Fast solution of the Shallow Water Equations using GPU technology

A Crossley, R Lamb, S Waller

JBA Consulting, South Barn, Broughton Hall, Skipton, North Yorkshire, BD23 3AE. amanda.crossley@jbaconsulting.co.uk

Abstract

The increasing reliance on 2D flood modelling provides a strong incentive to reduce model run times, especially for studies involving uncertainty analysis or simulation of multiple scenarios. This paper reports a 2D Shallow Water Equation (SWE) flood model that has been developed to run on the graphics card of a desktop PC, exploiting the fast parallel floating point performance of the graphics processing unit (GPU). We give an overview of the Riemann based finite volume scheme and its implementation in serial and parallelised algorithms. We show test results to demonstrate that the scheme produces depth and velocity simulations that compare well with other models widely used in the industry for a realistic urban flood modelling application. We also show that the parallel GPU implementation of our scheme runs significantly faster than the corresponding serial code on a conventional processor.

In addition to numerical results from the new SWE model, we report on practical aspects in the use of GPU technology for flood modelling, building on earlier experience with a 2D diffusion wave model. We discuss recent improvements in hardware and software, and the cost-efficiency of the technology for flood risk modelling and research applications.

Introduction

In recent years the use of two dimensional (2D) flood inundation models has become well established in flood risk management. A range of models has emerged based on different approaches to modelling the flow of water over a variable topography. One of the most popular approaches is to apply the Shallow Water Equations (SWE), which are simplifications of the Navier-Stokes fluid flow equations for depth-averaged, incompressible flows. In practical flood modelling applications there is a need for robust, stable numerical schemes that can cope with features observed in real flood situations, such as a flood wave advancing over dry ground and weir-like flows over breaks in slope.

In addition, calculation is becoming particularly important because of the use of scenario analysis and techniques such as Monte Carlo modelling that require many different simulations to be performed. The calculations that are performed to solve a numerical scheme for the shallow water equations involve thousands or millions of iterations where the same arithmetic operations are repeated. Often, parts of the solution can be written in such a way that arithmetic operations are carried out largely independently of each other. This makes it possible to do the numerical work in parallel, using multi-core processors where arithmetic and logic instructions can be executed concurrently on each core. Current technology trends point towards greater use of parallel processing rather than unlimited growth in the speed of each individual core.

One of the emerging technologies for parallel computation is the Graphics Processor Unit (GPU). This has been based on the development of fast, highly parallel processors to support

the video games and computer graphics markets. Although GPUs began as specialised graphics accelerators, the technology has now advanced such that graphics hardware supports scientific calculation. GPU hardware is relatively cheap and can easily be deployed in typical office workstation PCs.

We have previously applied GPU programming techniques to accelerate a 2D diffusion wave flood model, JFLOW (Lamb *et al*., (2009)). The 'JFLOW-GPU' code has since been used to produce national flood mapping data sets for rivers (in the UK, France Czech Republic, Slovakia and Hungary) and also areas susceptible to surface water flooding, as well as numerous smaller-scale studies. However, a SWE-based model can provide better results in situations where momentum is important and also more realistic prediction of velocities. Building on the advances made in the first generation of GPU flood modelling, we have therefore developed a SWE model for the GPU platform. The scheme reported in this paper is a new formulation that will be made available in future releases of the JFLOW software, hence we refer to it as "JFLOW SWE".

Model formulation

In two dimensions the shallow water equations can be written as

$$\mathbf{U}_t + \mathbf{F}_x + \mathbf{G}_y = \mathbf{R}$$

where **U** is the vector of conserved variable, **F** and **G** represent the flux terms and **R** is the source term vector. The vectors can be expressed as

$$\mathbf{U} = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{pmatrix} \text{ and } \mathbf{R} = \begin{pmatrix} 0 \\ gh(S_{0_x} - S_{f_x}) \\ gh(S_{0_y} - S_{f_y}) \end{pmatrix}$$

where h is the water depth and u and v represent the velocity components in the orthogonal grid directions. The source terms are comprised of a bed slope term and a friction component (which can be expressed in terms of Manning's n) whereby

$$S_{0_x} = \frac{\partial z}{\partial x}, \quad S_{0_y} = \frac{\partial z}{\partial y}$$

and

$$S_{f_x} = \frac{n^2 u\sqrt{u^2 + v^2}}{h^{4/3}}, \quad S_{f_y} = \frac{n^2 v\sqrt{u^2 + v^2}}{h^{4/3}}.$$

In the absence of source terms, the model equations written in this form describe a series of conservation laws where water depth and momentum are conserved. The scheme we use is a finite volume implementation of Roe's scheme (Roe and Pike, 1984, Toro, 2001, Le Veque, 2002), which can be written for a 2D system of conservation laws in the form

$$\mathbf{U}_{i,j}^{n+1} = \mathbf{U}_{i,j}^{n} - \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i+1/2,j}^{*} - \mathbf{F}_{i-1/2,j}^{*}\right) - \frac{\Delta t}{\Delta y}\left(\mathbf{G}_{i,j+1/2}^{*} - \mathbf{G}_{i,j-1/2}^{*}\right) + \Delta t \mathbf{R}_{i,j}^{n}$$

where the numerical fluxes are

$$\mathbf{F}_{i+1/2,j}^{*} = \frac{1}{2}\left[\mathbf{F}\left(\mathbf{U}_{i+1,j}^{n}\right) + \mathbf{F}\left(\mathbf{U}_{i,j}^{n}\right) - \left(\sum \tilde{\alpha}_k \left|\tilde{\lambda}_k\right| \tilde{\mathbf{e}}_k\right)_{i+1/2,j}\right]$$

$$\mathbf{G}_{i,j+1/2}^{*} = \frac{1}{2}\left[\mathbf{G}\left(\mathbf{U}_{i,j+1}^{n}\right) + \mathbf{G}\left(\mathbf{U}_{i,j}^{n}\right) - \left(\sum \tilde{\alpha}_k \left|\tilde{\lambda}_k\right| \tilde{\mathbf{e}}_k\right)_{i,j+1/2}\right].$$

We make use of a regular, structured rectangular grid that allows immediate use of raster Digital Elevation Model (DEM) data sets for flood mapping. The *i* and *j* subscripts refer to the cell indexes within the Cartesian grid. The use of a tilde (~) denotes a Roe averaged quantity across the specified interface, where the averaging takes account of the local wave speed. For the *x* direction, the average values are defined as

$$\tilde{\alpha}_1 = \frac{(\tilde{c} - \tilde{u})\Delta h + \Delta(hu)}{2\tilde{c}}, \quad \tilde{\alpha}_2 = \frac{(\tilde{c} + \tilde{u})\Delta h - \Delta(hu)}{2\tilde{c}}, \quad \tilde{\alpha}_3 = \Delta(hv) - \tilde{v}\Delta h$$

$$\tilde{\lambda}_1 = \tilde{u} + \tilde{c}, \quad \tilde{\lambda}_2 = \tilde{u} - \tilde{c}, \quad \tilde{\lambda}_3 = \tilde{u}$$

$$\tilde{\mathbf{e}}_1 = \begin{pmatrix} 1 \\ \tilde{u} + \tilde{c} \\ \tilde{v} \end{pmatrix}, \quad \tilde{\mathbf{e}}_2 = \begin{pmatrix} 1 \\ \tilde{u} - \tilde{c} \\ \tilde{v} \end{pmatrix}, \quad \tilde{\mathbf{e}}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

where

$$\tilde{u} = \frac{\sqrt{h_L}u_L + \sqrt{h_R}u_R}{\sqrt{h_L} + \sqrt{h_R}}, \quad \tilde{v} = \frac{\sqrt{h_L}v_L + \sqrt{h_R}v_R}{\sqrt{h_L} + \sqrt{h_R}}, \quad \tilde{h} = \frac{1}{2}(h_L + h_R), \quad \tilde{c} = \sqrt{g\tilde{h}}$$

where the Δ operator implies the difference between the right and left states.

The scheme falls under the classification of a Godunov type approach and is based on the approximate solution of Riemann problems yielding an upwinded shock capturing method. We have chosen to implement a first order version of the scheme. It is possible to extend the method to second order through the introduction of flux/slope limiters. We have also adopted a unsplit strategy for solving the 2D equations. This means that the update is performed in a single sweep through the domain. As Roe's scheme is explicit, it is subject to a stability constraint of the form

$$a\frac{\Delta t}{\Delta x} \le 1 \qquad (1)$$

where a is a characteristic velocity and the left hand side of Equation 1 is known as the Courant-Fredrich-Levy (CFL) number (Courant *et al.* (1967)). For the 1D shallow water equations, the appropriate velocity term is $a = |u|+c$. The maximum permissible time step for each cell is calculated and then the minimum of these values over the whole grid is used for the next update. In 2D, a number of alternative formulations are possible (Brandford and Sanders, 2002, Brufau *et al.*, 2004, Delis *et al.*, 2008, Toro, 1992 and Zhou *et al.*, 2001). Here, a formulation based on the 1D strategy has been adopted whereby

$$\Delta t_i = \mathrm{Min}\left(\frac{\Delta x}{|u_{i,j}| + c_{i,j}}, \frac{\Delta y}{|v_{i,j}| + c_{i,j}} \right) \times 0.5 .$$

The ground slope and friction slope enter the scheme as the source terms **R.** Recent studies (Bermudez, A. and Vázquez (1994), Vázquez-Cendón (1999), Brufau *et al.* (2002)) have highlighted the benefits of employing upwind source term treatments whereby the left and right components are effectively weighted based upon local wave speeds. We utilise an upwind treatment for the bed slope source term, which, coupled with the Roe flux, results in a well balanced method, meaning that a physically realistic equilibrium solution can be obtained for a lake at rest whereby the water level is constant and the velocity is zero. The friction source terms are calculated in a pointwise fashion.

The application of a 2D SWE code to real world situations requires some additional considerations because the domain may contain both wet and dry regions, and cells may become flooded or may dry out during a simulation. The situations where a special treatment is required are

1. Water surface advancing over an adverse dry bed slope.
2. Water cascading from cells of high elevation to neighbouring cells whose water level is below the ground elevation.
3. Dry cells and cells drying out.

Cases 1 and 2 are closely related and are treated essentially by applying similar calculations as for a transmissive boundary condition locally within the model grid. A local wall condition is implemented to prevent flow advancing unrealistically over adverse bed slopes. For dry or drying cells, a depth threshold is used to determine whether to perform the flux calculations for a given cell interface. Flux calculations are only performed if the depth in one of the cells exceeds the specified value, which may be treated as an additional model parameter. The threshold is also used to determine when to apply the friction formulation and the velocity components are set to zero if the depth falls below the threshold.

Implementation

The scheme summarised above has been implemented in code targeted at two types of computer processor. One implementation is written in C++ for a single x86 processor core, which is the standard type of CPU in most desktop PCs running Windows. This version of the algorithm is a serial code, in other words all loops and calculations are executed one after another on the same processor core.

The second implementation runs on NVIDIA graphics cards using the Graphics Processing Unit (GPU) as a fast parallel numerical co-processor. This makes use of CUDA, a language based on C and developed by NVIDIA to enable programmers to exploit the multiple arithmetic cores and fast memory available in modern graphics cards.

Although both codes implement an identical algorithm, there are inevitably some subtle differences in the outputs because the two codes target radically different processor architectures. However, for practical purposes the results are numerically nearly identical, typically to at least five or six significant figures. This is an advance over the earlier GPU implementation of the JFLOW diffusion wave model, where the x86 and GPU codes produced results that were similar, but could be visibly distinguished on plots of results such as water level outputs.

Test cases

The software implementations of the above 2D SWE solution scheme produce comparable results to those published by other authors for standard idealised test cases such as instantaneous lateral and radial dam breaks, flows over a plane surface and steady state equilibrium solutions. Such tests are of interest in evaluating the behaviour of the numerical scheme with respect to analytical or reference results, but are not 'real world' applications. Here we present tests results for a realistic model application to a detailed urban flood benchmark case.

Urban flood benchmark case

This test is based on modelling flooding from a hypothetical culvert blockage scenario in an area near to Glasgow, making use of data used in previous 2D model comparison studies (Hunter *et al.*, (2008), Lamb *et al.*, (2009)) and, currently, in a benchmark study commissioned by the Environment Agency of England and Wales. The case study uses a 2 m x 2 m DEM grid (Figure 1) and a spatially varying Manning's n roughness grid. A hypothetical, plausible flow hydrograph is applied as an input to the model at the point marked in the upper right of the grid and the simulation allowed to proceed for two hours (model time).

Figure 2 shows maximum flood depths simulated using the SWE scheme presented in this paper. Whilst there are no observations available for this hypothetical case study, there are results from other well known 2D hydraulic models including TUFLOW, which is widely used in flood modelling studies in the UK. For comparison the results from TUFLOW are also shown (using an implicit finite difference ADI solver for the shallow water equations).

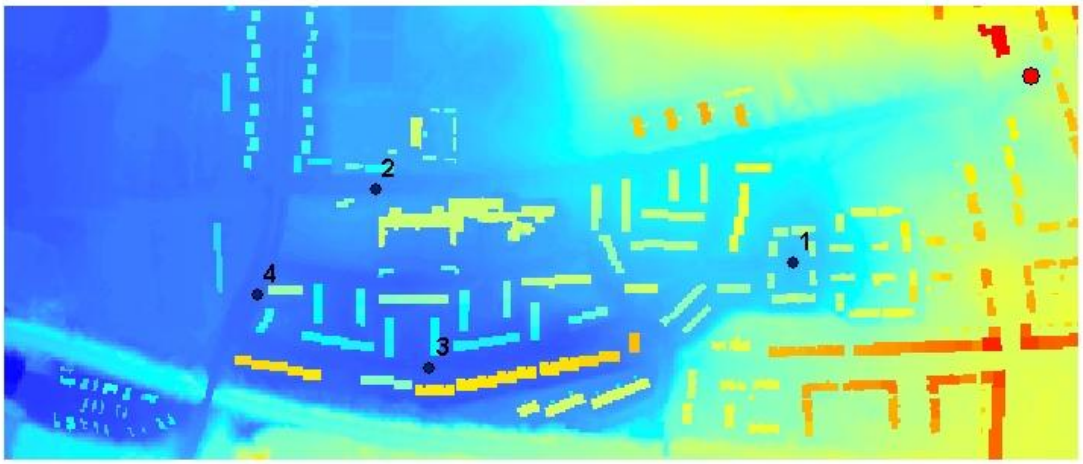

Figure 1. Glasgow urban test case DEM showing numbered monitor point locations and inflow location (red dot in upper right of the image).

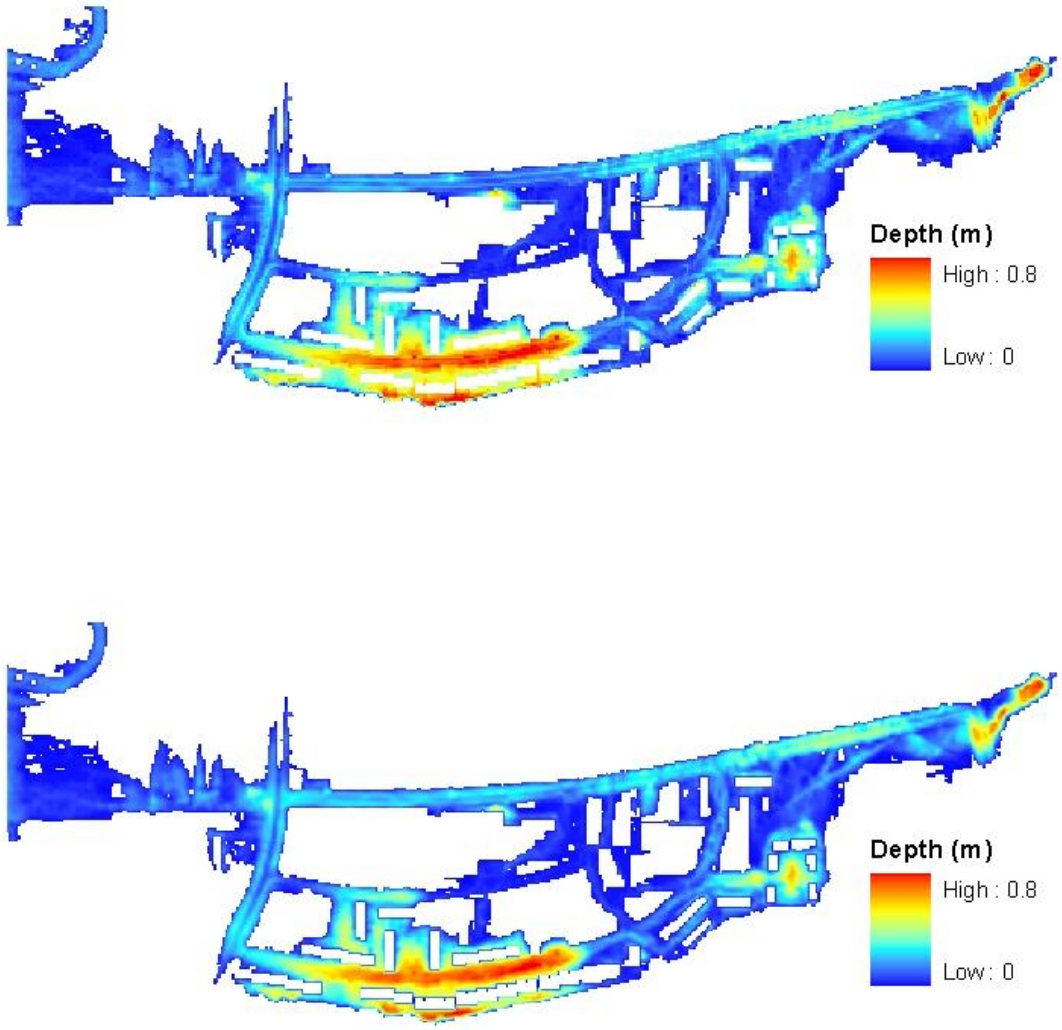

Figure 2. Maximum depth grids. Upper panel: JFLOW SWE scheme. Lower panel: TUFLOW.

Although the different numerical schemes and software implementations produce some slight differences in modelled depths, it can be seen that overall the two models generate very similar results. Similarly, the patterns of maximum velocity (Figure 3) produced by the two models are similar. There is some evidence that the TUFLOW comparison results are slightly smoother, as might be expected for a model that uses an implicit solution scheme.

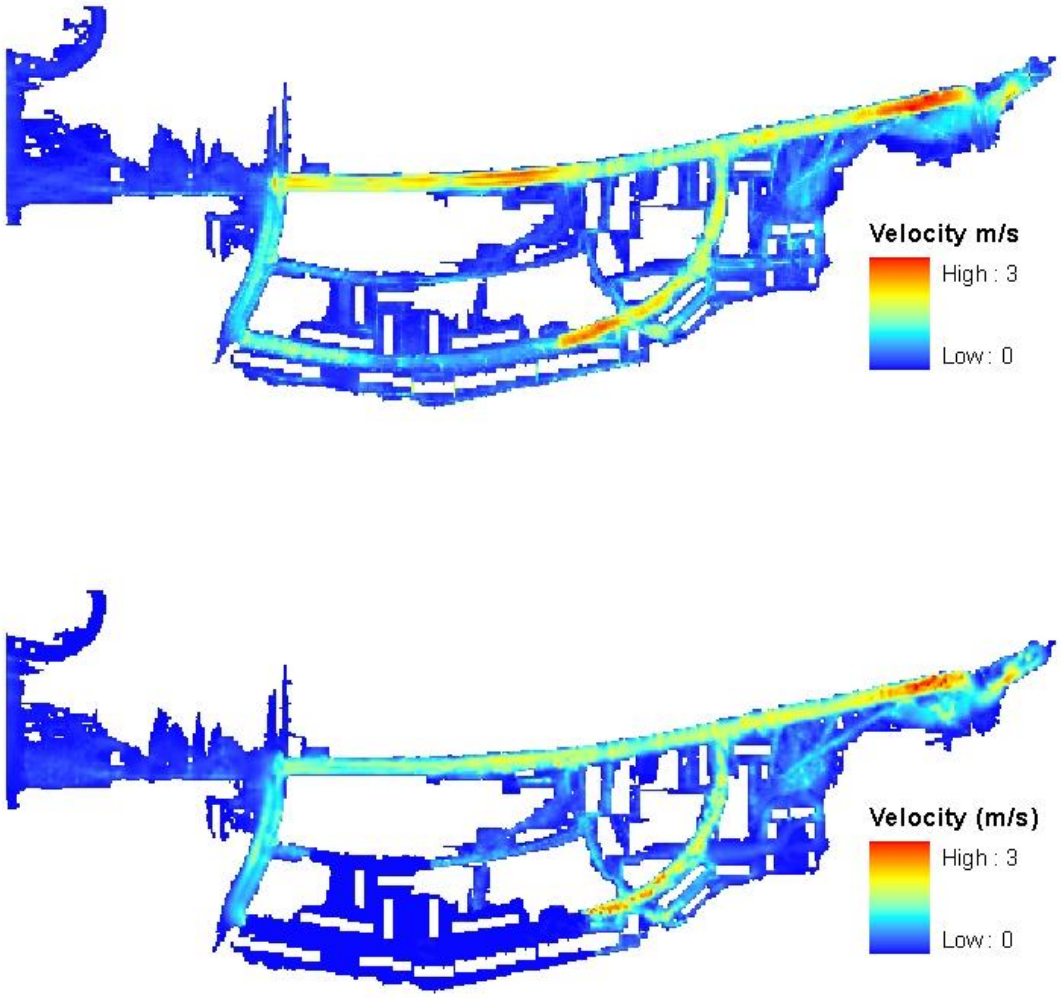

Figure 3. Maximum velocity grids. Upper panel: JFLOW SWE scheme. Lower panel: TUFLOW.

Time series plots are shown for depth (Figure 4) and velocity (Figure 5) at the monitor points labelled 1 – 4 in Figure 1. Again, the results of our scheme are compared with a TUFLOW simulation here. The data can be compared with several other 2D hydraulic models by reference to figures shown in Hunter *et al*. (2008).

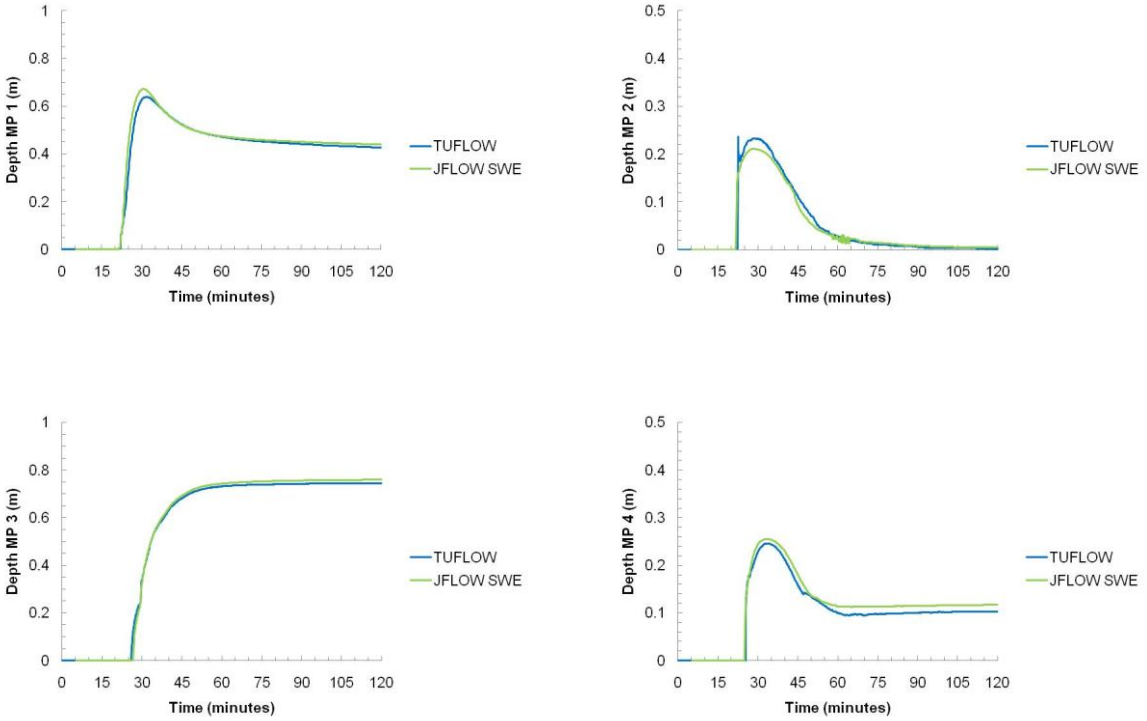

Figure 4. Depth time series at monitor points.

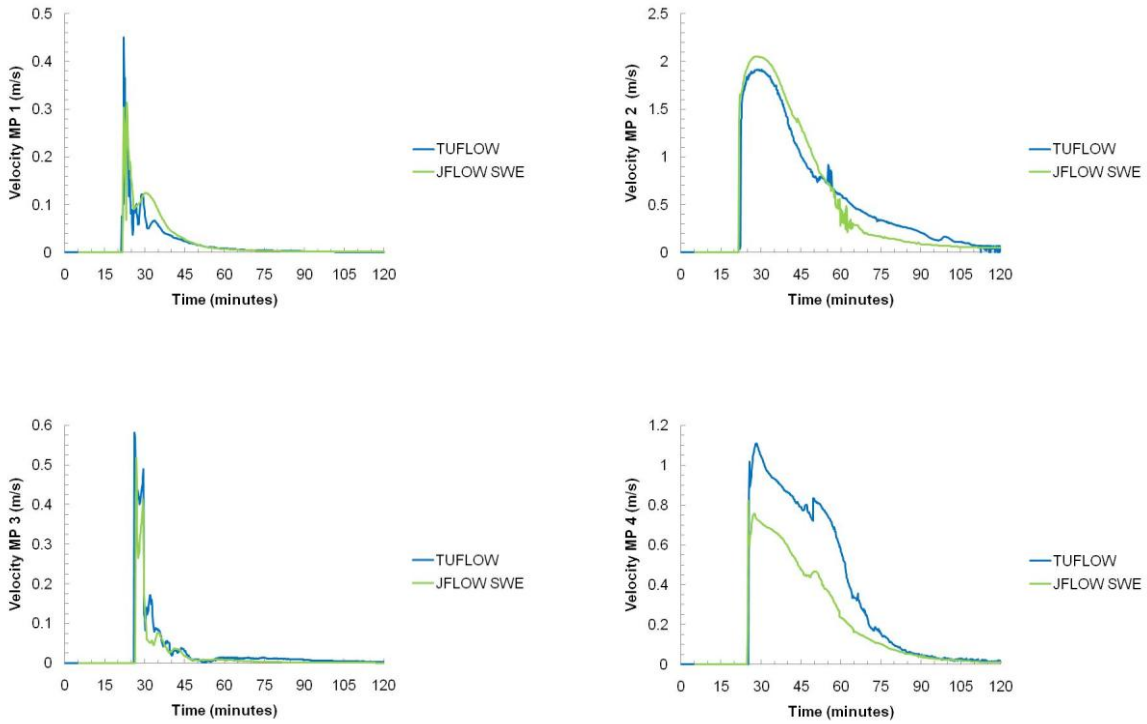

Figure 5. Velocity time series at monitor points.

Run time

The Glasgow benchmark tests were performed using an NVIDIA GeForce GTX285 graphics card. Run time for the JFLOW SWE GPU test simulation was 46 seconds. This compares with previously published results of 9.5 minutes for the previous JFLOW-GPU diffusion wave code. The host PC has an AMD Phenom II 2.8 Ghz CPU with 3GB RAM. Relative to the serial C++ (x86 CPU) code implementation of the JFLOW SWE scheme, the corresponding GPU code was 116 times faster.

Practical considerations

There is now considerable experience in the use of the GPU computing platform for flood modelling. Applications include flood mapping, as noted earlier, along with research projects where multiple Monte Carlo simulations have been carried out. An example is the area around Mexborough where work in FRMRC2 involved running 1000 simulations of the JFLOW diffusion wave model to explore uncertainty associated with specification of Manning's n for the floodplain. This type of analysis is possible on a desktop PC with GPU hardware that costs around £300.

For large scale (e.g. national) flood mapping, our experience has been that it is practical and cost effective to build a grid of GPU-based PCs. The development of appropriate software tools enables such a resource to be used either for large batch processing jobs, such as Monte Carlo analysis, or for the scheduling of individual model runs. With the advent of fast hydraulic calculations, it is often the case that computational bottlenecks appear in the databases serving information, particularly DEM data, rather than the hydraulic model itself.

Hardware reliability is very occasionally an issue when running a cluster of GPUs. Previous generations of GPU cards have lacked some of the features found in conventional (and far more expensive) supercomputers. However, as part of the trend towards increasing use of GPUs for technical computing, the latest generation of processors include 'safety' features such as error-correcting memory protection. In our practical experience running a grid of more than 50 GPUs, we have encountered very few issues with the reliability of the GPU hardware itself. Where reliability is of particular importance, graphics hardware vendors supply 'professional' versions of their hardware, albeit at prices several times more expensive than the equivalent mass market consumer products.

References

Bermudez, A. and Vázquez, M.E. 1994. Upwind methods for hyperbolic conservation laws with source terms. *Computers Fluids, 23*, 1049-1071.

Bradford, S.F. and Sanders, B.F. 2002. Finite-volume model for shallow-water flooding of arbitrary topography. *J. Hydraul. Eng.* **128**, 289-298.

Brufau, P., Vázquez-Cendón, M.E. and García-Navarro, P. 2002. A numerical model for the flooding and drying of irregular domains. *Int. J. Numer. Meth. Fluids.*, **39**, 247-275.

Brufau, P., García-Navarro, P. and Vázquez-Cendón, M.E. 2004. Zero mass error using unstrady wetting-drying conditions in shallow flows over dry irregular topography. *Int. J. Numer. Meth. Fluids., 45*, 1047-1082.

Courant, R., Friedrichs, K.O. and Lewy, H. 1967. On the partial difference equations of mathematical physics. *IBM Journal*, **11** 215-234.

Delis, A.I., Kazolea, M. and Kampanis, N.A. 2008. A robust high-resolution finite volume scheme for the simulation of long waves over complex domains. *Int. J. Numer. Meth. Fluids.*, **56**, 419-452.

Hunter, N.M. Bates, P.D. Neelz, S. Pender, G. Villanueva, I., Wright, N.G., Liang, D., Falconer, R.A., Lin, B., Waller, S., Crossley, A.J. and Mason, D.C. 2008. Benchmarking 2D hydraulic models for urban flood simulations. *Proceedings of the Institution of Civil Engineers: Water Management*, **161**, 13-30.

Lamb, R., Crossley M.and Waller, S. 2009. A fast two-dimensional floodplain inundation model. *Proceedings of the Institution of Civil Engineers: Water Management*, **162**, 363-370.

Le Veque, R. J. 2002. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press.

Roe, P.L. and Pike, J. 1984. Efficient construction and utilistaion of approximate riemann solutions. In: Glowinski, R. and Lions, J.L. (ed). *Proc. of the Sixth Int. Symposium on Computer Methods in Applied Sciences and Engineering*, 499-518.

Toro, E.F. 1992. Riemann problems and the WAF method for solving the two-dimensional shallow water equations. *Phil. Trans. R. Soc. Lond. A, 338*, 43-68.

Toro, E.F. 2001. *Shock–Capturing Methods for Free-Surface Shallow Flows*. Wiley.

Vázquez-Cendón, M.E. 1999. Improved treatment of source terms in upwind schemes for the shallow water equations in channels with irregular geometry. *J. Comput. Phys*., **148**, 497-526.

Zhou, J.G., Causon, D.M., Mingham, C.G. and Ingram D.M. 2001. The surface gradient method for the treatment of source terms in the shallow-water equations. *J. Comput. Phys*., **168**, 1-25.