Eleventh International Conference on CFD in the Minerals and Process Industries
CSIRO, Melbourne, Australia
7-9 December 2015

SWIFT: A GPU BASED COUPLED HYDRODYNAMIC/HYDRAULIC FRAMEWORK FOR URBAN FLOOD PREDICTION

Raymond C. Z. COHEN[1*], James E. HILTON[1], Shahriar HASAN KHAN[2], Yunze WANG[1] and Mahesh PRAKASH[1]

[1] CSIRO Data61, Clayton, VIC 3168, AUSTRALIA

[2] Dept. of Electrical and Computer Systems Engineering, Monash University, VIC 3800, AUSTRALIA
*Corresponding author, E-mail address: Raymond.Cohen@csiro.au

ABSTRACT

Rapid high-resolution flood predictions are necessary to provide warnings for areas at risk of coastal inundation and flash flooding. Typically, such areas are in urban environments where increased surface runoff greatly increases the effects of inundation and flooding. However, urban environments provide a difficult modelling challenge as surface flow is strongly affected by urban drainage networks. We present a GPU-based hydrodynamic model coupled to a hydraulic network model, both part of the CSIRO Swift modelling framework. The hydrodynamic flow model is based on a finite volume implementation of the shallow water equations, specifically formulated for overland flow. The hydraulic network model is based on a pressure relaxation method, and uses a GPU-based sparse matrix solver to increase the computational solution speed. To optimise the matrix solution a number of sparse matrix formats and solution techniques were examined. Bottlenecks in the solution process due to GPU architectural limitations have been identified and are discussed. The model has been validated for a number of test cases and has been used for modelling potential combined coastal and catchment inundation for the City of Port Phillip, Melbourne. Future uses of the Swift framework include inundation warning systems and flash flooding predictions using meteorological rainfall forecasts. Despite architectural limitations we have found the use of GPUs to be extremely valuable, providing an order of magnitude speed-up over multi-core CPUs in our flood modelling work.

NOMENCLATURE

h	water height / pressure head [m]
B	vertical height of the ground from a datum [m]
B_x, B_y	reconstructed base slopes
w	vertical position of the water surface ($B + h$) [m]
$\mathbf{u}$	2D water velocity (u_x,u_y) [m/s]
q_x, q_y	cell discharge rate (hu_x, hu_y) [m^2/s]
ρ	density [kg/m^3]
Q	volume flow rate in a pipe [m^3/s]
q	net volume flow into a junction [m^3/s]
L	pipe length
d	pipe diameter

INTRODUCTION

Accurate prediction of flooding of vulnerable urban areas is important for both long term activities of infrastructure planning and mitigation; and for short term responses such as evacuation of people and property. Regions at high risk of flooding include coastal areas, from storm surges, and low-lying areas, from flash flooding events. Flood prediction using computational models is usually carried out as part of risk assessments with simulations of single scenarios. However, increasing computational power, especially with the availability of new GPU based architectures, is now allowing faster than real-time flood prediction. This ability is very valuable for early warnings and operational management during flood events.

For detailed planning and risk mitigation, it is necessary to consider flooding at street level length scales. The computation time for hydrodynamic simulations scales strongly with resolution, making detailed simulations very computationally expensive. Furthermore, urban drainage networks are an important factor in flood modelling that is often overlooked in many urban studies due to the difficulty of coupling hydraulic and hydrodynamic models. Ideally drains help to reduce flood impact by efficiently transporting water away from above ground infrastructure. Unfortunately during extreme flood events the flow in drains can reverse, worsening the flooding impact. This is an especial hazard in coastal areas which may simultaneously have high tides, storm surge and heavy rainfall events. Drains are incorporated into the present model and are coupled to the hydrodynamic above-ground flow model.

Only a small number of real time urban flood studies have been published in the literature for Thailand (Mark et al., 2002) and France (Raymond et al., 2006). Henonin et al. (2013) provided a summary of the state of the art in real-time flood prediction, including assessments of coupled hydrodynamic-hydraulic drainage network models. More recently René et al. (2014) claimed that the lack of real-time urban flood forecasting is not due to lack of data availability but is due to many practitioners being unaware that that they have the means of carrying out rainfall flood forecasts. The Australian Bureau of Meteorology currently issues general flood warnings for regions and rivers across Australia. These general warnings do not contain detailed flood prediction maps, making it difficult for people to understand local risk accurately and whether action is necessary to protect lives and property. Recently a European trans-national project called “RainGain” was commenced to enable real-time urban pluvial flooding prediction in pilot cities of Rotterdam, Paris, London and Leuven (“RAINGAIN,” 2013). This work seeks to use new fine scale (temporal and spatial) rain radar information with accurate terrain information and drainage networks in pluvial flooding modelling.

This paper details the challenges of efficiently coupling hydrodynamic and hydraulic models, providing a framework for accurate prediction of urban flooding known as Swift. As much of their computation as possible is carried out on GPUs. Use of GPU hardware is found to produce a factor of approximately twenty speed up over multi-core CPU implementations. Furthermore, the OpenCL GPU implementation used is highly scalable, allowing simulations to be run on systems ranging from desktop computers to cloud-based GPU clusters. Since software portability was a critical requirement of Swift, CUDA implementations were not considered in this study as it's only available on NVidia hardware, whereas OpenCL is supported by all major software vendors for both GPUs and CPUs.

HYDRODYNAMICS AND HYDRAULICS

The most general set of governing equations for hydrodynamics are known as the 3D incompressible Navier-Stokes equations (NSE). For modelling of flooding over widespread areas it is more practical to use a simplification of these equations called the 2D shallow water equations (SWE) which assume that there is no vertical velocity and that the base friction acts through a drag coefficient (Kurganov and Petrova, 2007):

$$\frac{\partial w}{\partial t}+\frac{\partial q_x}{\partial x}+\frac{\partial q_y}{\partial y}=0 \qquad (1)$$

$$\frac{\partial q_x}{\partial t}+\frac{\partial}{\partial x}\left[\frac{q_x^2}{w-B}+\frac{g(w-B)^2}{2}\right]+\frac{\partial}{\partial y}\left[\frac{q_x q_y}{w-B}\right] = -(w-B)B_x \qquad (2)$$

$$\frac{\partial q_y}{\partial t}+\frac{\partial}{\partial x}\left[\frac{q_x q_y}{w-B}\right]+\frac{\partial}{\partial y}\left[\frac{q_y^2}{w-B}+\frac{g(w-B)^2}{2}\right] = -(w-B)B_y \qquad (3)$$

These equations represent the depth averaged height and velocity of water above a fixed datum. The drag between the bed and the fluid is included using a Manning roughness factor (Bradford and Sanders, 2002). The shallow water equations are discretised onto a Cartesian grid and implemented using a finite volume method.

The hydraulic pipe network model is developed from two fundamental equations (Boulos et al., 2006) – the pipe flow head loss equation:

$$h_i - h_j = K_{ij}Q_{ij}^n \qquad (4)$$

And the conservation of mass equation:

$$\sum_i Q_{ij} = q_j \qquad (5)$$

In equations (4) and (5), the pressure head *h* (m) is defined at nodes or junctions in the network, shown as filled circles in Figure 1 and Figure 2. The junction index is denoted by the subscript. The flow rate *Q* (m^3/s) is directionally defined between two nodes *i* and *j* as Q_{ij}. Eq. (4) relates the drop in pressure to the friction in a pipe. Eq. (5) relates the mass inflow into a junction to an imposed inflow or outflow q_i, where the summation only applies to the *i* nodes connected to node *j*.

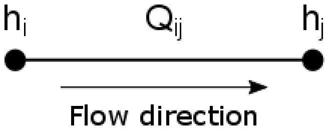

Figure 1: Head loss along a pipe

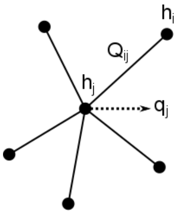

Figure 2: Conservation of mass at a junction

The constant K_{ij} and exponent *n* in Eq. (4) are taken from different empirical expressions. For example, the Darcy-Weisbach equation (Roberson and Crowe, 1997) has:

$$K_{ij}=\frac{8fL}{\pi^2 g d^5}, \qquad n=2 \qquad (6)$$

where *f* is the empirical Darcy friction factor, *L* is the pipe length and *d* is the pipe diameter. The Hazen-Williams equation (Yıldırım and Özger, 2009), which is used in this implementation has:

$$K_{ij}=\frac{10.67L}{C^{1.85}d^{4.87}}, \qquad n=1.85 \qquad (7)$$

where *C* is a pipe roughness coefficient, typically ≈100. The equations are solved for a system of pipes by substitution of Eq. (4) into Eq. (5) and minimising the residual *F*:

$$F=\sum_i Q_{ij}-q_j \qquad (8)$$

Directionality of flow must be encoded in the set of equations to give the correct result. This is implemented by re-arranging Eq. (4) for *Q* and defining the flow in the direction of the head gradient as:

$$Q_{ij}=\operatorname{sign}\left(h_i-h_j\right)\left(\frac{\left|h_i-h_j\right|}{K_{ij}}\right)^{\frac{1}{n}} \qquad (9)$$

Substitution of Eq. (9) into Eq. (8) gives:

$$F_0(h_0,h_1\dots h_N)=s_{01}\left(\frac{\left|h_0-h_1\right|}{K_{01}}\right)^{\frac{1}{n}} + s_{02}\left(\frac{\left|h_0-h_2\right|}{K_{02}}\right)^{\frac{1}{n}}+\cdots + s_{0N}\left(\frac{\left|h_0-h_N\right|}{K_{0N}}\right)^{\frac{1}{n}}-q_0 \qquad (10)$$

Where the shorthand $s_{ij}=\operatorname{sign}\left(h_i-h_j\right)$ has been used and nodes *i* =0 and *j* = 1…*N* are used for illustration. Note that Eq. (10) only applies for node 0, and the entire system is represented by a set of such equations for F_1, F_2 … F_N. The system is solved using a multi-variate Newton-Raphson method over the pressure heads *h*. Using the vectors $\mathbf{h}=\{h_0,h_1\dots h_N\}$, $\mathbf{F}=\{F_0,F_1\dots F_N\}$ and applying a Taylor expansion to *F*(**h**) gives:

$$\mathbf{F}(\mathbf{h}+\delta\mathbf{h})=\mathbf{F}(\mathbf{h})+\mathbf{J}_F\delta\mathbf{h}+O(\delta\mathbf{h}^2) \qquad (11)$$

where $\mathbf{J}_F$ is the Jacobian matrix:

$$\mathbf{J}_F=\begin{pmatrix}\frac{\partial F_0}{\partial h_0} & \cdots & \frac{\partial F_0}{\partial h_N}\\ \cdots & \cdots & \cdots\\ \frac{\partial F_N}{\partial h_0} & \cdots & \frac{\partial F_N}{\partial h_N}\end{pmatrix} \qquad (12)$$

Neglecting higher order terms and setting the left hand side of Eq. (11) to zero gives:

$$\mathbf{J}_F \delta\mathbf{h} = -\mathbf{F}(\mathbf{h}) \tag{13}$$

Solving Eq. (13) for $\delta\mathbf{h}$ allows the head vector to be iteratively updated as $\mathbf{h} \rightarrow \mathbf{h} + \delta\mathbf{h}$. The algorithm is started using an initial guess vector $\mathbf{h}_0$. The head vector is updated until the relative difference between the initial and new vectors falls under a certain tolerance, $(\mathbf{h} \cdot \mathbf{h})/(\mathbf{h}_0 \cdot \mathbf{h}_0) < \varepsilon$. In our implementation ε = 0.001. The terms in the Jacobian matrix, Eq. (12) can be evaluated from Eq. (10). This gives:

$$s_{ij} \frac{\partial}{\partial h_i} \left(\frac{|h_i - h_j|}{K_{ij}} \right)^{\frac{1}{n}} = n^{-1} K_{ij}^{\left(-\frac{1}{n}\right)} |h_i - h_j|^{\left(\frac{1}{n}-1\right)} \tag{14}$$

$$s_{ij} \frac{\partial}{\partial h_j} \left(\frac{|h_i - h_j|}{K_{ij}} \right)^{\frac{1}{n}} = -n^{-1} K_{ij}^{\left(-\frac{1}{n}\right)} |h_i - h_j|^{\left(\frac{1}{n}-1\right)} \tag{15}$$

Note that Eq. (14) is the partial derivative with respect to h_i and Eq. (15) with respect to h_j, resulting in a negative sign on the right-hand side of Eq. (15). The sign functions also become unity after the derivative is taken. The expressions on the right-hand sides are defined as d_{ij}:

$$d_{ij} = n^{-1} K_{ij}^{\left(-\frac{1}{n}\right)} |h_i - h_j|^{\left(\frac{1}{n}-1\right)} \tag{16}$$

From Eqs. (10), (12) and (16) the resulting Jacobian matrix $\mathbf{J}_F$ is sparse and symmetric with a full diagonal. The diagonal elements are equal to the sum of d_{ij} over all j nodes connected to node i. The off diagonal elements are $-d_{ij}$ at row (column) i and column (row) j for all connections between nodes i and j. A resulting matrix example for a four node network is:

$$\mathbf{J}_F = \begin{pmatrix} \sum_j d_{0j} & 0 & -d_{02} & -d_{03} \\ 0 & \sum_j d_{1j} & -d_{12} & 0 \\ -d_{02} & -d_{12} & \sum_j d_{2j} & 0 \\ -d_{03} & 0 & 0 & \sum_j d_{3j} \end{pmatrix} \tag{17}$$

The steps for solving the pipe network are to firstly construct the right-hand side forcing vector of Eq. (13). All imposed flows are added to the vector, then Eq. (10) is evaluated for all connections to each node. Next the derivative, Eq. (16) is evaluated for each connection then added to the diagonal and the corresponding row and column of the Jacobian matrix. Once the stiffness vector and Jacobian has been constructed, the matrix equation (13) is solved for $\delta\mathbf{h}$. The vector of heads is then updated, the matrix and stiffness vector re-built and the matrix equation solved again until convergence in $\mathbf{h}$.

The method can be extended to incorporate nodes at end points at a fixed pressure. These nodes are called terminators, and only have one pipe attached. To implement these terminators, the vector $\mathbf{h}$ is simply extended to include these terminators at the end of the vector. The index j can then point to these vector locations, which are otherwise not updated in the solution.

The hydrodynamic and hydraulic models are coupled at the location of terminators within the hydrodynamic grid. The pressure head at a terminator is given by the height of the topography plus the water height, w, in their enclosing grid cell. Once all terminator pressures have been set, the network is solved. Solution of the network gives the flow rate in pipes attached to the terminator. This flow rate is integrated and added or subtracted from the water level in the enclosing grid cell. The coupling must allow for cases in which the pressure head is less than the water depth within a cell. For the case shown as in Figure 3, the height of water in grid cell 'A' could be zero, but the pressure in the terminator within cell 'A' could be non-zero due to the pressure from the lower cell 'B'. Terminators within cells are therefore implemented using a 'well', shown as vertical pipes in Figure 3. The height of water in these wells can be below the base level of the cell. When the height of the water in a grid cell is below zero, the height within the well is used to calculate the terminator pressure. The lowest depth of all wells in the simulation is set to the lowest point in the simulation bathymetry.

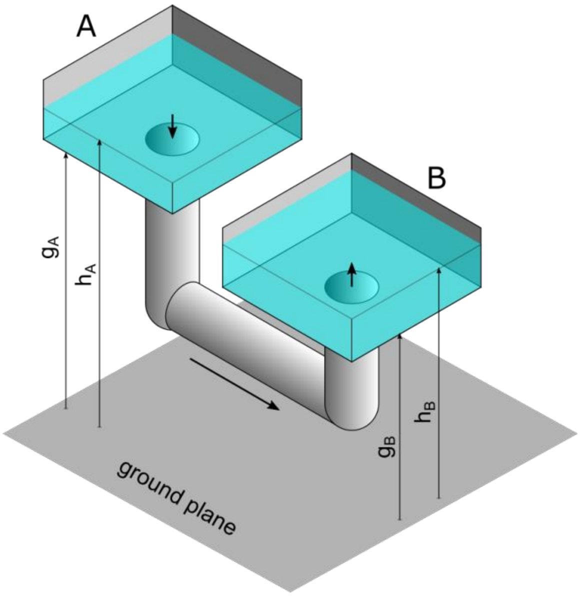

Figure 3: Connection between one-dimensional hydraulic network and two-dimensional hydrodynamic model

COMPUTATIONAL METHOD

The two-dimensional shallow-water equations are discretised onto a fixed grid representing the land surface and computationally processed to predict the water dynamics. Numerical implementations of the shallow water equations must ensure both positive water heights over the domain and not introduce spurious oscillations when the water surface is at rest. We use the 'well-balanced' method given by Kurganov and Petrova (2007) to enforce both of these conditions.

For the pipe network solver the matrix expression, Eq. (13), was solved using a preconditioned conjugate gradient algorithm. A Jacobi preconditioner was used because of the ease of implementation in a parallel environment. The choice of algorithm was chosen due to the sparsity, symmetry and positive definiteness of the matrix (approximately 8,000 × 8,000 with around 24,000 non-zero entries for the urban pipe network studies). The conjugate gradient is very suited to matrices of this type and is based on two major components, a sparse matrix vector product and a dot product. Both of these are parallelisable and the matrix-vector product requires only non-zero elements to be stored. A wide range of possible variations of matrix storage implementations were investigated on the GPU: coordinate format, ELLPACK format, compressed sparse row format, diagonal format and block based formats (Stanimirovic and Tasic, 2009). Unfortunately, the unstructured nature of large pipe networks made the certain formats ineffective. For example, the number of entries on each row or column was the number of connections to each node, which could range from two upwards. Storage optimised for a certain number of sparse elements per row could not be used. As the connections were unstructured, optimisations based on a banded structure also could not be used. Diagonal format and block based formats were therefore ruled out. Out of the remainder sparse matrix formats, the compressed sparse row format was selected for use because it was found to have marginally better performance for this particular application over coordinate and ELLPACK formats. Initial experiments in reordering the pipe network to provide better conditioning of the matrix were not found to improve overall performance, although more sophisticated re-ordering algorithms will be trialled for future optimisation improvements.

However, a number of optimisations were possible. As the structure of the network does not change, the matrix was implemented on the GPU as a continuous memory block. At the initialisation of the algorithm, the locations of the non-zero matrix indices were stored. These were then calculated in order and copied to the GPU in a single write to optimise memory access. The conjugate gradient method on the GPU was divided into a number of kernels with the division between the kernels dependant on the reduction operations in the conjugate gradient method. The requirement to return to the host after each reduction operation (in OpenCL) was found to be the greatest barrier to optimisation. Architectures which could be tricked into evaluating the entire conjugate gradient method within one kernel were up to twenty times faster than those split at reduction points. However, these fast kernels were not functional on most OpenCL platforms.

DRAINAGE INVESTIGATIONS

Four simple test cases were used to obtain confidence in the solver performance. These consisted of a tank filled to 0.5 m depth adjacent to an empty identical tank connected together via simple pipe networks (Figure 4). All pipes had a diameter of 0.5 m and a Hazen-Williams pipe coefficient (C) of 100.

Idealised flow from one tank to the other via the pipe network can be modelled in a simple spreadsheet (using the head loss equation and conservation of mass equation) if one assumes that the water height in each tank remains uniform and that there is no sloshing around the tank as water is transferred. Full simulations using Swift which has the coupled hydrodynamic and pipe network algorithm were also completed. The results from both methods (Figure 5) show that the idealised solutions drain water slightly faster between the tanks than the fully coupled Swift solution. All cases result in both tanks ending up filled to the same depth. Since the water drains in the expected manner between the tanks, this provides baseline confidence in the highly complex coupled solver.

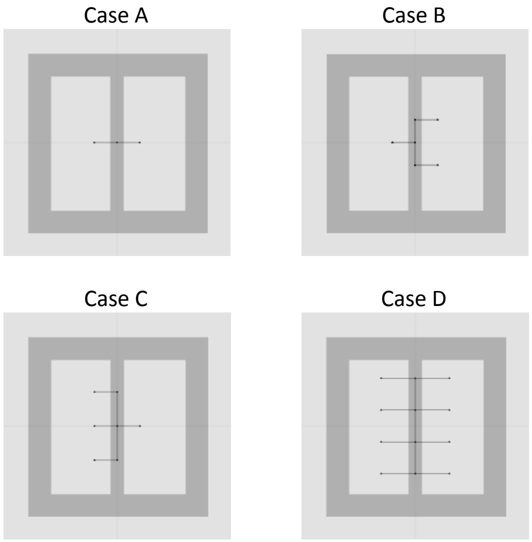

Figure 4: Two tanks (26 m × 59 m) connected together via different pipe networks. The shaded areas represent different base heights: light grey is 0 m and dark grey is 1 m. The pipe networks are shown by the connected line segments. The tank on the left is initially filled to a depth of 0.5 m in each case.

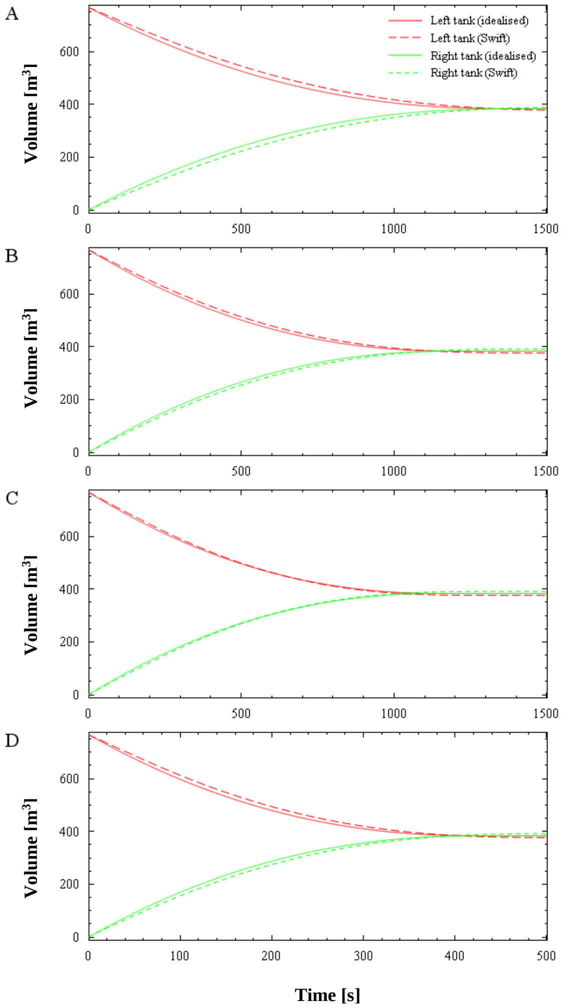

Figure 5: Heights in each of the tanks for the simple pipe network test cases.

APPLICATION TO URBAN FLOODING

The Swift model was used for flood predictions in the City of Port Phillip (CoPP) in Melbourne which has many low-lying areas that are prone to flooding including storm surge, heavy rainfall and catchment flooding. CoPP is a large region which covers approximately 20.62 km^2. A combined topographic and bathymetric map was composited from separate data sources and downscaled to a horizontal grid resolution of 4 m (Figure 6). Underground drains in the CoPP were incorporated into the hydrodynamic model using a 1D pipe network model consisting of approximately 8,000 pipes with a maximum of 8 pipes connecting at a junction and approximately 24,000 non-zero entries in the Jacobian matrix (Figure 7). The scenarios considered in this study included combinations of storm surge events (based on data from Melbourne Water); 1 in 5, 1 in 10 and 1 in 100 ARI rainfall events; and 0.4 m, 0.8 m, and 1.1 m future sea level rises.

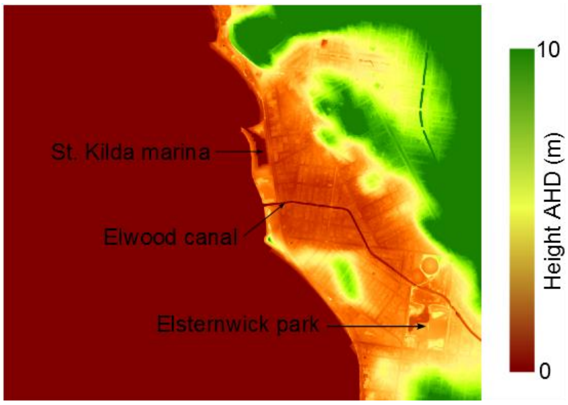

Figure 6: Combined topography and bathymetry for Port Phillip simulations.

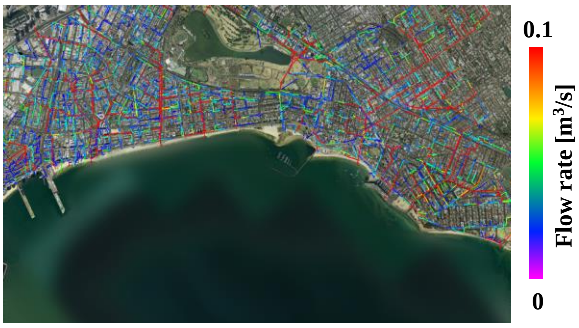

Figure 7: Flow in drainage network for the City of Port Phillip 3 hours after a 1 in a 10 year ARI rainfall.

The Swift solver took 14 hours to predict 24 hours of flooding with approximately 80% of the time being spent in the pipe network calculation. There were approximately 3-4 Newton-Raphson iterations per time-step, each requiring around 150 conjugate gradient iterations. The multiple matrix solves per time step during the relaxation, highlights the need for optimising this part of the algorithm.

A number of flood mitigation options were suggested by the CoPP which were added to the model to study their effects. These included non-return valves along the canal and in some drains; street level retention detention systems; household retention using rainwater tanks; and a large scale retention system in a flood prone area.

The CoPP council previously identified several hot spots that are highly susceptible to flooding in the region, including the corner of Meredith and Barkly Streets in Elwood (Figure 8). A simulation of flooding for present day sea levels without any mitigations applied is able to accurately capture this flooding around the corner of Meredith and Barkly Streets (Figure 9a). Applying all the mitigation options and running the simulation again predicts no flooding in this vicinity (Figure 9b). This suggests that the mitigation options are effective for the present day sea level conditions. Another example of the impact of these mitigation options on the localised flooding for future sea level rise of 0.4 m is shown in Figure 10. The extent of the flooding is dramatically reduced by the mitigation options. This demonstrates that integrated mitigation modelling is a practical way of providing accurate cost-benefit analysis data.

Overall feedback from the CoPP from this study included that the simulation flood hot-spots were consistent with the historical experiences during previous floods. Additionally, flow reversal back up the pipe networks in the simulation was found in the correct streets. This critical feature causes flooding that can’t be captured by hydrodynamic modelling alone.

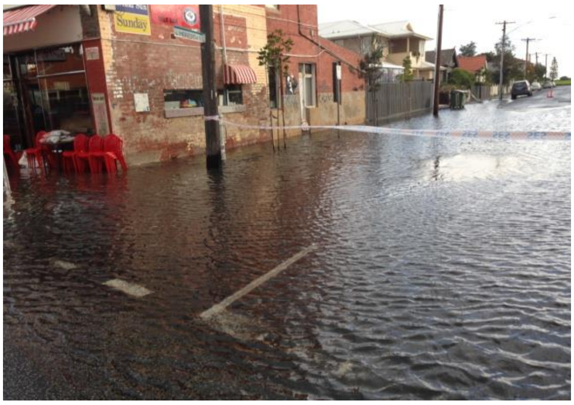

Figure 8: Photograph of recent flooding at the corner of Meredith and Barkly Streets in Elwood (supplied by the City of Port Phillip).

CONCLUSION

GPUs were shown to be well suited for grid-based computational methods such as computation of the shallow water equations. This result is not unexpected as the equations only rely on local stencil operations and GPUs are designed to operate efficiently on similarly structured texture data. For solving hydraulic networks GPUs can improve the speed of the matrix calculations, but the improvement is not as great as for grid-based raster operations due to the random access pattern and scattered memory access. The conjugate gradient iterations on the hydraulic part of the model is the main bottleneck for the simulations considered in this study, but there was still a reasonable improvement. A 22 hour simulation on an 8-core CPU was reduced to 14 hours on a GPU using the same OpenCL-based code. Running the entire conjugate gradient method within a single GPU kernel was found to greatly improve computational speed. Unfortunately, this relied on implementation of a global synchronisation barrier, which is unsupported in OpenCL. Currently, only a few cards can do this through undocumented means. Future version of OpenCL may have global synchronisation. CUDA supports global synchronisation, but is not a viable option due to the portability requirements of Swift. Overall GPUs were found to help make integrated hydrodynamic-hydraulic modelling faster and allow more practical investigations of mitigation options by engineers and councils.

REFERENCES

BOULOUS, P.F., LANSEY, K.E., KARNEY, B.W., (2006), “Comprehensive Water Distribution Systems Analysis Handbook for Engineers and Planners”, 2nd Ed., MWH Soft Press, Pasadena, Calif.

BRADFORD, S.F., SANDERS, B.F., (2002), “Finite-volume model for shallow-water flooding of arbitrary topography”, *J. Hydraul. Eng.*, **128**, 289–298.

HENONIN, J., RUSSO, B., MARK, O., GOURBESVILLE, P., (2013), “Real-time urban flood forecasting and modelling – a state of the art”, *J. Hydroinformatics*, **15**, 717.

KURGANOV, A., PETROVA, G., (2007), “A second-order well-balanced positivity preserving central-upwind scheme for the Saint-Venant system”, *Commun. Math. Sci.*, **5**, 133–160.

MARK, O., WEESAKUL, U., CHALIRAKTRAKUL, C., (2002), “A real-time hydrological information system for Bangkok”, *Proc. International Conference on Urban Hydrology for the 21st Century*, Kuala Lumpur, Malaysia.

RAINGAIN. (2013). Retrieved September 25, 2015, from http://www.raingain.eu/en/four-cities-gain-rain

RAYMOND, M., PEYRON, N., MARTIN, A., (2006), “ESPADA, A unique flood management tool: first feedback from the September 2005 flood in Nîmes”, *Proc. 7th International Conference on Hydroinformatics*, Nice, France.

RENÉ, J.-R., DJORDJEVIĆ, S., BUTLER, D., MADSEN, H., MARK, O., (2014), “Assessing the potential for real-time urban flood forecasting based on a worldwide survey on data availability”, *Urban Water J.*, **11**, 573–583.

Roberson, J.A., Crowe, C.T., (1997), “Engineering Fluid Mechanics”, 6th ed. John Wiley & Sons.

STANIMIROVIC, I.P., TASIC, M.B., (2009). “Performance comparison of storage formats for sparse matrices”, Ser. Math. Inform, **24**, 39–51.

YILDIRIM, G., ÖZGER, M., (2009), “Neuro-fuzzy approach in estimating Hazen–Williams friction coefficient for small-diameter polyethylene pipes”, *Adv. Eng. Softw.*, **40**, 593–599.

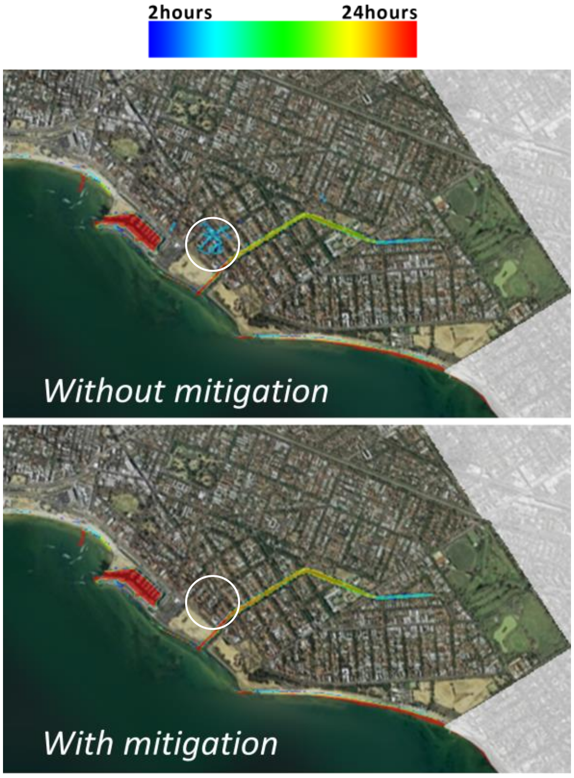

Figure 9: Comparison of the storm-surge flood water retention times in the case of present day sea levels with no rainfall; (a) without mitigation; and (b) with mitigation. The circled region indicates the vicinity of Meredith and Barkly Streets in Elwood.

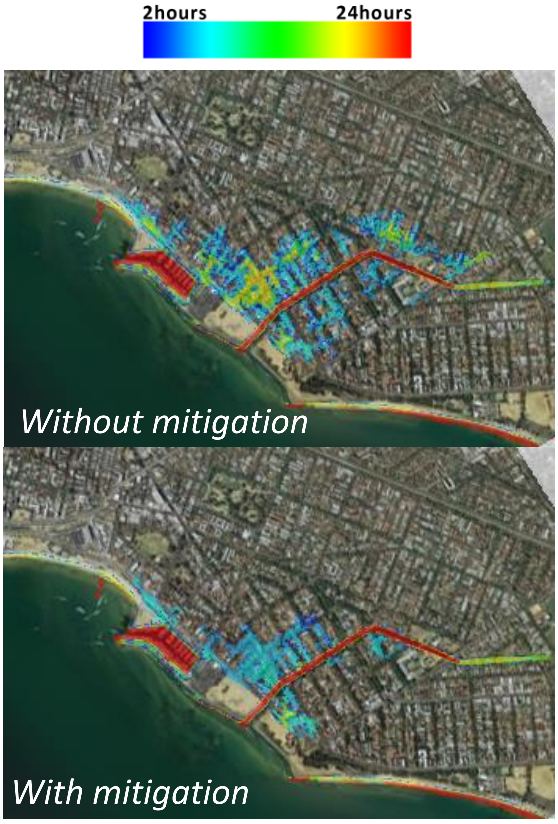

Figure 10: Comparison of the storm-surge flood water retention times in the case of 0.4 m sea level rise with no rainfall; (a) without mitigation; and (b) with mitigation.