MATH 4701 Project:
River Flow Modelling

Andrew Musgrave

October 10, 2020

Contents

1 Introduction

In many cases, the easiest way to understand rivers is historically: to guess what a river might do in the future, one looks back at what it has done in the past. However, due to changes in weather patterns brought about by climate change, historical trends may not hold and causal models which can relate the expected changes in weather patterns to changes in river flow will become increasingly important.

This report will discuss modelling approaches used to forecast river flow, with a focus on the simulation of open channel flow using the St. Venant equations. These equations are solved numerically using a finite volume scheme implemented on a graphics processing unit using the Thrust toolkit. Additionally, results from a neural network trained to predict ten days of river flow from the previous 60 days of flow, temperature and precipitation data will be presented. The report is divided into two sections. The modelling section will provide an overview of methods used to predict river flow before focusing on the St Venant equations of unsteady flow in open

channels and their numerical solution using the finite volume method. The computation section will begin with a comparison of GPU and CPU computing, and then present the results from a GPU implementation of a finite volume scheme used to solve the St. Venant equations in a square channel. Finally, the results of an ANN trained to produce ten day forecasts of river flow for the Mississippi river in Appleton, Ontario will be presented.

2 Modelling

2.1 Considerations for Modelling River Flow

There are many factors which influence the flow of rivers. Water flows into rivers from a variety of sources including rainfall, snowmelt, and ground water [1]. Predicting the quantity of water arising from each of these sources requires an in-depth understanding of both the governing phenomenon and the geography of the river basin. Once in the river, the movement water through the channel has an intricate dependence on the slope and geometry of the channel which must also be accounted for.

Given the myriad of factors which affect river flow, a wide variety of modelling approaches have been devised to produce meaningful forecasts from limited data. The inputs and outputs of the methods vary based on the river basin and the goal of the forecast. For example, a model intended to predict spring flooding in region with snowy winters would focus on snowmelt as the primary input, whereas a model intended to predict drinking water availability in a dry region would focus on groundwater as the primary input [1].

Modelling approaches also vary greatly in their level of abstraction and the assumptions they rely on. Perhaps the least abstract method is to model river flow using the St. Venant equations, which are a system of partial differential equations describing the flow of liquid in an open channel. Although this approach has the potential to provide direct insight into the flow of water in the river channel, it requires exact knowledge of the channel geometry and inflow locations which is often not feasible for large river basins. For this reason, the St. Venant equations are generally only applied to short segments of river, such as predicting the flow of floodwater between one river gauge and a second gauge further downstream [1].

At intermediate levels of abstraction sit conceptual models which attempt to capture the behaviour of a river using a simpler hydraulic system, such as a network of linear reservoirs. [2] If used properly these simplified models can provide a good proxy for the river itself, but forming an appropriate conceptual model and finding the correct parameters for a given river system is not always an easy task.

At the highest level of abstraction sit black-box models which simply aim to use known input and output data to form a model which can be used to predict future outputs. The predominate black-box approach used to predict river flow from runoff is the linear systems approach, where the goal is to find a system impulse response (or 'unit hydrograph') $h(t)$ such that the output $y(t)$ is given by convolving the input $x(t)$ with $h(t)$ [2]. A more recent black-box approach is to use an artificial neural network (ANN) to predict river flow from previous weather and flow data [3]. ANNs are advantageous because they can use any number of inputs to predict river flow and don't assume linearity of the system, but they require a large quantity of training data to learn the correct relationship between inputs and outputs.

The primary focus of this report is numerical solution of the St. Venant equations in a vertical-walled channel. This sits at the lower extreme of the abstraction spectrum: it allows for direct simulation of water level and flow rate at each point along the length of the channel, but in a simplified context which is far removed from the forecasting river flow in a real-world scenario.

To touch upon the other extreme of the abstraction spectrum, an artificial neural network will be trained using hydrometric data from the Mississippi river in Appleton, Ontario and weather data from the Ottawa International Airport. This model will be used to produce ten day forecasts from past data, which will then be compared to the actual flow rate from that time period.

2.2 St. Venant Equations for Unsteady Flow in Open Channels

The flow of fluid in an open channel can be described using the St. Venant equations. These equations allow the flow in an open channel at position x and time t to be solved using a pair of partial differential equations: the continuity equation (1) and the momentum equation (2) [1].

$$\frac{\partial Q}{\partial x} + \frac{\partial A}{\partial t} = r(x,t) \tag{1}$$

$$\frac{\partial y}{\partial x} + \frac{u}{y}\frac{\partial u}{\partial x} + \frac{1}{g}\frac{\partial u}{\partial t} = S_{eff} - \frac{u}{gy}r(x,t) \tag{2}$$

Here Q is flow rate, A is cross sectional area, r is lateral inflow, y is depth of flow, u is mean velocity of flow, g is gravity, and S_eff is the effective slope of the channel bed. The variables in (1) and (2) are related by $Q(x,t) = u(x,t)A(x,t)$ and $A(x,t) = f(x, y(x,t))$, with $f(x,y)$ dictated by the cross sectional profile of the channel at position x. In the simplest case is that of a vertical walled channel of unit width. In this case, the relationship is simply $A = y$ and $Q = uy$. This case will be the focus of this report, with the additional simplification that the analysis will be done in units where $g = 1$.

2.3 Formulation as System of Conservative Equations

When modelling phenomena where certain quantities must remain constant (for example, matter must neither appear nor disappear), it can be advantageous to express the situation as a system of conservation laws. Such a system is of the form:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial}{\partial x}\mathbf{f}(\mathbf{U}) = 0 \tag{3}$$

Where $\mathbf{f}(\mathbf{U})$ is called the flux. The result is that the quantity $\mathbf{U}$ is conserved, which is to say that the integral of $\mathbf{U}$ over x is the same for all t [4]:

$$\int \mathbf{U}(x,t)dx = \text{Constant} \tag{4}$$

Equation 3 can be generalized to include a source term $\mathbf{S}$, allowing the previously conserved quantities to change within the model.

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial}{\partial x}\mathbf{f}(\mathbf{U}) = \mathbf{S} \tag{5}$$

Comparing (5) to the open channel equations, we find that (1) is already expressed as a conservation law, with $\mathbf{U} = (Q, A)$. Here Q is the flux term for A and $r(x,t)$ is the source term for A. Using the relationship for a vertical-walled channel, equation 2 can be expressed as a conservation law in Q giving the following set of conservation equations:

$$\frac{\partial Q}{\partial t} = -\frac{\partial}{\partial x}\left(\frac{A^2}{2} + \frac{Q^2}{A}\right) + AS_{eff}(x) \tag{6}$$

$$\frac{\partial A}{\partial x} = -\frac{\partial}{\partial x}Q + r(x,t) \tag{7}$$

2.4 The Finite Volume Method

When when using numerical methods to solve systems which can be expressed as a set of conservation laws, special care must be taken to ensure that the numerical solution doesn't violate (4) in order for that solution to be valid. A simple approach such as the finite difference method is often not sufficient. One approach which is more suitable to this type of equation is the finite volume approach [5]. In this approach the spacial domain is divided into a set of finite intervals centered at x_j, with boundaries at $x_{j\pm\frac{1}{2}}$. The solution scheme then seeks to find the average value of the conserved variables in each volume, which it updates for each timestep by calculating the flux between adjacent volumes at the $x_{j\pm\frac{1}{2}}$ boundaries. The approach can be stated more explicitly as follows:

The finite volume method seeks to solve the evolution of some vector $\mathbf{v}$ defined on volumes centred at x_j at times t_n such that $\mathbf{v}_j^n = \mathbf{v}(x_j, t^n)$ using the flux $\bar{\mathbf{f}}$ flowing between volumes at the boundaries $x_{j\pm\frac{1}{2}}$ [5]:

$$\mathbf{v}_j^{n+1} = \mathbf{v}_j^n - \frac{\Delta t}{\Delta x}\left(\bar{\mathbf{f}}_{j+\frac{1}{2}}^n - \bar{\mathbf{f}}_{j-\frac{1}{2}}^n\right) \tag{8}$$

To use this method, an approximation of the flux $\bar{\mathbf{f}}$ must be obtained from the conservation laws which define the system. Such an approximation must keep track of the direction which information is flowing at each point. This can be done by analysing the Jacobian of the function $\mathbf{f}$ which defines the system. For hyperbolic systems, the Jacobian will have real eigenvalues $\lambda_1(\mathbf{v}) \leq \lambda_2(\mathbf{v}) \leq ... \leq \lambda_m(\mathbf{v})$ These eigenvalues can be used with their associated eigenvectors to diagonalize the Jacobian.

$$J_{j\pm\frac{1}{2}}^n = P_{j\pm\frac{1}{2}}^n \begin{pmatrix} \lambda_1(\mathbf{v}_{j\pm\frac{1}{2}}^n) & 0 & \cdots & 0 \\ 0 & \lambda_2(\mathbf{v}_{j\pm\frac{1}{2}}^n) & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_m(\mathbf{v}_{j\pm\frac{1}{2}}^n) \end{pmatrix} (P_{j\pm\frac{1}{2}}^n)^{-1} \tag{9}$$

In it's diagonalized form, the Jacobian expresses the problem as a system of nonlinear transport equations. Simply put, the eigenvector at each position defines a quantity which is being transported at that position, and corresponding eigenvalue gives the velocity at which it is being transported. Therefore the signs of the eigenvalues give the direction in which the corresponding quantity is moving at each position. This is what informs the characteristic flux approximation which can be used in the finite volume method.

This method makes use of a matrix $U_{j\pm\frac{1}{2}}^n$ which is the sign matrix of the Jacobian at $(x_{j\pm\frac{1}{2}}, t^n)$:

$$U_{j\pm\frac{1}{2}}^n = P_{j\pm\frac{1}{2}}^n \begin{pmatrix} \operatorname{sgn}\left(\lambda_1(\mathbf{v}_{j\pm\frac{1}{2}}^n)\right) & 0 & \cdots & 0 \\ 0 & \operatorname{sgn}\left(\lambda_2(\mathbf{v}_{j\pm\frac{1}{2}}^n)\right) & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \operatorname{sgn}\left(\lambda_m(\mathbf{v}_{j\pm\frac{1}{2}}^n)\right) \end{pmatrix} (P_{j\pm\frac{1}{2}}^n)^{-1} \tag{10}$$

Using this sign matrix, the numerical characteristic flux g^{CF} between the nodes x_j and x_{j+1} is calculated as follows [5]:

$$g^{CF}(\mathbf{v}_j^n, \mathbf{v}_{\mathbf{j+1}}^{\mathbf{n}}) = \frac{\mathbf{f}(\mathbf{v}_j^n) + \mathbf{f}(\mathbf{v}_{j+1}^n)}{2} - U_{j+\frac{1}{2}}^n \frac{\mathbf{f}(\mathbf{v}_{j+1}^n) - \mathbf{f}(\mathbf{v}_j^n)}{2} \tag{11}$$

If this characteristic flux is used as an approximation of the numerical flux in (8) such that $\bar{\mathbf{f}}^n_{f+\frac{1}{2}} \approx g^{CF}(\mathbf{v}^n_j, \mathbf{v}^n_{\mathbf{j+1}})$, then the following three-point scheme is obtained:

$$\mathbf{v}^{n+1}_j = \mathbf{v}^n_j - \frac{\Delta t}{\Delta x}\left(\frac{\mathbf{f}(\mathbf{v}^n_{j+1}) - \mathbf{f}(\mathbf{v}^n_{j-1})}{2} - U^n_{j+\frac{1}{2}}\frac{\mathbf{f}(\mathbf{v}^n_{j+1}) - \mathbf{f}(\mathbf{v}^n_j)}{2} + U^n_{j-\frac{1}{2}}\frac{\mathbf{f}(\mathbf{v}^n_j) - \mathbf{f}(\mathbf{v}^n_{j-1})}{2}\right) \tag{12}$$

2.5 Jacobian of the St. Venant Equations

By comparing conservative form the St. Venent equations (6 and 7) to the general form of a conservative system with a source term (5), the terms can be identified as follows:

$$\mathbf{U} = \begin{pmatrix} Q \\ A \end{pmatrix}, \quad \mathbf{f}(\mathbf{U}) = \begin{pmatrix} \frac{A^2}{2} + \frac{Q^2}{A} \\ Q \end{pmatrix} \quad \mathbf{S} = \begin{pmatrix} AS_{eff}(x) \\ r(x,t) \end{pmatrix} \tag{13}$$

Differentiating $\mathbf{f}$, the following Jacobian is obtained:

$$J = \begin{pmatrix} \frac{2Q}{A} & A - \frac{Q^2}{A^2} \\ 1 & 0 \end{pmatrix} \tag{14}$$

The Jacobian has two eigenvalues:

$$\lambda_1 = \frac{Q}{A} - \sqrt{A} \tag{15}$$

$$\lambda_2 = \frac{Q}{A} + \sqrt{A} \tag{16}$$

From these eigenvalues, a distinction between two flow regimes can be made: if $\left|\frac{Q}{A}\right| < \sqrt{A}$, one eigenvalue will have a positive sign and the other will have a negative sign, indicating that information can flow from that point in both directions. If $\left|\frac{Q}{A}\right| > \sqrt{A}$ both eigenvalues will have the same sign, and information can only flow in one direction. In fluid dynamics the former case is referred to as subcritical flow, and the latter is referred to as supercritical flow.

3 Computation

3.1 Graphics Processing Units

The processing workload of a computer is generally split between two devices: the central processing unit (CPU) and the graphics processing unit (GPU). As its name implies, the GPU is traditionally used for rendering graphics whereas the CPU serves as the main control center of the computer. This is because graphics processing involves the solution of many simple equations which can be solved in parallel, and the GPU has a large number or cores to accommodate these massively parallel computations. More recently, the massive parallelization opportunities afforded by GPUs have been put to use in other applications including artificial intelligence and scientific computing.

One factor which has slowed the adoption of GPU computing is the specialized programming required to implement numerical calculations on a GPU. The basic platform used for GPU programming is CUDA, which is an extension of C++ allowing data to be transferred to the GPU and operated on with specially-defined functions. Although CUDA is very powerful, it is not very intuitive and required in-depth understanding of GPU architecture to use properly. Fortunately, the there is an open-source toolkit called Thrust which builds on top of CUDA to offer a high-level

interface for GPU computing. Using thrust, one can define vectors on the GPU and define operations to be applied to these vectors. The operations are then applied element-wise and in parallel on the GPU.

3.2 GPU Performance Comparison

As part of the course materials for MATH 4701, Professor Lorin provided a domain-decomposition method of solving the one dimensional transport equation implemented using MPI (message passing interface) in C++. To verify the effectiveness of Thrust for the numerical solution of differential equations, and to compare the performance of GPU computing to domain-decomposition on multi-processor CPUs, the same equation was solved for the same number of grid points and time steps on the GPU using Thrust. A comparison in running time between the GPU implementation and the MPI implementation using various numbers of processors is summarized in the table below.

Implementation	**Running Time (ms)**
MPI (2 processors)	100
MPI (4 processors)	70
MPI (8 processors)	50
MPI (16 Processors)	30
GPU	40

As seen above, the performance of the GPU implementation greatly exceeds that of the CPU implementation when 2 or 4 CPU processors are used, and still offers a slight improvement when 8 CPU processors are used. When 16 processors are used the CPU implementation runs slightly faster than the GPU implementation. However, it is worth noting that the advantage offered by the GPU depends greatly on the specifics of the problem being solved. If more spatial grid points are added the time taken by the CPU will increase because each processor must handle more points, but the time taken by the GPU will not increase so long as the number of grid points does not exceed the number of cores in the GPU. For very large simulations the optimal approach would likely be to have many GPUs with each GPU handling as many grid points it has cores, with domain decomposition techniques used to divide the problem between the GPUs.

3.3 Implementation of Finite Volume Method using Thrust

As the primary objective of the project, a numerical scheme to apply the finite volume method to the St. Venant equations was derived and implemented for solution on the GPU using Thrust. The scheme was used to simulate several scenarios to investigate the behaviour of the model. The value of Δt was updated every five timesteps based on the largest eigenvalue magnitude present in the simulation to ensure that the solution remained stable.

3.3.1 Wave Propagation with no Net Flow

The first case investigated is the wave propagation resulting from a Gaussian initial area profile in a basin with no net flow. Stated more formally, equations (6) and (7) are solved on $x \in [0, 20]$ subject to:

$$Q(x, 0) = 0$$

$$A(x,0) = 1 + 0.1e^{-5(x-10)^2}$$

$$S_{eff}(x) = 0$$

$$r(x,t) = 0$$

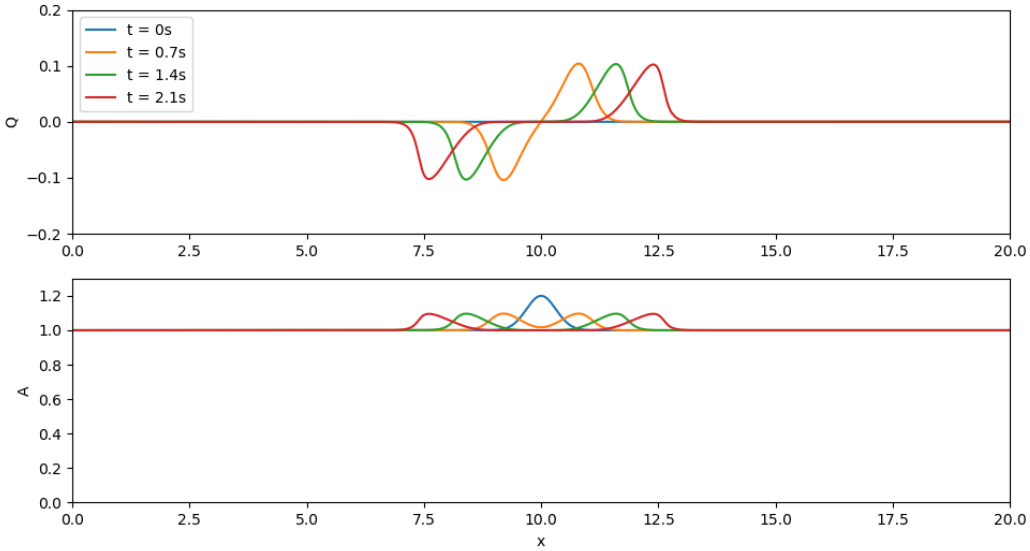

Figure 1: Wave propagation observed in simulation of St. Venant equations with no source terms, $Q(x,0) = 0$, and $A(x,0) = 1 + 0.1e^{-5(x-10)^2}$

The result of this simulation can be seen in figure 1. The simulation shows that the initial disturbance produces a pair of waves which propagate outwards from the center of the basin. Note the steepening of the leading edge of the waves, which is characteristic of wave propagation in shallow water.

3.3.2 Flow over Region with Positive Effective Slope

To investigate the effects of source terms on the evolution of the system, simulations including each of the two source terms were run starting from constant initial conditions. To include the source term of the momentum equation (7) a region of positive effective slope was included using the following conditions:

$$Q(x,0) = 1$$

$$A(x,0) = 2$$

$$S_{eff} = \begin{cases} 0 & 0 \leq x \leq 5 \\ 1 & 5 < x < 10 \\ 0 & 10 \leq x \leq 20 \end{cases}$$

$$r(x,t) = 0$$

Snapshots of the resulting channel flow at $t = 0$s, $t = 2.6$s, $t = 4.6$s and $t = 6.5s$ are shown in figure 2. Note that at the start of the downward slope ($x = 5$) the cross sectional area drops rapidly without a corresponding decrease in flow rate, showing the acceleration of the water as it passes over the lip of the drop. Downstream of the slope there is a hydraulic jump where the cross sectional area increases suddenly and the flow becomes subcritical. The location of the jump moves downstream as time progresses.

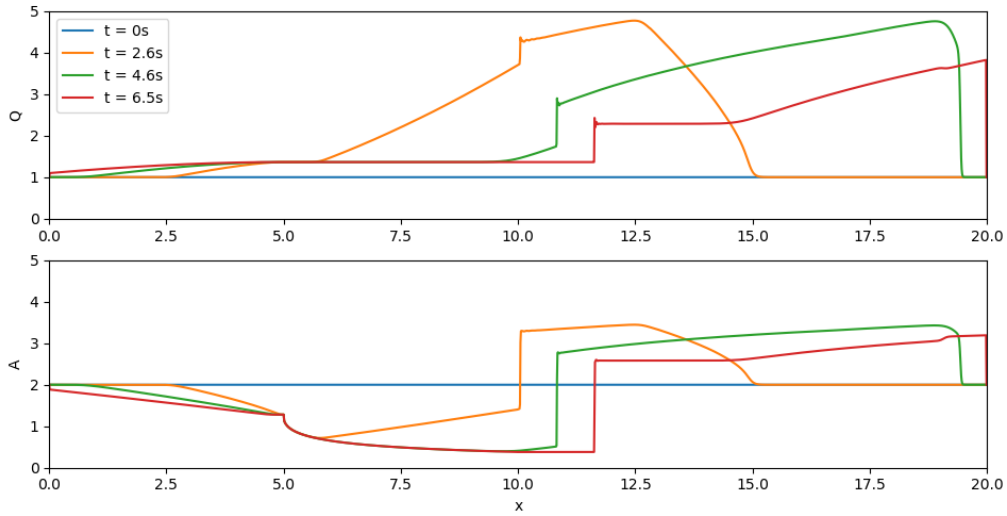

Figure 2: Flow over a downward slope of $S_{eff} = 1$ from $x = 5$ to $x = 10$

3.3.3 Lateral Inflow to Channel with Constant Flow

To investigate the effect of the continuity equation source term (6) and simulate the injection of water into a flowing channel, the simulation was run with the inclusion an $r(x,t)$ term. The $r(x,t)$ term was chosen to correspond to the constant injection of one unit of flow at the center of the channel, with the spatial profile of the lateral inflow given by a normalized gaussian:

$$Q(x,0) = 0.5$$
$$A(x,0) = 1$$
$$S_{eff}(x) = 0$$
$$r(x,t) = \frac{1}{0.1\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-10}{0.1}\right)^2}$$

The result is shown in figure 3. Here it can be seen that the channel flow is backed up upstream of the inflow location, indicated by the increase in cross sectional area and decrease in flow rate in this region. Downstream of the inflow location the flow rate and cross sectional area are increased from their initial values as water from the inflow site makes its way downstream.

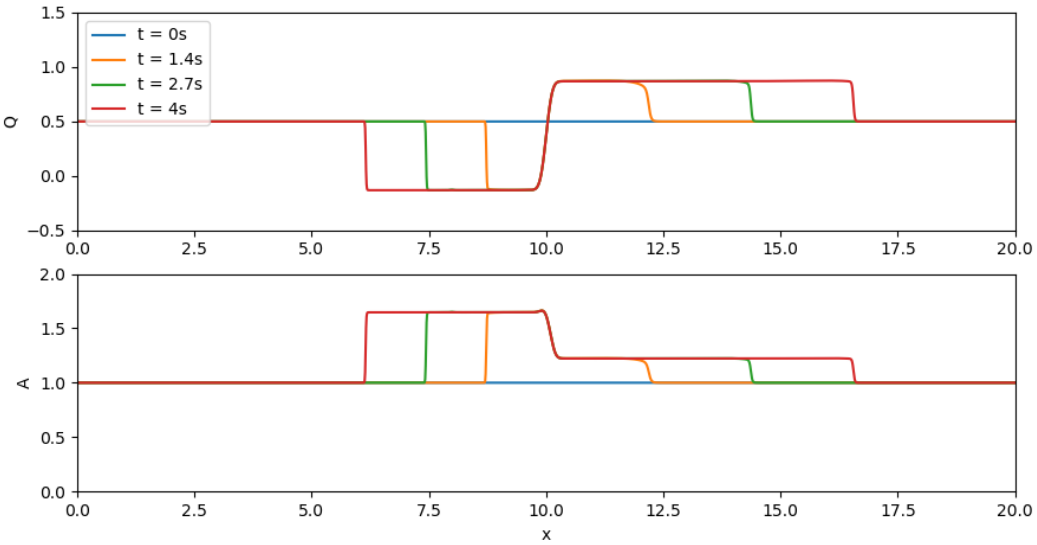

Figure 3:

Note that the initial flow in the above case is subcritical: the Jacobian has one negative eigenvalue and one positive eigenvalue. This is what allows the disturbance created by the inflow at $x = 10$ to propagate smoothly upstream. To compare the effects of inflow on a supercritical channel (where both eigenvalues are positive) the above example was run with a higher initial flow rate:

$$Q(x, 0) = 2$$
$$A(x, 0) = 1$$
$$S_{eff}(x) = 0$$
$$r(x, t) = \frac{1}{0.1\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-10}{0.1}\right)^2}$$

The results of this simulation is shown in figure 4. Note that at the location of the inflow the cross sectional area is much higher than its initial value and the flow velocity Q/A is decreased as momentum must be transferred to the incoming water. This drops the flow to the subcritical regime in this region allowing the disturbance to propagate upstream, albeit much more slowly than in the case where the entire channel is in the subcritical flow regime.

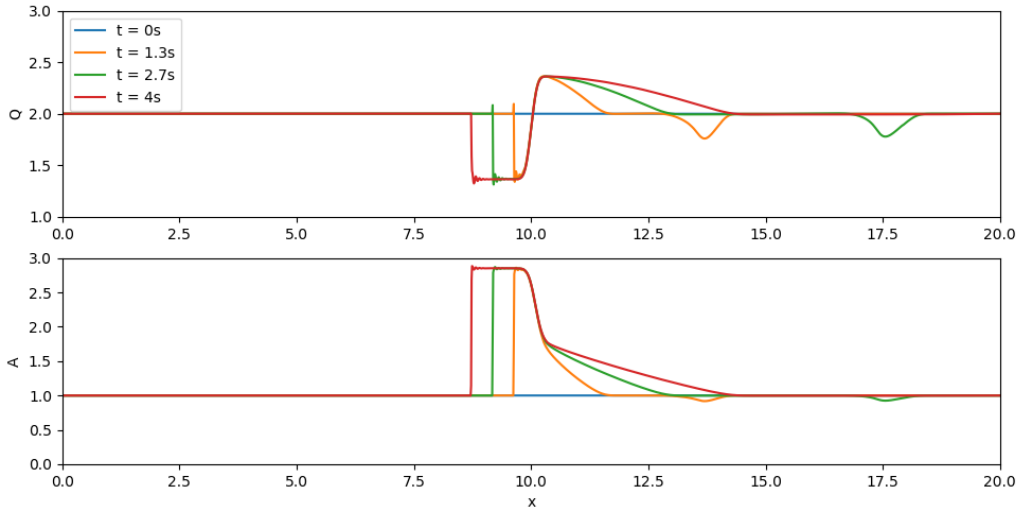

Figure 4:

3.4 River Level Prediction Using an Artificial Neural Network

An artificial neural network to forecast river flow was constructed and trained in python using the Tensorflow library. The site chosen for forecasting was the Missisippi river in Appleton, Ontario, Canada, which is the site of a hydrometric station run by Environment Canada. Data from this station was paired with weather data from the nearby Ottawa International Airport to produce the training and validation data for the ANN. Thirty-six years of data spanning from 1975-2011 were available for this site. From this data, three one-year runs of data (starting in 1978, 1993, and 2006) were set aside for validation, and the remainder was used as training data.

The artificial neural network is designed to forecast the flow rate of the Mississippi river in Appleton for the next ten days using the values of flow rate, temperature, and precipitation observed over the previous 60 days. This means its input is a 60x3 array and its output is a 10x1 array. As is common in time-series forecasting, the future values of the series are not predicted directly but rather the difference between each value and the previous day's value are predicted. These predicted differences are then used in conjunction with the known flow rate at the start of the forecast period to produce the forecast. Several neural network configurations were tested to find the configuration which produced the most accurate forecasts. The configuration which produced the best results was a single hidden layer consisting of a 20 node recurrent neural network (RNN) with ReLU activation. This layer was connected to the 10 node output layer using a dense layer with no activation function (i.e. linear).

Three examples of forecasts produced by this neural network using the validation dataset are shown in figure 5. In the first example, the prediction is in good agreement with the actual value for the first few days of the forecast period, before diverging towards the end of the forecast. In the second example the forecast and actual flow rates remain in good agreement for the duration of the forecast period. In the third example, the actual flow rate experiences a spike on the first day which was not predicted by the model, but then decays and approaches the predicted values by the end of the forecast period.

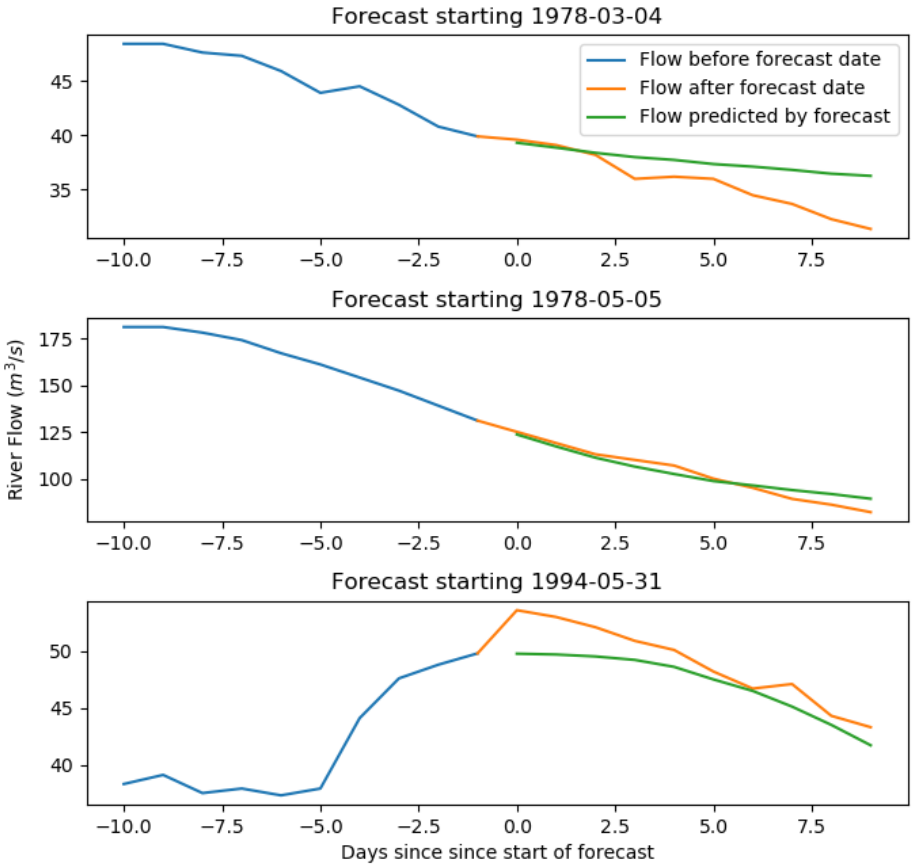

Figure 5: Flow rate forecasts for the Mississippi river at Appleton, ON produced by an artificial neural network

4 Conclusion

This report has three main focal points: simulation of open channel flow using the finite volume method, numerical solution of differential equations using GPUs, and forecasting river levels from observational data. The first two were combined in the implementation of a finite volume solver using Thrust, a toolkit which allows GPUs to be used for numerical calculation. The third was realized via the implementation of a neural network trained to produce ten day forecasts of the flow rate of the Mississippi river at Appleton, Ontario.

Both endeavours featured some successes and some limitations. The finite volume solver which is that main focus of the report was implemented successfully and several scenarios were simulated to observe the effects of initial conditions and source terms on the evolution of the system. Wave propagation, flow over a downwards slope, and flow with lateral inflow were all simulated to this end. Apart from the inherent limitation that the solver can only simulate the flow in channels with vertical walls, the primary limitation of the solver relates to the boundary conditions it imposes. For simplicity, the finite volume solver imposes constant boundary conditions on Q and A. More precisely, the update procedure does not change the values at the first and last node so they remain constant at their initialized values for the duration of the simulation. This boundary condition is

not physical as in reality the cross sectional area and flow rate at either end of a segment of channel are not constant and will typically be influenced by the flow inside the channel. The simulation could be improved by finding boundary conditions which are closer to the physical reality and allowing the flow parameters at the endpoints to be updated based on those conditions. Regarding the ANN forecaster, the results in figure 5 show promise but more work is required to properly characterize the accuracy of the forecasts at various points in the forecast period and to identify cases where the forecast fails.

References

[1] D. Kraijenhoff and J. Moll, *River Flow Modelling and Forecasting.* D. Reidel Publishing Company, 1986.

[2] J. Dooge and J. O'Kane, *Deterministic Methods in Systems Hydrology.* A.A. Balkema Publishers, 2003.

[3] A. Jayawardena and T. Fernando, "River flow prediction: an artificial neural network approach," in *Regional Management of Water Resources (Proceedings of a symposium held during die Sixth IAHS Scientific Assembly at Maastricht, The Netherlands, July 2001)*, 2001.

[4] D. Zwillinger, *Handbook of Differential Equations.* Academic Press, 1992.

[5] D. Dutykh, *Mathematical Modelling of Tsunami Waves.* PhD thesis, ENS Cachan, 2007.