_{i+1,j}} - q^p_{x_{i,j}})\right) \tag{10}$$

$$\begin{aligned} q^c_{x_{i,j}} &= \frac{1}{2}\Bigg(q^o_{x_{i,j}} + q^p_{x_{i,j}} - \frac{\Delta t}{\Delta x}(F^p_{i+1,j} - F^p_{i,j}) \\ &+ g\Delta t \left(\frac{\eta^p_{i+1,j} - \eta^p_{i,j}}{2}\right)\left[-\left(\frac{z_{f_{i+1,j}} - z_{f_{i,j}}}{\Delta x}\right) - E^p_{x_{i,j}}\right] \\ &+ \epsilon \frac{\Delta t}{\Delta x^2}(U^p_{i+1,j} - 2U^p_{i,j} + U^p_{i-1,j})\Bigg) \end{aligned} \tag{11}$$

$$q^c_{y_{i,j}} = \frac{1}{2}\left(q^o_{y_{i,j}} + q^p_{y_{i,j}} - \frac{\Delta t}{\Delta x}(G^p_{i+1,j} - G^p_{i,j}) + \epsilon \frac{\Delta t}{\Delta x^2}(q^p_{y_{i+1,j}} - 2q^p_{y_{i,j}} + q^p_{y_{i-1,j}})\right) \tag{12}$$

where z is the bottom of the domain and ϵ is the desingularization coefficient, and typically remains zero, unless otherwise specified.

2.2 Boundary Conditions

Specification of the boundary conditions follows the recommendations of Fennema and Chaudhry (Fennema and Chaudhry 1989). Fictitious grid points in the solid wall mirror the values of points immediately interior to the boundary. By changing the sign of the normal component of velocity within the fictitious point, reflective boundary conditions may be created. Full-slip boundary conditions are imposed at solid walls. For subcritical flow, two inflow and one outflow boundary condition must be maintained to satisfy the 2D SWEs. For supercritical flow three influent boundary conditions must be provided for 2D flow (Garcia and Kahawita 1986; Chaudhry 2007).

2.3 Adaptive Time Step

To minimize computational run times while satisfying the Courant-Friedrich-Lewy stability condition, the time-step Δt, is adjusted by examining the velocities and water depths within the domain and re-evaluating Δt once per x and y direction solution pass:

$$\Delta t_{new} = \frac{\Delta x \cdot C_{max}}{\max\left(\sqrt{q_x^2 + q_y^2}/h + \eta\sqrt{gh + \eta}\right)_{i,j}} \tag{13}$$

Where C_{max} is the predetermined maximum Courant number, and the i, j indicator in the denominator signifies all values for an iteration that arise from a single cell. As flow within the domain trends towards critical flow conditions, time step size will decrease to maintain the assigned Courant number, and as flow trends towards steady-state, time step size will be maximized to speed computation. Implementation of this portion of the code is serial in nature, and was therefore conducted through the CPU. An alternative method would be to set the timestep and not allow for variation. Testing has shown that in situations where shocks or other discontinuities are transient, the benefit of the variable timestep outweighs the incurred memory handoff.

2.4 Total Variation Diminishing Method

The adopted TVD method derives from the proposals of (Davis 1984) and (Louaked and Hanich 1998). It is implemented in the corrector step of the MacCormack method and takes the form:

$$TVD_i = [G(r_i^+) + G(r_{i+1}^-)] \cdot \Delta X_{i+1/2}^n - [G(r_{i-1}^+) + G(r_i^-)] \cdot \Delta X_{i-1/2}^n \tag{14}$$

where:

$$\Delta X_{i+1/2}^n = X_{i+1}^n - X_i^n \tag{15}$$

$$\Delta X_{i-1/2}^n = X_i^n - X_{i-1}^n \tag{16}$$

$$r_i^+ = \frac{\Delta X_{i-1/2}^n \cdot \Delta X_{i+1/2}^n}{\Delta X_{i+1/2}^n \cdot \Delta X_{i+1/2}^n} \tag{17}$$

$$r_i^- = \frac{\Delta X_{i-1/2}^n \cdot \Delta X_{i+1/2}^n}{\Delta X_{i-1/2}^n \cdot \Delta X_{i-1/2}^n} \tag{18}$$

The X in (14–18) refers to terms in (2) while (17) amd (18) represent the dot product between the two vectors. G(x), $\phi(x)$ - the flux limiter, C, and C_l the local Courant number are given as:

$$G(x) = 0.5 \times C \times [1 - \phi(x)] \tag{19}$$

$$\phi(x) = \max(0, \min(2x, 1)) \tag{20}$$

$$\mathbf{C} = \begin{cases} C_l \times (1 - C_l), & C_l \leq 0.5 \\ 0.25, & C_l > 0.5 \end{cases} \tag{21}$$

$$C_l = \frac{(|q_x/H| + \sqrt{gH})\Delta t}{\Delta x} \tag{22}$$

This implementation of the TVD method requires no solution of eigenvalues or eigenvectors, thus reducing the number of computational steps needed per time step making it more efficient than other TVD schemes (Vincent et al. 2000; Benkhaldoun and Quivy 2006).

2.5 Wetting and Drying

For use in flood plains before, during, and after flood events, an algorithm needed to be developed to handle wetting and drying of cells without being adversely effected by large roughness values typically associated with vegetative growth. A modified version of the wetting/drying algorithm of (Liang et al. 2006) was employed. Two examinations of cell condition per timestep are performed. Each grid cell is examined initially for water surface elevation below a prescribed depth H_{dry}. If the surface is below the cutoff depth, the cell is deemed dry, and velocities in both the u and v directions are set to zero. The second step involves examination of all cells surrounding the dry cells from the first step. If any of the surrounding cells have a water surface elevation above $2H_{dry}$, then the dry cell depth will be increased by H_{dry} and the cell water depth from the donating cell will be reduced by H_{dry} for overall mass conservation. For each operator-split step (either X- or Y-direction) TVD fluxes and velocities in the direction of computation towards a dry cell are set to zero, and velocities tangent to the direction of computation as well as water surface height calculations are allowed to proceed as normal. Although this method prevents zero depths from occurring in the denominator of terms, the bed friction term can still dominate at very small water depths, causing instabilities. In order to alleviate this problem, an implicit step is used as an approximation when the water depth falls below a user-defined level (Liang et al. 2006).

In areas of very small depth, the momentum equation can be reduced to just the local acceleration and bed friction terms, which can be arranged to form the semi implicit equation:

$$\frac{q_x^{n+1} - q_x^n}{\Delta t} = -\left(\frac{gn^2\sqrt{q_x^2 + q_y^2}}{(h + \eta)^2}\right) \cdot q_x^{n+1} \tag{23}$$

It should be noted that this wetting/drying method has two distinct drawbacks. First, since velocity of the cell adjacent to the wet/dry interfaces is dictated and not calculated, overall conservation of momentum is affected. The effects of this velocity skewing are apparent in the analytical test case, presented in Section 5.1.1. Repercussions of this method in large scale physical test case scenarios are negligible, as seen in (Liang et al. 2006). Secondly, the use of any semi-implicit method within the scope of a parallelized GPU codes is non-ideal since information from two time steps is required in order to evaluate the equation. The code that has been written to deal with wetting, drying, and the relevant depth and velocity equations has a CPU based and serial implementation. For large domains where wetting and drying is occurring throughout the domain at all time steps this serial implementation will have a finite impact on the overall runtime of the model. Given the robustness and simplicity of the method, this tradeoff is acceptable.

Proper choice of H_{dry} values is critical for numerical accuracy and stability. At a minimum, the value should be greater than the maximum Manning's n for the domain. Ideally, H_{dry} should be chosen to be slightly larger than the bottom elevation difference between any two adjacent cells in a large and 'flat' section of the domain in order to not artificially impede flow across these surfaces. If H_{dry} is chosen carefully the semi-implicit calculations for shallow flow can be omitted with minimal accuracy penalty. However, omission of the semi implicit method has a measurable impact on code runtime, as discussed in Section 5.2. H_{dry} values for real-world testing within this paper were set to 5cm or less.

Additional steps were implemented to deal with wetting and drying within the GPU code as described in Section 3

3 Implementation

Two implementations of this methodology have been constructed for the purpose of comparison. The first was written in Fortran 95 and runs only on CPUs. The second is a combination of C and C++, and includes a set of CUDA kernels that solve the main numerical scheme on the GPU. Following the work of Seitz (Seitz et al. 2013), the GPU implementation of the code passes information between the GPU and CPU at every time step. These data transfers allow for the wetting/drying algorithm and the adaptive time step to be calculated once per time step on the CPU, and therefore avoid the runtime penalty incurred by running serial processes on the GPU (Seitz et al. 2013). TVD terms must be calculated on the GPU once per time step before depth and velocity calculations can occur. The predictor/corrector formulation used within requires 16 sub-steps per timestep, of which each are dependent on the previous for proper formulation. Values for each independent cell within each of the 16 sub-steps can be calculated simultaneously and in an arbitrary order without effecting the deterministic quality of the code. Variable values for these calculations are stored in shared memory, which has a latency that is generally on the order of 100x lower than that of global memory. For further information refer to Seitz.

Since information must be passed between the CPU and GPU based codes at every timestep, special attention was given to reducing the memory overhead. Defining wet and dry cells within the CPU based code at every timestep allows for variable information of only wet cells to be passed to the GPU. If a cell is dry, only the binary value representing 'dry' must be passed to the GPU shared memory for the kernel calculations. In order to better address complex boundaries that may arise in real world models, the current code utilizes array based indirect grid addressing, a graphical example of which is shown in Fig. 1. Indirect addressing for complex boundaries can be advantageous if all cells do not have the same number of neighbors (Jankowski 2009), or if in the future the code was extended to operate on both one- and two-dimensional domains simultaneously.

Domain decomposition must occur before our solution kernels are launched on the GPU. Breaking down the domain into 16 cell by 16 cell 'blocks' allows for solution of large portions of the domain simultaneously. Block size is calculated based on the configuration of the hardware on which the code is being run. For a complete discussion of hardware

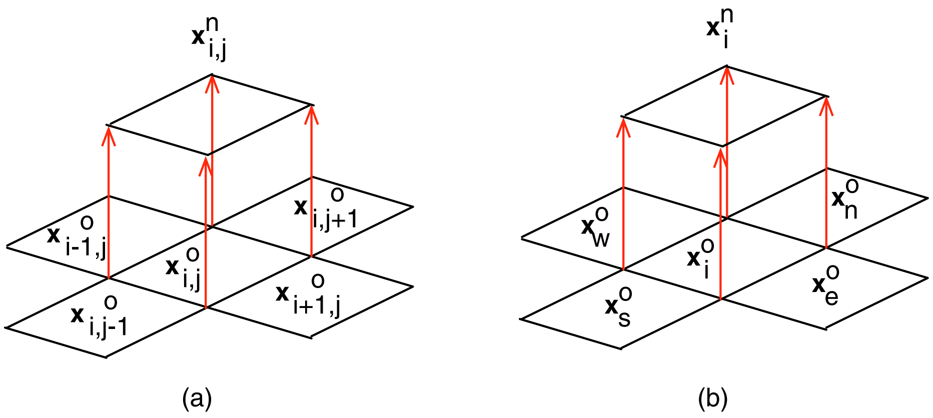

Fig. 1 (**a**) stencil-matrix form - (**b**) stencil-array form

utilization refer to the CUDA Toolkit Documentation. Methodology to allow individual blocks to calculate quickly and independent of neighboring blocks relies of the methods described by Seitz (Seitz et al. 2013) and Brodtkorb (Brodtkorb et al. 2012). To further optimize runtime, blocks containing dry cells only were not computed, and cells that were dry and more than two cells away from a wet cell carried no variable information into the shared GPU memory, reducing the memory overhead of each timestep. However, the cell locations making up each block were passed to the GPU at each time step for determination of state and block exclusion computations (Brodtkorb et al. 2012).

The Fortran 95 code was compiled using GFortran - a GNU compiler. The C/C++ CUDA code was built using the CUDA toolkit on Mac OSX 10.9.5. All codes were run on a Mid 2014 15" Retina Mac Book Pro with a NVIDIA GT 750M discrete graphics card.

The United States Bureau of Reclamation's Sedimentation and River Hydraulics (SRH-2D) solves the SWE using finite volumes, and can handle steady, unsteady, and transcritical flows. It can handle wetting and drying of cells and has proven to be a robust and accurate modeling tool. SRH-2D is based on an unstructured mesh utilizing both triangular and quadrilateral elements, and therefore a direct cell count comparison is not available. The domain and all other inputs were the same in all calculations.

4 Feather River Model Domain Significance and Surface Creation

4.1 Hydraulic and Hydrologic Significance of the Lower Feather River

The main stem of the Feather River is a tributary to the Sacramento River in the Central Valley of California. It is a drainage basin for approximately 16,000 km^2, collecting runoff from major portions of the Sierra Nevada mountain range, the Sacramento Valley, and a small portion of the southern Cascade Range. Four tributaries to the Feather River combine directly above Lake Oroville, a man-made reservoir and hydroelectric dam. The river is a major contributor to the California State Water Project, which provides water to Central and Southern California.

4.1.1 Flooding

In 1967 construction of the 230 m (770 foot) high Oroville Dam was completed, submerging the confluence of the four major tributaries and forming Lake Oroville. There have been three major floods in the last century which have resulted in levee failures on the Feather River, occurring in 1955, 1986, and 1997. The floods of 1986 and 1997 occurred after completion of the Oroville Dam. Peak discharges and river stages as measured by the California Department of Water Resources (CA-DWR) at gage designation 'GRL' (39.366577°N, -121.647369°W) measured 150,000 CFS and 100.06 feet for the 1986 flood, and 163,000 and 98.83 feet for the 1997 flood event.

4.1.2 100 Year Rainfall Event

The influent contribution to Lake Oroville during the 1997 rainfall event is considered to be the runoff from the 100 year storm. The Oroville Dam is designed to limit the effects of the 200 year storm to 170,000 cfs. The only gaged location within the study domain is gage 'GRL'. Additional information, provided as impacts for given flood stages are provided by CA-DWR (site: http://www.cnrfc.noaa.gov/graphicalRVF.php?id=gric1 Supplementary anecdotal peak flood depth measurements from the 1997 flood can be found throughout the study area. Given the empirical nature of these measurements they were only used for calibration and verification purposes.

4.2 Domain Surface Creation

The surface for this model comes from three major data sources, 2010 LiDAR data provided by Butte County, California, initially collected as part of the CVFED project, transects of the main river channel within the model domain, and UAV aerially collected data. The three sources were combined in AutoCAD Civil 3D and ArcGIS to form the final surface for modeling. Certain features, such as the tops of levees and road cuts were manually inserted into the surface in order to have a continuous elevation.

4.2.1 Updates to the LiDAR Surface

Surface creation for the model presented in this paper was conducted in the Fall of 2014, over four years after the LiDAR data was gathered. For a model to be useful for prediction of depth and velocity in specific areas an accurate and up to date surface must be used. Therefore, channel bathymetry, which may have been effected by high flow events of 2011 and subsequent drought years in 2013 and 2014 was verified and updated by taking transects in the model domain every 1,000 feet. These transects were overlaid into the LiDAR surface and interpolated to create an up to date river channel bathymetry.

Transects were measured using both a Trimble R10 RTK-GPS and a Teledyne Odom Hydrographic Echotrac CVM. Transect accuracy meets or exceeds that of LiDAR mapping. Sites of interest for PV installation had updated digital terrain models (DTM) created through the use of UAV and digital photogrammetry. Multiple 20 acre potential PV sites were identified. For the purpose of this paper, we will focus on a single site - located near the Gridley Waste Water Treatment Plant, on East Gridley Road.

In order to ortho-rectify the DTM and aerial photo, 16 ground control points (GCP) were laid out on approximate equal spacing across the site. The GCP were tied into the proper coordinate system along X,Y, and Z axis by referencing local known survey benchmarks.

Several flights of the area were made utilizing purpose built UAV and a modified camera. Flight programs and patterns were pre-programmed and controlled using a combination of GPS and magnetic compass (Westoby, (Westoby et al. 2012)). The offset and overlapping UAV obtained digital images were merged into one large coverage of the site. A technique called 'Structure-from-Motion' (SfM) was used to obtain a densely populated three-dimensional point cloud of the site. Eight of the 16 GCP were used to align the resultant aerial photograph and point cloud. The remaining eight GCP were used to verify the x,y, and z accuracies. The average absolute accuracy of the point cloud was 1.65 cm, 1.68 cm, and 2.24 cm for the x, y, and z directions, respectively. Further details may be found in Westoby, (Westoby et al. 2012) and Tighe, (Tighe et al. 2014).

4.3 Digital Terrain Model Creation

The combined surface was imported into AutoCAD Civil 3D and break lines along prominent features were created. This surface was then imported into ArcGIS, where it was trimmed to the final domain size and resampled to have 20 meter by 20 meter cell dimensions, resulting in 743,000 total domain cells. The final DTM and domain is shown in Fig. 2. The same surface was used in SRH-2D. SRH-2D employs an unstructured mesh, and although max cell edge lengths can be dictated, an exact number of cells is not reported as part of the output.

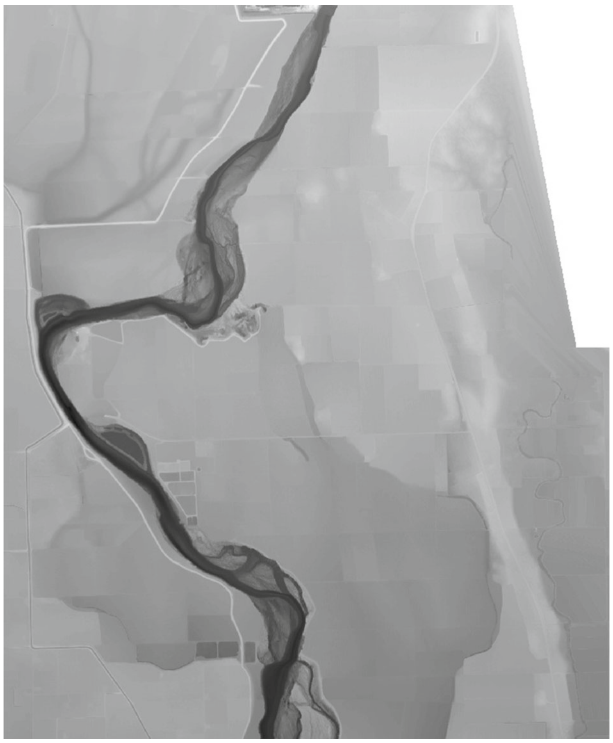

Fig. 2 Model Domain and Final Surface

Table 1 Manning's n values for different land uses

Manning's n	Land Type
0.015	Road Surface
0.020	River channel /disced and rolled land / flat dirt land
0.025	Disced land, grass land
0.030	Reeded areas, dense scrub land
0.040	Young orchard
0.045	Mature orchard

For the initial conditions of the Feather River, the wet/dry designator and initial water depth were assigned to cells in the river area whose land surface elevations were below a calculated depth-in-channel, based on gage elevations and bed slopes. $Mannings\_n$ was assigned based on land use classification and observed conditions. Table 1 lists the Manning's n values used for land use categories in the model.

5 Results

5.1 Analytical Tests for the Presented Method

5.1.1 Oscillating Water in a Parabolic Basin

In order to assess the accuracy of the proposed method, a two-dimensional free surface flow problem with known analytical solutions was modeled. The chosen problem follows the case initially presented by Thacker (Thacker 1981). This test case has been widely used for analysis of two dimensional shallow water numerical solutions, for example in: (Liang and Marche 2009; Sampson et al. 2006). The setup for this test case is exactly the same as what is found in Holdahl, (Holdahl et al. 1999).

The basin geometry for the given test case is parabolic, and given by the equation:

$$B(x, y) = D_0 \left(\frac{x^2 + y^2}{L^2 - 1} \right) \tag{24}$$

Water surface elevation (h), and u and v direction velocities are given as:

$$h = 2AD_0 \left(\frac{x \cos \omega t \pm y \sin \omega t + LB_0}{L^2} \right) \tag{25}$$

$$u = -A\omega \sin \omega t \tag{26}$$

$$v = \pm A\omega \cos \omega t \tag{27}$$

Where $\omega = \sqrt{2D_0/L^2}$, $D_0 = 1$, $L = 2500$, $A = L/2$, and $B_0 = -A/2L$. Manning's coefficient is set to zero for this test, ϵ is set to 0.01. Δx and Δy are each set to $100m$. The Courant number is set at 0.01.

All geometries were created for a 101x101 cell area on the domain [-50:50] in both the x and y directions. The results plotted in Fig. 3 are taken from the 1D cut located at $y = 0$. There is good agreement between the calculated and analytical solution for water surface elevation except near the dry/wet interface, as seen in the top row of Fig. 3. However, u direction velocities show disagreement at the drying/wetting interface. This is a common

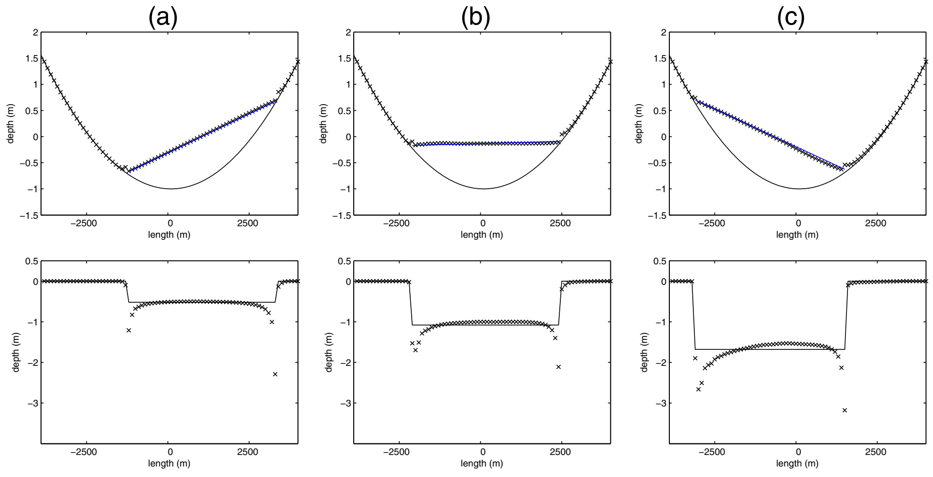

Fig. 3 (**A**) 1300 seconds (**B**) 2700 seconds (**C**) 4200 seconds

error, and one that is difficult to avoid for any shallow water scheme that includes wetting and drying (Holdahl et al. 1999). Results of this test are encouraging and show that the numerical method will adequately perform under conditions of wetting and drying within a basin. Absolute errors for the three tests of both h and u are given in Table 2.

5.1.2 Five Drops

An easily scalable and computationally intensive test was chosen to evaluate the speed increases between CPU and GPU code evaluation for the presented method. The chosen domain consisted of 1,002,001 computational cells arranged in a 1001x1001 cells with Δx and Δy of 1.0 meters. The initial condition was five columns of water 10m high with a 5m radius centered on ([501,501] [251,251] [751,251] [251,751] [751,751]). At time $t = 0$ the columns were released creating five circular shocks within the 5m deep domain. This test is considered an extreme example of shocks and transcritical flow within a domain. Frames from 5.0, 18.0, and 500 seconds can be seen in Fig. 4 below.

Manning's n was set to 0.025, ϵ was set to 0, and timesteps were adapted to a Courant number of 0.35.

Three different time variants were run in order to calculate an average speed increase for a fully-wet domain with trans critical flow. Runtimes of 500, 1000, and 2000 seconds were run for the 1001x1001 cell domain. Speed increases can be observed in Table 4.

Table 2 Absolute error for parabolic test case

time	h error (absolute)	u error (absolute)
1300s	0.0541	0.2162
2700s	0.0012	0.1225
4200s	0.0936	0.1647

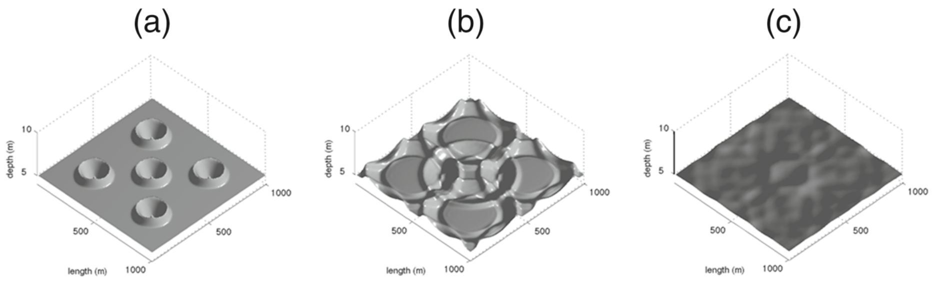

Fig. 4 (**a**) 5.0 seconds, (**b**) 18.0 seconds, (**c**) 500.0 seconds

5.1.3 Application to Feather River Surface

After determining the accuracy of the model it was calibrated to the Feather River surface. Three domain wide input layers were input, $Bottom\_Elevation$, $Surface\_Water\_Elevation$, and $Mannings\_n$, and two upstream boundary conditions and two downstream boundary condition were dictated. Surface Water Elevation is dictated as zero in initially dry cells. The upstream boundary conditions were depth and velocity, and were based on extrapolated depths and volumetric flow rates of the hydrograph from NOAA, and found at: http://www.cnrfc.noaa.gov/images/storm_summaries/jan1997/hydrographs/ordc1_inflow_outflow.gif .

Any cells below the calculated river depth at the upper boundary are deemed wet, and given a velocity based on the ratio of the volumetric flow rate and the wet cross-sectional area on the upstream boundary. The downstream boundary was set to open, achieved by assigning the same flow velocities and η's to the cells immediately to the north. The western and eastern boundaries are set to be closed and full slip, meaning the tangent velocity component is set to zero in the boundary, while the parallel component is set to that of the adjacent cell. The only calibration parameters used was adjustment Manning's n values, and inserting break lines into the original surface.

The CFL number used was 0.035, which was necessary to maintain stability without using any type of relaxation in the method or artificially smoothing the domain surface.

A table comparing gage and anecdotal depths to calculated values is given as Table 3. This table also includes the results of the SRH-2D run, demonstrating that the water surface elevation absolute error calculated by SRH-2D and the presented method generally differs

Table 3 Absolute error measured against max recorded flood depth, January 2, 1997

Station	Measured 1997 Flood Elevation (feet)	h absolute error (CPU/GPU)[1] (feet)	h absolute error (SRH-2D) (feet)
GRL	98.83	0.12	0.13
WWTP	98.6	0.2	0.1
KS1052	100.3	0.3	0.2
Robinson's Corner	100.4	0.1	0.0

[1]Results from CPU and GPU code were typically within one hundredth of a foot due to round-off errors being handled differently between the codes CPU results reported here

by less than 0.1 foot. The only gage measured data point in this table is from Station GRL, however, comparisons at the other measured locations are relevant when the present model and SRH-2D are compared against each other.

The resulting figures showing flood depth at time t = 4 Days and directly after the peak flow event are shown in Fig. 5. The flow outside the levee top (visible as a thick white line) in Fig. 5a is due to the algorithm used to determine inlet depth creating a flood wave elevation greater than that of the levee elevation. The pooled water eventually dissipates.

5.2 Runtime Comparison

Both code variants, CPU and GPU, were compiled and run for all tests within this paper. Table 4 lists all run variants and their respective runtimes.

The wetting/drying parabolic test case of Section 5.1.1 had extremely short computation times for both CPU and GPU implementations. The average overall speed increase for this test case for the three runtimes was only 1.55x. This is due in large part to the effects of the read time and preprocessing time of the two codes. Although provided, due to the overall speed of computation, these numbers should not be used for determining GPU processing effectiveness.

The Five Drops test was the most rigorously tested in terms of CPU/GPU comparison. The test was easily scaled with meaningful results for both test duration and domain size. The average speed increase, measured as a ratio of CPU runtime to GPU runtime was 62.86x for the 201 x 201 domain test, and 38.19x for the larger domains tested. It is hypothesized that the difference between the two codes has to do with the amount of information that enters and exits the GPU shared memory at every time step. As run time increases, the speed

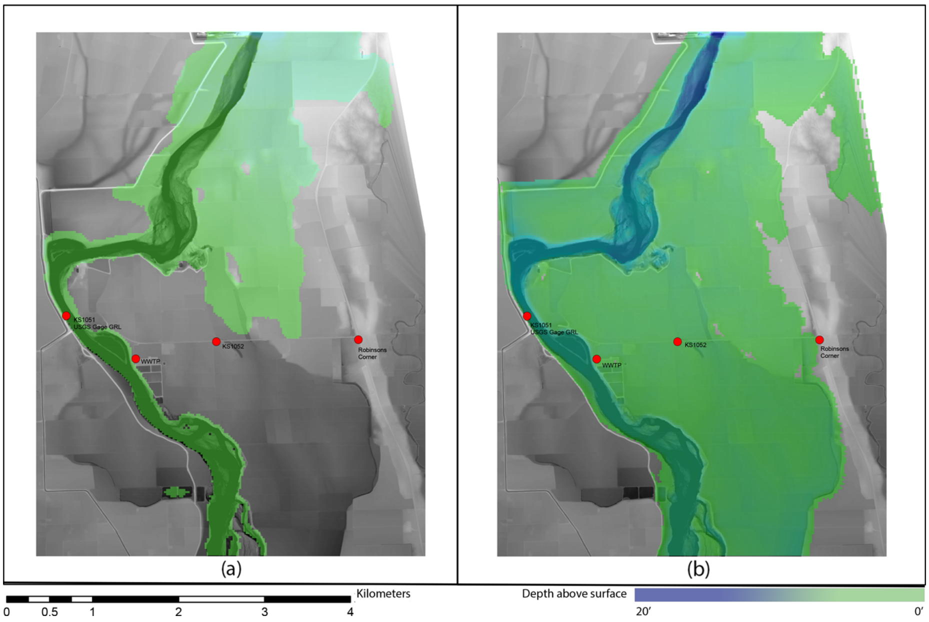

Fig. 5 (**a**) Flood extent and depth, t = 4 days. (**b**) Flood extent and depth, 2 hours after peak flood depth (January 2, 1997, 06:00) Gage GRL, and supplementary depth locations: WWTP, KS1052, and Robinsion's Corner

Table 4 Code and test varient runtimes

Test	Domain size (cell x cell)	Time (seconds)	Fortran 95 runtime (seconds)	CUDA runtime (seconds)	Speed increase (ratio F95:CUDA)	SRH-2D runtime (seconds)
Parabolic	101 x 101	1,300	2.63	1.84	1.43	NA
		2,700	5.71	3.24	1.76	NA
		4,200	9.01	5.08	1.45	NA
Five Drops	1001 x 1001	500	3,101.88	85.46	36.30	NA
		1,000	6,194.21	161.11	38.45	NA
		2,000	12,362.22	315.11	39.23	NA
Feather River	1000 x 743	172,800	59,453.96	12,234.64	4.86	13,560.0

(NA) Not applicable, code type was not run for this testcase

at which the GPU solution kernels execute outweigh the memory read/write penalty, and the ratio increases. The theoretical speed increase for fully-wet and large domains ($>$1,000,000 cells) seems to be on the order of 40x for this code comparison.

The complexity of the Feather River test case involves wetting and drying of large portions of the domain at any given time step. As the domain is initially dry, the number of blocks that contain executed kernels begins near zero. Due to the shape of this bypass relative to the flood event modeled within, a large portion of the domain remains dry for the duration of the run. These dry areas penalize the GPU code at a much higher rate than its CPU counterpart. The cells and corresponding blocks of the GPU code are effectively evaluated three times for every time step, twice in the serial portion of code; once to check if they should be considered dry, once to check if they will wet, and once as part of the block check portion of the GPU code. Additionally, even though the domain is largely dry, identifiers for the the cell condition (wet or dry) must be swapped from CPU to GPU memory and back at each time step. The summation of these penalties results in a markedly lower overall speed increase for this test case - 4.86x.

6 Conclusions

A robust and efficient second-order accurate explicit algorithm for solving the shallow water equations was developed. Two versions of the algorithm were compared, one based solely on the CPU for computation, and one that utilized both CPU and GPU computation. Comparison between the two versions showed that speed increases of the order of 40 fold can be achieved for idealized, rectangular, fully-wet conditions. For floods in complex geometries with constant drying, wetting, and transcritical flow, the speed increase, although still substantial, is reduced.

Comparisons of model predictions with existing measurements showed that SRH-2D performed generally better in the floodplain, but was matched in accuracy for in-bank flow. The non in-bank locations that had observed water surface elevations were in areas of typically sub-critical and non-rapidly-varying flow; i.e., areas where the accuracy of implicit methods have been proven. It would be valuable to compare these methods in a large and spatially accurate domain where depths and velocities in areas of transcritical flow are known. Also of interest would be assessment of the performance of the models developed herein when interfaced to a three dimensional hydrological model for applications over extended periods of time (Haahti et al. 2014)

Completing calculations on the GPU cores of a consumer-level graphics card yielded an overall faster runtime than using the implicit method of SRH-2D. Although more rigorous comparisons should occur between the code and implicit methods, this paper demonstrates that a high-accuracy second-order explicit in time method, its type generally historically ignored due to runtime limitations, can complete model runs at similar speeds to implicit methods if the GPU is utilized for calculation.

7 Compliance with Ethical Standards

This paper has not been previously been published nor is it currently under consideration for publication elsewhere. The authors affirm that the work performed and its reporting in this manuscript is free of any and all conflicts of interest.

References

de la Asuncion M, Castro MJ, Fernandez-Nieto E. D, Mantas JM, Acosta SO, Gonzalez-Vida JM (2013) Efficient GPU implementation of a two waves TVD-WAF method for the two-dimensional one layer shallow water system on structured meshes. Comput Fluids 80:441–452. Selected contributions of the 23rd International Conference on Parallel Fluid Dynamics ParCFD2011

Benkhaldoun F, Quivy L (2006) A non homogeneous Riemann Solver for shallow water and two phase flows. Flow Turb Combust 76:391–402

Brodtkorb AR, Martin SL, Altinakar M (2012) Efficient shallow water simulations on GPUs: Implementation, visualization, verification, and validation. Comput Fluids 55:1–12

Chaudhry MH (2007) Open-channel flow. Springer

Davis SF (1984) TVD finite difference schemes and artificial viscosity. Tech. Rep. 84-20. ICASE

Fennema RJ, Chaudhry MH (1989) Implicit methods for two-dimensional unsteady free-surface flows. J Hydraul Res 27:321–332

Garcia R, Kahawita R (1986) Numerical solution of the St. Venant equations with the MacCormack finite-difference scheme. Int J Numer Methods Fluids 6(5):259–274

Haahti K, Younis BA, Stenberg L, Koivusalo H (2014) Unsteady flow simulation and erosion assessment in a ditch network of a drained peatland forest catchment in eastern finland. Water Resour Manag 28(14):5175–5197. doi:10.1007/s11269-014-0805-x

Holdahl R, Holden H, Lie KA (1999) Unconditionally stable splitting methods for the shallow water equations. BIT Numer Math 39(3):451–472

Jankowski JA (2009) Parallel implementation of a non-hydrostatic model for free surface flows with semi-Lagrangian advection treatment. Int J Numer Methods Fluids 59(10):1157–1179

Liang D, Falconer RA, Lin B (2006) Comparison between TVD-MacCormack and ADI-type solvers of the shallow water equations. Adv Water Resour 29:1833–1845

Liang D, Lin B, Falconer RA (2007) Simulation of rapidly varying flow using an efficient TVD-MacCormack scheme. Int J Numer Methods Fluids 53:811–826

Liang Q, Marche F (2009) Numerical resolution of well-balanced shallow water equations with complex source terms. Adv Water Resour 32(6):873–884

Lin GF, Lai JS, Guo WD (2003) Finite-volume component-wise TVD schemes for 2D shallow water equations. Adv Water Resour 26:861–873

Louaked M, Hanich L (1998) TVD scheme for the shallow water equations. J Hydraul Res 36(3):363–378

Néelz S, Pender G (2013) Benchmarking the latest generation of 2d hydraulic modelling packages. Report SC12000. Environmental Agency, UK

Ransom O, Younis BA (2015) Selective application of a total variation diminishing term to an implicit method for two-dimensional flow modelling. Journal of Flood Risk Management 8(1):52–61

Rogers BD, Borthwick AGL, Taylor PH (2003) Mathematical balancing of flux gradient and source terms prior to using Roe's approximate Riemann solver. Chin J Comput Phys 192:422–451

Sampson J, Easton A, Singh M (2006) Moving boundary shallow water flow above parabolic bottom topography. ANZ J Surg 47:C373–C387

Sastra M. L, Brodtkorb A. R (2012). In: Jonasson K. (ed) Shallow water simulations on multiple GPUs, vol 7134. Springer, Berlin, pp 56–66. doi:10.1007/978-3-642-28145-7_6

Seitz KA, Kennedy A, Ransom O, Younis BA, Owens JD (2013) A GPU implementation for two-dimensional shallow water modeling. arXiv preprint. arXiv:1309.1230

Smith LS, Liang Q (2013) Towards a generalized GPU/CPU shallow-flow modeling tool. Comput Fluids 88:334–343

Thacker WC (1981) Some exact solutions to the nonlinear shallow-water wave equations. J Fluid Mech 107:499–508

Tighe J, Niethammer M, Lazebnik S (2014) Scene parsing with object instance inference using regions and per-exemplar detectors. Int J Comput Vis:1–22

Tseng MH (1999) Explicit finite volume non-oscillatory schemes for 2d transient free-surface flows. Int J Numer Methods Fluids 30:831–843

Vacondio R, Palù AD, Mignosa P (2014) Gpu-enhanced finite volume shallow water solver for fast flood simulations. Environ Model Softw 57:60–75

Vincent S, Caltagirone JP, Bonneton P (2000) Numerical modeling of bore propagation and run-up on sloping beaches using a MacCormack TVD scheme. J Hydraul Res 39(1):41–49

Westoby MJ, Brasington J, Glasser NF, Hambrey MJ, Reynolds JM (2012) Structure-from-Motion photogrammetry: A low-cost, effective tool for geoscience applications. Geomorphology 179:300–314