Representation Of Dam-Breach Geometry Using Quadtree Local Mesh Refinement

Marcus Zachariah Mcgrath
University of Mississippi

REPRESENTATION OF DAM-BREACH GEOMETRY USING QUADTREE LOCAL MESH REFINEMENT

A Thesis

presented in partial fulfillment of requirements
for the degree of Master of Science
in the Department of Engineering
The University of Mississippi

by

MARCUS Z. MCGRATH

May 2013

ABSTRACT

Advances in two-dimensional numerical modeling have allowed dam break floods to be simulated with larger domains than ever before. These types of simulations are important to meet the needs of inundation mapping, consequence analysis, and emergency planning for the large number of significant and high-hazard dams in the United States. Globally refining the mesh to the small cell sizes necessary to resolve small features such as dam breach geometry result in significant computational burden for these types of simulations. This manuscript details the research done to facilitate the implementation of quadtree local mesh refinement to represent dam breach geometry in an existing two-dimensional flood model and to test the model's results and performance with several test cases. Results using local refinement agree with the results of global refinement with a significant reduction in computational burden.

DEDICATION

This thesis is dedicated to my family, friends, mentors and advisors, and colleagues, for their love and support.

LIST OF ABBREVIATIONS AND SYMBOLS

x, y	horizontal and vertical Cartesian coordinates in physical plane
h	water depth
u, v	depth-averaged velocity components in x- and y-directions
Zb	bed elevation
Z	water surface elevation
Q	discharge per unit width
Qx, Qy	discharge per unit width in the x- and y- directions
g	acceleration due to gravity
n	Manning roughness coefficient
Δt	time step
T	simulation time
Δx	cell size
h_c	depth at cell center
Zb_c	bed elevation at cell center

$\boldsymbol{U}$	vector of conserved variables
$\boldsymbol{F}, \boldsymbol{G}$	flux vectors in x- and y- directions
$\boldsymbol{S}$	vector of source terms
S_{fx}, S_{fy}	bottom friction terms in x- and y- directions
S_L, S_R, S_*	estimated wave speeds
R	refinement level
R_{max}	maximum refinement level
Δt_R	time step for refinement level R
Δx_R	cell size for refinement level R
h_i	depth at the cell interface
Zb_i	bed elevation at the cell interface

ACKNOWLEDGEMENTS

I would like to acknowledge my advisor, Dr. Mustafa Altinakar, Director and Research Professor at the National Center for Computational Hydroscience and Engineering at the University of Mississippi, for his patience and thoughtful guidance. I also acknowledge my other two thesis committee members, Dr. Yafei Jia and Dr. Weiming Wu.

This research was funded by the Department of Homeland Security-sponsored Southeast Region Research Initiative (SERRI) at the Department of Energy's Oak Ridge National Laboratory.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. SCOPE AND MOTIVATION

Dams are manmade structures used to impound water for irrigation, consumption, energy generation, flood control, or a host of other uses. They can often be large and hold back a large amount of water; therefore, they present an inherent risk to people or infrastructure downstream in the event of a failure. Failure is an uncontrolled flow of water which can be caused by structural damage of the dam body or appurtenance by natural or unnatural means, which may be overtopping due to excessive inflow into the reservoir, failure of operation of gates or outlet structures, piping, or design or construction flaws. The United States National Inventory of Dams lists dams as low hazard, where failure or incorrect operation results in no probable loss of life and low economic or environmental losses; significant hazard, where no loss of life occurs but environmental damage and economic loss may result; and high hazard, where loss of human life is probable. As of 2011, there are a total of about 84,000 dams in the United States, of which 57,000 are low hazard, 13,000 are significant hazard, and 14,000 are high hazard. About one-fifth of significant and high-hazard dams do not have an emergency action plan although it is required by law.

Emergency management efforts involving dam failures cover several phases, including preparedness, response, and recovery. The preparedness phase occurs before the emergency, and this allows more time to consider many options and potential failure scenarios. Emergency Action Plans (EAPs) created during the preparedness phase are a requirement for high hazard dams, and a basic part of the EAP is an understanding of the area inundated by a potential flood caused by a dam failure. Data from previous dam break floods may be used as a general guide to delineate a rough estimate of the inundation zone, but simulating specific failure events with physical or numerical models gives a more accurate solution. Many sets of results for different scenarios may be required during the preparedness phase of planning for a dam failure, and results may be required much quicker during the response phase, when the dam failure event is actually happening. Therefore, the model upon which the flood inundation maps are based must be able to generate results with adequate accuracy in a timely manner.

Recent advancements in computer technology have allowed numerical models to become a useful tool for solving dam break problems and problems involving fluid flows in general. Dam break models must account for the dam failure, the outflow of water from behind the dam, and the spread of water over the terrain downstream. Due to past limited computational resources, one-dimensional models such as Hydrologic Engineering Centers River Analysis System (HEC-RAS) were developed and remain popular. These models require the generation of cross sections from the terrain and an estimation of the time-varying outflow of water through the dam breach. Non-channelized flow over flat areas is difficult to simulate because of the 1D nature of the models. In addition, 1D models assume that the flood wave arrives at the entire width of each cross section at the same time, leading to errors in arrival time calculation. For consequence analysis, the results simulated using 1D models need to be converted into a two-dimensional inundation map using painstaking and time consuming interpolation procedures on a Geographic Information System (GIS) platform.

As computing power increased, two-dimensional models became more feasible. These models improved upon some of the difficulties encountered by 1D models, but early numerical schemes had problems with mixed flow regimes and discontinuous flows. The emergence of GIS and remote sensing technologies facilitated setting up simulations and visualizing results. These GIS platforms simplify data preparation by importing bed elevation and roughness data, spatial layers of infrastructure, and by allowing the user to quickly define modeling feature such as dams, roads, levees, embankments, and source or sink areas. One such modeling software package based on a GIS platform is the Decision Support System for Water Infrastructural Security (DSS-WISE™) with a numerical model called CCHE2D-FLOOD, developed at the National Center for Computational Hydroscience at the University of Mississippi.

The research efforts described in this thesis document pertain to improving the numerical model to better predict the inundation from dam break flood simulations. The original CCHE2D-FLOOD code operates on a two-dimensional regular grid of cells covering the domain, with terrain elevations and flow variables defined at the center of these equal-size square cells. The dam and its associated breach geometry are modeled by directly imposing their elevations onto the cells in the domain. Even with current computer technology, it may not possible to solve large flood inundations in an agreeable amount of time because of limitations in cell size required by modeling small features in the domain and the associated time step imposed by the small cells. The potential inundation area may be thousands of square kilometers, and the desired cell size may be less than 10m, resulting in hundreds of millions of cells. Another challenge may be that the terrain data does not exist in the desired detail. Small features such as levees, channels, and dam breaches may not be captured adequately in the existing DEM data. To model a large flood in a reasonable amount of time, concessions may be made by increasing the cell size, thereby reducing the total number of cells to cover the given area and reducing the associated computational burden. However, since the dam breach is modeled directly by modifying the terrain elevation of the cells covering the breach area, the cells may be too large to properly model the breach and resulting outflow of water. Adding to the difficulty is the fact that dam orientation may not align with gridlines in the domain. This results in the need to use a larger breach width to provide a minimum opening in the dam through which water can flow, and this may not depict the desired breach geometry correctly.

To model the dam breach properly, one option is to simply model the entire domain with more detail using smaller cells and accept the longer time requirements for computation. Another option is to model the breach process and flow discharge with some other software and use the results as input for the inflow of water into the area just downstream of the dam. The method undertaken for this research involves a third option using local refinement- modeling the breach with greater detail simultaneously with the coarser cells in the rest of the domain. This quadtree local refinement technique involves data structures and computational algorithms for calculating a hierarchy of refined cells, where one larger cell is split into four smaller cells, each of which may be refined in turn until the desired level is achieved. This allows the breach to be modeled by changing the elevations of the small cells in the dam while keeping the cells in the rest of the domain at their regular size. This is beneficial since dam break floods can cover a large area compared to the size of the breach itself.

In order to implement the quadtree local refinement technique, the numerical solver was changed from its original first-order upwinding scheme to an HLLC scheme. Several parts of the code were rewritten to accommodate the new computational mesh, including data structures, flux solver, input and output subroutines, automatic calculation of the required dam refinement level, and automatic refinement of cells near the dam to the appropriate level. The HLLC solver was tested with classical cases, and gives good agreement with the known analytical solutions. Several test cases involving the breaching of dams with different orientations relative to the grid lines and different refinement levels were then carried out. For each orientation, a control case was simulated where all cells in the domain were the minimum size (global refinement). The time to compute the control case and the discharge hydrograph of water going through the breach were calculated. Then for each given dam orientation, the cells in the domain were coarsened while using local refinement to keep the cells near the dam breach at approximately the original small size. This method allows the breach to be represented with sufficient detail while reducing the number of cells in the rest of the domain. The computational time and breach discharge were then compared with the control case. In each case the intended breach geometry was the same, matched as closely as possible by the refinement technique. It is the aim of the research presented here to show that local refinement allows the cell size in the domain to become independent of the breach geometry size, and that the quadtree local refinement method offers improvement in computational time over global refinement while maintaining accuracy in breach discharge.

Chapter 2 of this document gives some background and context for this research. Chapter 3 details the existing CCHE2D-FLOOD model and the incorporation of the HLLC numerical scheme. Chapter 4 describes the implementation of the quadtree local refinement method into the model. Chapter 5 presents cases used to test the quadtree refinement method and their results along with the performance of the model. Conclusions from this research are drawn in Chapter 6.

II. BACKGROUND AND THEORY

Water covers the majority of the surface of the Earth and is required for life, so it is only natural that humankind has made attempts to study and control it through the ages. Ancient Chinese built dikes along rivers thousands of years ago, and Leonardo Da Vinci wrote about the study of it in the fifteenth century. We have built our knowledge and understanding of water over time to help us use it for our purposes, be they for drinking and irrigation, controlling floods, recreational activities, or to use its energy to do useful work. One of the key ways to accomplish this task is the construction of dams.

Dams are built to be safe, but things change over time. People may move into the inundation zone downstream of the dam, and new infrastructure may be built there as well. The dam's construction materials age, and its foundation can settle. Natural disasters such as earthquakes or flooding may strike the dam, or man-made hazards such as explosions or mismanagement may affect it as well. There is a need to prepare for such unknown hazards ahead of time, and to be able to plan and respond in a way that minimizes the potential consequences should the dam fail. To understand the consequences of a flood, the areas affected by the flood must be known. The affected area, or inundation zone, is estimated by modeling.

As has been very well explained in Altinakar et. al., 2009, a large number of numerical models exist for computation of dam break flows, which range from simplified envelope methods, to one-dimensional and two-dimensional models of various complexity. One-dimensional models are common in engineering practice due to the historical and perceived difficulties in setting up and running 2D models, despite the fact that dam break floods may violate the assumptions of the 1D model. Rapidly-varying flows common in dam breaks may lead to stability problems in models designed to simulate slowly-changing fluvial floods, and the restriction of using a single value of flow depth at each cross-section may give inaccurate results for the arrival time and a misleading envelope of maximum depths for the flood. Additionally, the 1D flow information at the cross sections must be interpolated back to the 2D map in a process that can be time consuming and require an experienced engineer to do properly. In contrast, 2D models directly solve for flow values at each grid cell in the domain, eliminating the need for cross-section generation and interpolation of 1D results into a 2D map. Advances in geographical information systems (GIS) have facilitated storage, retrieval, and modification of large amounts of geospatial data required by 2D models, including topography, land cover, roughness data, etc. These advances in GIS technology along with ever advancing computing speed and storage capabilities of modern hardware have made 2D flood models practical on large domains.

Flood modeling is accomplished by applying governing equations to a problem domain and a set of assumptions. For fluid flow problems, the solution is governed by the Navier-Stokes partial differential equations, named after Claude-Louis Navier and George Stokes. For open channel flow, the Saint Venant Equations, or shallow water equations, are derived from the Navier-Stokes equations by making the following assumptions:

1. Horizontal length scale is much larger than the vertical length scale.
2. Streamline curvature is small.
3. Vertical velocity of the fluid is small.

4. Vertical accelerations of the fluid are negligible, resulting in vertical pressure gradients that are hydrostatic.

In one dimension, the shallow water equations can be written as Equations (1) and (2), where A is the wetted cross section area, h the local flow depth, R_h the hydraulic radius, $Q = VA$ the discharge with V being the average velocity, q_ℓ the lateral discharge (which is a volume source or sink per unit length per unit time), g the gravitational acceleration, z_b the elevation of channel invert, and C represents the Chezy coefficient of roughness.

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = q_\ell \tag{1}$$

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}\left(\frac{Q^2}{A}\right) + gA\frac{\partial h}{\partial x} = -gA\frac{\partial z_b}{\partial x} - g\frac{Q\left|Q\right|}{C^2 R_h A} \tag{2}$$

These nonlinear, hyperbolic equations represent the conservation of mass and momentum, respectively. Analytical solutions of these equations exist only for a limited set of simplified cases, so the numerical solution must be used for problems involving irregular channels, complex boundary conditions, etc.

Numerical solution of the shallow water equations can be done using an implicit scheme, where values at all computational nodes are solved simultaneously, or using an explicit scheme. The explicit scheme is more straightforward to program, where flow values at the next time step are computed based only on known values at the current time step. However, this time step is governed by the Courant–Friedrichs-Lewy (CFL) condition for stability of the solution, and may be small.

Dam break flood models must be able to handle the various challenges inherent to the flow. The flow regimes may be mixed between subcritical, transcritical, and supercritical, and may evolve over time. There may be discontinuities in the flow involving hydraulic jumps, positive translatory waves, etc. There may be dry conditions downstream of the dam that become flooded and possibly dry out again, requiring the model to handle these cases of wetting and drying. The topography of the domain may be steep or highly irregular. This may lead to large source terms generated by the bed slopes.

An effort to address these challenges resulted in shock-capturing finite volume methods for solving the shallow water equations. Since shocks may arise in the solution even with smooth initial conditions, models must be able to handle discontinuities and mixed flow regimes. This requires the model to take into account the wave structure of the shallow water equations. Models must be able to solve smooth regions with high spatial accuracy while capturing discontinuities as sharply as possible without generating non-physical oscillations. Recent decades have seen improvements of finite volume high-resolution upwind methods which take into account the wave structure of the equations and respect the direction of propagation of perturbations while capturing discontinuities in the flow.

In order to maintain these properties, the 1D form of the shallow water equations must be re-written into the conservative form in Equation (3), where $\mathbf{U}$ is the vector of conserved variables, $\mathbf{F(U)}$ is the flux vector and $\mathbf{S(U)}$ represents forcing function consisting of source and sink terms. These terms are defined in Equation (4).

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

$$\mathbf{U} = \begin{bmatrix} A \\ Q \end{bmatrix}, \; \mathbf{F}(\mathbf{U}) = \begin{bmatrix} Q \\ \dfrac{Q^2}{A} \end{bmatrix}, \; \mathbf{S}(\mathbf{U}) = \begin{bmatrix} q_\ell \\ -gA\left(\dfrac{\partial Z}{\partial x}\right) - g\left(\dfrac{n^2\, Q|Q|}{R^{4/3} A}\right) \end{bmatrix} \tag{4}$$

The equations above are in differential form, but can be cast into integral form in order to capture discontinuities. Integrating over these equations over the solution space shown in Figure II-1 and applying Green's Theorem results in Equation (5).

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x_i}\left(F_{i+1/2} - F_{i-1/2}\right) + \Delta t\, S_i^n \tag{5}$$

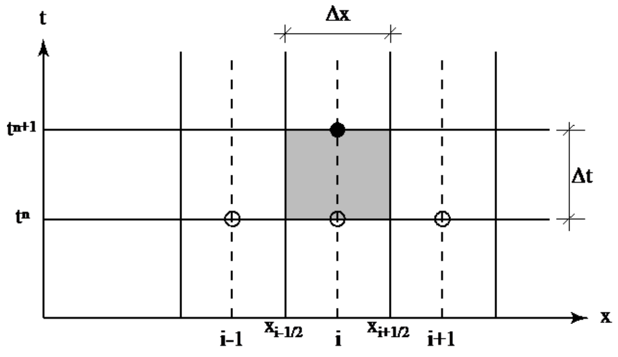


Figure II-1: Time-space discretization of solution domain for 1D shallow water equations

This form of the equations is governed by the time integrals of the numerical fluxes at the left and right interfaces of the cell, and depends only on the values of the conserved variables at the current time step.

Classical solutions to the flux terms in these equations do not take into account its wave structure. Forward Time Centered Space, Lax-Friedrichs, and Godunov Centered are examples of such schemes that can result in unstable solutions that grow without bound in certain conditions. S. K. Godunov is credited with writing solutions for intercell fluxes in what are called Godunov-type upwind methods, which take into account the wave structure of the equations to achieve a better representation of the propagation of information in the computational domain. The basic principles of Godunov-type upwind methods are shown in Figure II-2. This figure shows the x-t domain of a 1D problem domain with the conserved variable along the x axis and the time domain of the solution in the t axis. The channel is discretized with a cell size of Δx and a time step of Δt that respects the CFL condition.

a) The initial conditions are known everywhere at the beginning in (a). The goal is to advance this solution to the next time step respecting the wave structure and propagation direction of the information.
b) In (b), the average value in each cell is taken by integrating over the control volume.
c) In (c), the integrated values are assigned as piecewise-constant functions with discontinuities at the interfaces.
d) In (d) the Generalized Riemann Problem (GRP) is included at each cell interface. The GRP is likened to a dam-break problem with different depths and velocities on either side of the discontinuity. The resulting left and right-moving waves can be either shock waves or negative waves.
e) The GRP may be solved exactly or approximately. Only the conserved variables at the cell interfaces are needed to determine the Godunov-type intercell fluxes.

f) With the intercell fluxes calculated, the solution at the next time step can be evolved using Equation (5).

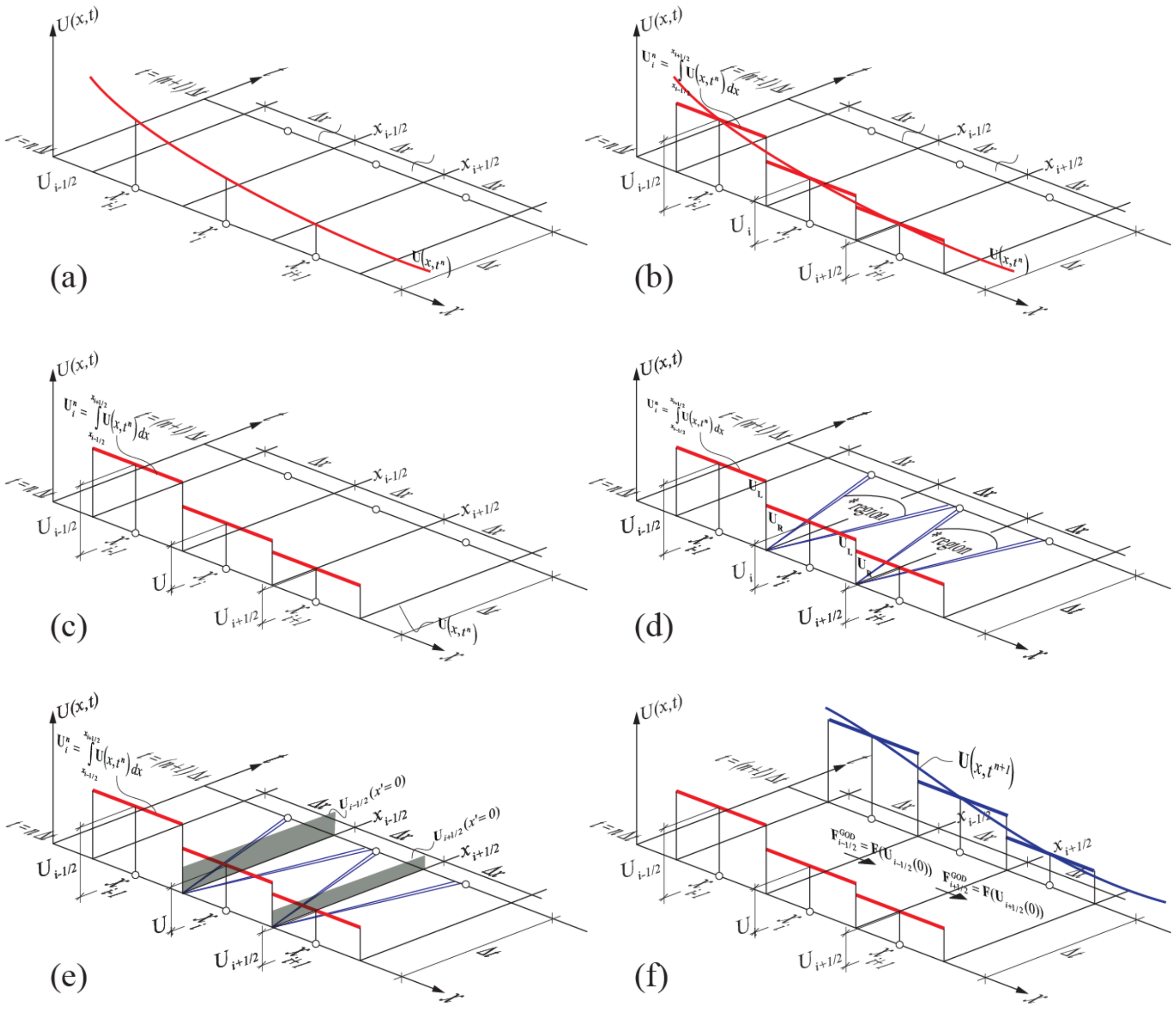


Figure II-2: Procedure for computing first-order Godunov-type upwind scheme using the Generalized Riemann Problem.

Two-dimensional solutions involve the same steps as above, but the second dimension is solved at each time step along with the first. The first-order Godunov method is a starting point for various other schemes that use different calculations for numerical flux at the cell interfaces.

In 1983 Harten, Lax, and Van Leer published the HLL approximate Riemann solver that uses an estimation of the local wave speed information to calculate the numerical flux across the cell interface. While stable and having no need for an entropy fix, this scheme ignores the contact wave, the possible jump in one of the three conserved variables in the direction perpendicular to the interface being solved. Toro modified the HLL scheme to include the contact wave, resulting in the HLLC numerical scheme that more appropriately handles 2D flow. Various authors describe ways of handling source terms and wetting and drying within the HLLC framework. This allows the scheme to be used in complex topography in mixed flow regimes which are common in dam break problems.

Ever advancing speed and storage capabilities of modern hardware has enabled simulating larger and more detailed flood models. Still, modules using explicit numerical schemes have time steps limited by the CFL condition, which is a necessary condition such that information moving at the fastest wave speed in the domain will not cross more than a single cell in a time step. Thus decreasing the cell size of the mesh used during computation by a factor of two has the effect of doubling the number of cells in both the x- and y- directions as well as halving the time step for a total of eight times the computational burden. In a regular mesh with a single global cell size, the cell size must be a compromise between having small enough cells to get an adequate level of detail for features of interest while also limiting the computational burden so the simulation finishes in a reasonable amount of time.

Recently domain decomposition techniques have been applied to flood modeling to address the need for locally-varying level of detail while retaining the benefits of using a Cartesian mesh. One such technique is the telescopic mesh, where a region of the domain is solved at a different resolution from the rest, and the two meshes are connected via edge fluxes.

Quadtree refinement is another technique which uses the idea of a quadtree, named by Finkel and Bentley (1974), to spatially relate cells of different sizes through a hierarchy of parent-child relationships. These ideas are popular in the field of image processing for quickly resampling images, and they have been applied to computational fluid dynamics for the same reasons. In this method, cells are refined according to a set of rules that maintain a strict one parent cell to four child cell relationship. Through careful consideration of refinement rules, time step, flux terms, and source terms, the quadtree local refinement technique has been successfully used to model areas with greater level of detail than the rest of the domain.

The current work replaces the numerical scheme in an existing model with HLLC and uses quadtree local refinement techniques to model dam breach geometry. This enhanced model inherits the benefits of the HLLC numerical scheme, including being able to handle rapidly-varying flows over discontinuous areas with different flow regimes over complex bed topography with wetting and drying. These properties are essential to a fast and robust flood model solving dam break flows. In addition, the model can resolve the dam breach geometry with greater level of detail using quadtree refinement without having to increase the resolution of the much larger inundation area. This combination retains the benefits of a regular Cartesian mesh while being able to decouple the size of the dam breach geometry from the global cell size.

III. DISCUSSION OF EXISTING MODEL

1. INTRODUCTION

The computational engine of DSS-WISE™ is the CCHE2D-FLOOD program. This state-of-the-art numerical model, written in modern Fortran, solves the conservative form of two-dimensional (2D) shallow water equations (SWE) over complex natural topography using a first-order, finite volume, shock-capturing scheme. It can handle mixed flow regimes, wetting and drying, and spatially-varied roughness values. It is relatively easy to set up because it uses a DEM (digital elevation model) directly as its computational domain. CCHE2D-FLOOD contains an assortment of engineering capabilities, including modeling of controlled release of water using source and sink areas, calculation of partial dam breaches, delineation of sub-grid linear features through the cut-cell immersed boundary method, and numerous forms of output including time history of flow values at specified points or discharges across arbitrary observation lines in the domain. The current work inherited this existing code base with the intent of improving the representation of partial dam breaches. However, the existing first-order upwinding numerical scheme proved to be incompatible with the design goals and had to be replaced before quadtree local refinement could be introduced. The new scheme had to fulfill the same requirements as the original scheme while reducing complexity and improving speed wherever possible. The HLLC numerical flux solver was selected to fulfill these requirements. A general description of the numerical scheme in CCHE2D-FLOOD will be discussed, followed by a description of the original first-order upwinding scheme. Finally the new HLLC scheme will be discussed, and its verification presented.

2. FIRST-ORDER UPWINDING NUMERICAL SCHEME

In general, the computational domain is a rectangular area divided into a regular grid of square cells. Each cell has four neighbors with which it shares an edge. The cell and its four immediate neighbors form the computational stencil, shown in Figure III-1, for both schemes referenced above.

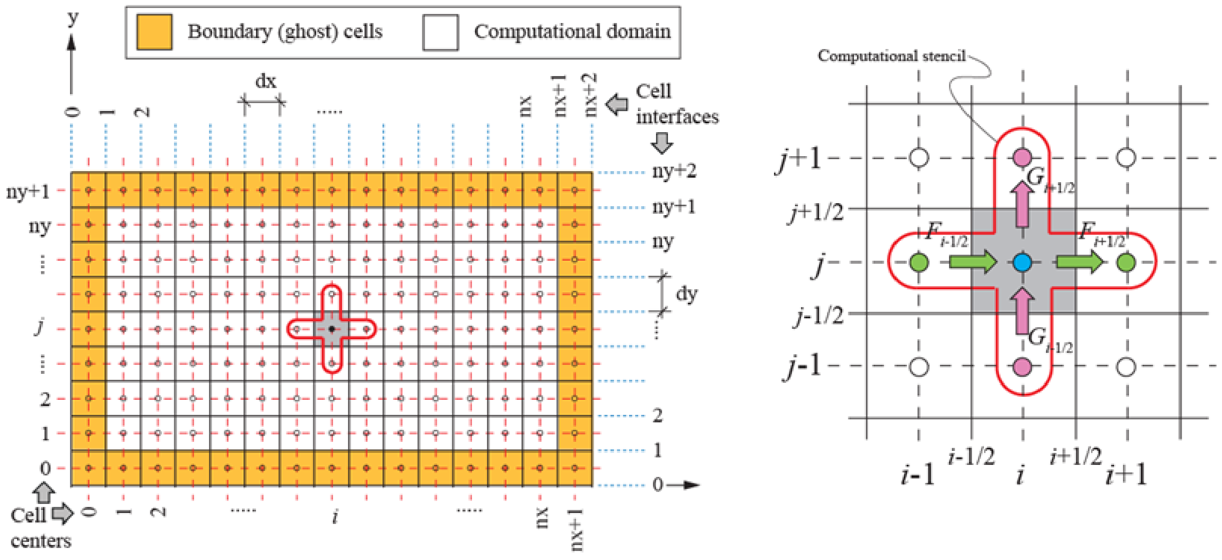


Figure III-1. Cartesian regular mesh and the computational stencil used by CCHE2D-FLOOD. White area is the computed domain and orange cells are boundary cells.

This domain represents the natural topography of the modeled area, which may be complex, as shown in Figure III-2.

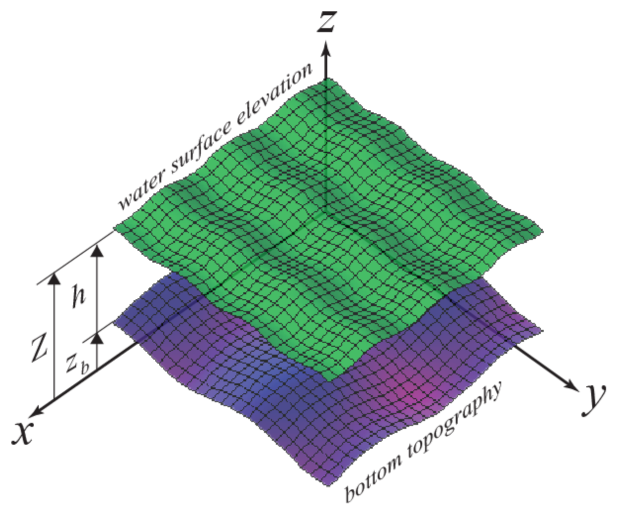


Figure III-2: Definition sketch for two-dimensional shallow water flow over complex topography

Referring to Figure III-2, the vector form of conservative 2D shallow water equations is given as:

$$\frac{\partial \boldsymbol{U}}{\partial t}+\frac{\partial \boldsymbol{F}(\boldsymbol{U})}{\partial t}+\frac{\partial \boldsymbol{G}(\boldsymbol{U})}{\partial t}=\boldsymbol{S}(\boldsymbol{U}) \tag{6}$$

In the above equation, $\boldsymbol{U}$ is the vector of conserved variables, $\boldsymbol{F}(\boldsymbol{U})$ and $\boldsymbol{G}(\boldsymbol{U})$ are the fluxes in x and y directions, respectively, and $\boldsymbol{S}(\boldsymbol{U})$ represents the vector of source and sink terms.

Finite volume discretization of Equation (1) over a regular Cartesian mesh depicted in Figure III-1, such as a DEM, provides an explicit equation for computing the values of the conserved variables at the next time step $m+1$ using the values known at the current time step m:

$$\boldsymbol{U}_{i,j}^{m+1}=\boldsymbol{U}_{i,j}^{m}-\left(\frac{\Delta t}{\Delta x}\right)\left(\boldsymbol{F}_{i+\frac{1}{2},j}-\boldsymbol{F}_{i-\frac{1}{2},j}\right)-\left(\frac{\Delta t}{\Delta y}\right)\left(\boldsymbol{G}_{i,j+\frac{1}{2}}-\boldsymbol{G}_{i,j-\frac{1}{2}}\right)+\Delta t\boldsymbol{S}_{i,j} \tag{7}$$

where Δx and Δy define cell size in x and y directions, respectively, and Δt represents the time step.

The computational stencil for Equation (7) is shown in the right image in Figure III-1. To develop an upwind scheme that takes into account the wave structure of the 2D SWE, we adopt the Godunov approach (Godunov et al., 1976) to express the fluxes through east and west intercell boundaries, $F_{i+1/2,j}$ and $F_{i-1/2,j}$, and north and south boundaries, $G_{i+1/2,j}$ and $G_{i-1/2,j}$. The Godunov approach computes the intercell fluxes by solving a Generalized Riemann Problem (GRP) at each cell interface. The exact solution of GRP is iterative and time consuming. Consequently, it is rarely used in numerical models. A large number of approximate methods have been proposed. Detailed information about GRP and its approximate solution methods can be found in LeVeque (2002) and Toro (2010).

The explicit scheme used in CCHE2D-FLOOD to solve the 2D SWE is subjected to Courant-Friedrichs-Lewy (CFL) condition for stability and convergence. The CFL condition states that during a time step, the fastest wave in the domain should not travel a distance longer than the cell size. Given a computational mesh with a specified cell size ($\Delta x \times \Delta y$), the CFL condition places an upper bound on the time step (Δt) as follows:

$$N_{CFL} = max\left[\frac{\Delta t}{\Delta x}\left(|u| + \sqrt{gh}\right), \frac{\Delta t}{\Delta y}\left(|v| + \sqrt{gh}\right)\right] < 1 \tag{8}$$

This property is used to automatically select the time step, Δt, at the beginning of each new time step. The CFL number is a necessary but not sufficient condition to ensure model stability, therefore, in order to address other concerns for stability and convergence, such as conserving positivity of depth in drying cells and large source terms, a CFL number significantly less than one is used.

The original first-order upwinding scheme defined the vectors in Equations (1) and (7) as follows:

$$\boldsymbol{U} = \begin{bmatrix} h \\ hu \\ hv \end{bmatrix}, \boldsymbol{F}(\boldsymbol{U}) = \begin{bmatrix} Q_x \\ \dfrac{Q_x^2}{h} \\ \dfrac{Q_x Q_y}{h} \end{bmatrix}, \boldsymbol{G}(\boldsymbol{U}) = \begin{bmatrix} Q_y \\ \dfrac{Q_x Q_y}{h} \\ \dfrac{Q_y^2}{h} \end{bmatrix}, \boldsymbol{S}(\boldsymbol{U}) = \begin{bmatrix} q_v \\ -gh(\partial Z/\partial x) - ghS_{fx} \\ -gh(\partial Z/\partial y) - ghS_{fy} \end{bmatrix} \tag{9}$$

In Equations (8) and (9), u and v are the local velocity components in x and y directions, Q_x is the unit discharge in the x-direction, Q_y is the unit discharge in the y-direction, h the flow depth, z_b the bed elevation, g the gravitational acceleration and q_v the net source/sink discharge (or mass per cell area per unit time) added without momentum input. The system of equations is closed by assuming that the source terms due to friction can be expressed using the Manning's equation:

$$S_{fx} = \frac{un^2\sqrt{u^2+v^2}}{h^{4/3}} \quad \text{and} \quad S_{fy} = \frac{vn^2\sqrt{u^2+v^2}}{h^{4/3}} \tag{10}$$

where n is the Manning roughness coefficient.

The original numerical flux solver utilized these formulations to determine the intercell fluxes using the upwinding direction:

$$\boldsymbol{F}_{i+\frac{1}{2},j} = \begin{cases} \begin{bmatrix} Q_{x_{i,j}} \\ \left(Q_{x_{i,j}}\right)^2 / h_{i,j} \\ Q_{x_{i,j}} Q_{y_{i,j}} / h_{i,j} \end{bmatrix}, & Q_{x_{i,j}}, Q_{x_{i+1,j}} > 0 \\ \begin{bmatrix} Q_{x_{i+1,j}} \\ \left(Q_{x_{i+1,j}}\right)^2 / h_{i+1,j} \\ Q_{x_{i+1,j}} Q_{y_{i+1,j}} / h_{i+1,j} \end{bmatrix}, & Q_{x_{i,j}}, Q_{x_{i+1,j}} < 0 \\ \begin{bmatrix} \dfrac{Q_{x_{i,j}} + Q_{x_{i+1,j}}}{2} - \dfrac{g}{2}\dfrac{\Delta t}{\Delta x}\left(Z_{i+1,j} - Z_{i,j}\right)\left(h_{i+1,j} + h_{i,j}\right) \\ \dfrac{1}{3}\left[\dfrac{\left(Q_{x_{i,j}}\right)^2}{h_{i,j}} + \dfrac{\left(Q_{x_{i+1,j}}\right)^2}{h_{i+1,j}}\right] \\ \dfrac{1}{3}\left[\dfrac{Q_{x_{i,j}} Q_{y_{i,j}}}{h_{i,j}} + \dfrac{Q_{x_{i+1,j}} Q_{y_{i+1,j}}}{h_{i+1,j}}\right] \end{bmatrix}, & otherwise \end{cases} \tag{11}$$

$$\boldsymbol{G}_{i,j+\frac{1}{2}} = \begin{cases} \begin{bmatrix} Q_{y_{i,j}} \\ Q_{x_{i,j}} Q_{y_{i,j}} / h_{i,j} \\ \left(Q_{y_{i,j}}\right)^2 / h_{i,j} \end{bmatrix}, & Q_{y_{i,j}}, Q_{y_{i,j+1}} > 0 \\ \begin{bmatrix} Q_{y_{i,j+1}} \\ Q_{x_{i,j+1}} Q_{y_{i,j+1}} / h_{i,j+1} \\ \left(Q_{y_{i,j+1}}\right)^2 / h_{i,j+1} \end{bmatrix}, & Q_{y_{i,j}}, Q_{y_{i+1,j}} < 0 \\ \begin{bmatrix} \dfrac{Q_{y_{i,j}} + Q_{y_{i+1,j}}}{2} - \dfrac{g}{2}\dfrac{\Delta t}{\Delta y}\left(Z_{i,j+1} - Z_{i,j}\right)\left(h_{i,j+1} + h_{i,j}\right) \\ \dfrac{1}{3}\left[\dfrac{Q_{x_{i,j}} Q_{y_{i,j}}}{h_{i,j}} + \dfrac{Q_{x_{i,j+1}} Q_{y_{i,j+1}}}{h_{i+1,j}}\right] \\ \dfrac{1}{3}\left[\dfrac{\left(Q_{y_{i,j}}\right)^2}{h_{i,j}} + \dfrac{\left(Q_{y_{i,j+1}}\right)^2}{h_{i,j+1}}\right] \end{bmatrix}, & otherwise \end{cases} \tag{12}$$

Essentially, if the flow normal to the edge on either side has the same direction, the values are taken from the upstream direction. If the flow on either side of an edge have different directions (contrary flow), then flux values are a combination of the upstream and downstream flow values. However, there is little mathematical justification for the latter case and the factor of one-third. Also, these equations do not allow for the influence of the downstream flow values in unidirectional sub-critical flow. Finally, the movement of part of the bed slope source term to the left hand side of the equation in the contrary flow case creates a difficult situation to handle if the edges are allowed to be split as in the case of quadtree refinement.

In this method during a time step, the fluxes on each cell edge are calculated and stored. The values of the conserved variables are then updated according to Equation (7) with the time step limited by the global limiting flow values in Equation (8). Despite a CFL value of only 0.2, an additional sweep over all cells is required to ensure the depth-positivity requirement. If a cell's depth becomes negative during a single time step, then either the time step was too large or the combination of its calculated intercell edge fluxes was too large. Therefore, a complicated method of reducing outgoing fluxes by some factor is applied to those cells with negative depths. Each edge with a mass flux that carries water out of a cell has all its fluxes scaled down by a factor. The factor is the total available mass (depth) in the cell divided by the total outgoing mass fluxes. However, this flux-reduction procedure does not guarantee an acceptable solution, since reducing the outgoing fluxes from one cell also reduces the incoming fluxes to its neighbors. This reduction strategy might induce negative depths in neighboring cells, which in turn would require adjustment of their fluxes.

This flow solver has certain features that make it undesirable for the implementation of local refinement. The first is that the updating of a cell's flow values uses upwinding, and a decision on the upwinding direction requires information about cells on both sides of a given cell. This is not a problem in cases with a uniform cell size, but decision becomes difficult when there is more than one adjacent cell on a on a given side as in the case of local refinement. In fact, with quadtree local mesh refinement, a cell can have up to 8 neighbors. This would make it very difficult to determine the upwinding direction in a robust way or to implement the flux reduction scheme mentioned above These drawbacks to the first-order upwinding numerical scheme made the implementation of quadtree local refinement impractical. These limitations could be removed and results made more accurate by replacing the numerical scheme with one that is more robust, and HLLC was introduced as a result.

3. HLLC NUMERICAL SCHEME

CCHE2D-FLOOD has been adapted to use the first order HLLC Riemann solver (Toro et al., 1992, and 1994), which is a modified version of the HLL (Harten, Lax and van Leer) Riemann solver originally proposed by Harten, Lax and van Leer (1983). The C in HLLC method stands for the contact wave (when solving for one direction, the changes in the perpendicular direction behave as contact waves). The implementation of HLLC in CCHE2D-FLOOD follows the methodology presented in Kim et al. (2007).

The vectors of conserved variables become:

$$\boldsymbol{U}=\begin{bmatrix}h\\hu\\hv\end{bmatrix}\quad \boldsymbol{F}(\boldsymbol{U})=\begin{bmatrix}hu\\huu+\frac{1}{2}gh^2\\huv\end{bmatrix}\quad \boldsymbol{G}(\boldsymbol{U})=\begin{bmatrix}hv\\hvu\\hvv+\frac{1}{2}gh^2\end{bmatrix}$$

$$\boldsymbol{S}(\boldsymbol{U})=\begin{bmatrix}0\\-ghS_{fx}-g\frac{1}{2}(h_{iL}+h_{iR})\frac{\partial z_b}{\partial x}\\-ghS_{fy}-g\frac{1}{2}(h_{iB}+h_{iT})\frac{\partial z_b}{\partial y}\end{bmatrix} \tag{13}$$

The variable meanings are the same as in Equation (9), and h_{iL}, h_{iR}, h_{iB}, and h_{iT} are the depths computed and the left, right, bottom, and top cell interfaces, respectively.

To compute the HLLC fluxes, first the maximum and minimum wave speeds and the speed of the contact wave are determined using the Equations (14) to (17):

If the left cell is dry

$$\begin{gathered}S_L=u_R-2\sqrt{gh_R}\\S_*=S_L\\S_R=u_R+\sqrt{gh_R}\end{gathered} \tag{14}$$

If the right cell is dry

$$\begin{gathered}S_L=u_L-\sqrt{gh_L}\\S_*=S_R\\S_R=u_L+2\sqrt{gh_L}\end{gathered} \tag{15}$$

Otherwise

$$\begin{gathered}S_L=\min\left(u_L-\sqrt{gh_L},\,u_*-\sqrt{gh_*}\right)\\S_*=\,u_*=\frac{u_L+u_R}{2}+\sqrt{gh_L},-\sqrt{gh_R}\\S_R=max\left(u_R+\sqrt{gh_R},\,u_*+\sqrt{gh_*}\right)\end{gathered} \tag{16}$$

where

$$h_*=\frac{1}{g}\left[\frac{1}{2}\left(\sqrt{gh_L}+\sqrt{gh_R}\right)+\frac{1}{4}(u_L-u_R)\right]^2 \tag{17}$$

Based on the wave speeds given by Equations (14) to (17), the intercell flux can be directly computed from one of the four expressions given below:

$$\boldsymbol{F} = \begin{cases} \boldsymbol{F}_L & \text{for} \quad 0 \le S_L \\ \boldsymbol{F}_L^* = \boldsymbol{F}_L + S_L(\boldsymbol{U}_L^* - \boldsymbol{U}_L) & \text{for} \quad S_L \le 0 \le S_* \\ \boldsymbol{F}_R^* = \boldsymbol{F}_R + S_R(\boldsymbol{U}_{\boldsymbol{R}}^* - \boldsymbol{U}_R) & \text{for} \quad S_* \le 0 \le S_R \\ \boldsymbol{F}_R & \text{for} \quad S_R \le 0 \end{cases} \tag{18}$$

with

$$\boldsymbol{U}_K^* = h_K \left(\frac{S_K - u_K}{S_K - S_*}\right) \begin{pmatrix} 1 \\ S_* \\ v_K \end{pmatrix} \tag{19}$$

where $K = L, R$.

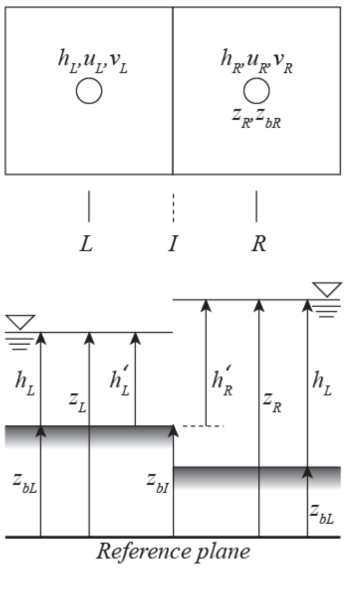


Figure III-3: Procedure for calculating the intercell flux by taking into account differences in bed elevation

As it can be seen, in this implementation the wet/dry interface is considered explicitly and the computations are done accordingly. A special procedure is implemented to take into account the bed elevation differences between the left and right cells. Consider the situation depicted in Figure III-3. Let the interface I be located between the left cell (L) and the right cell (R). The bed elevation at the interface is defined as:

$$z_{bI} = max(z_{bL}, z_{bR}) \tag{20}$$

The following temporary values of variables are used for the left (L) and right (R) cells in computing the fluxes:

$$\begin{array}{ll} \text{if } (h_L + z_{bL} - z_{bI}) > 0 & h_L^{'} = h_L + z_{bL} - z_{bI} \\ \text{otherwise} & h_L^{'} = u_L^{'} = v_L^{'} = 0 \end{array} \tag{21}$$

$$\begin{array}{ll} \text{if } (h_R + z_{bR} - z_{bI}) > 0 & h_R{'} = h_R + z_{bR} - z_{bI} \\ \text{otherwise} & h_R^{'} = u_R^{'} = v_R^{'} = 0 \end{array} \tag{22}$$

Computation of the gradients of the bed elevation, which appear in the source terms, also requires a special consideration. For the cell (*i,j*), the gradients of bed elevation in x and y directions are carried out using the following expressions:

$$\frac{\partial z_b}{\partial x} = \frac{z_{b_{down}} - z_{b_{up}}}{\Delta x} \text{ and } \frac{\partial z_b}{\partial y} = \frac{z_{b_{down}} - z_{b_{up}}}{\Delta y} \tag{23}$$

in which the up and down bed elevations are defined by

$$
\begin{aligned}
&\text{in } x \text{ direction} \quad \begin{cases} z_{b_{down}} = min\left(z_{b_{i+1/2j}}, h_{ij} + z_{b_{ij}}\right) \\ z_{b_{up}} = min\left(z_{b_{i-1/2j}}, h_{ij} + z_{b_{ij}}\right) \end{cases} \\
&\text{in } y \text{ direction} \quad \begin{cases} z_{b_{down}} = min\left(z_{b_{ij+1/2}}, h_{ij} + z_{b_{ij}}\right) \\ z_{b_{up}} = min\left(z_{b_{ij-1/2}}, h_{ij} + z_{b_{ij}}\right) \end{cases}
\end{aligned}
\tag{24}
$$

The depth part of the bed slope source term, $(h_{iL} + h_{iR})$ in the x-direction and $(h_{iB} + h_{iT})$ in the y-direction, are the sums of the depths at a cell's left and right interfaces, or bottom and top interfaces, respectively.

It is also important to note that when the computed depth in a cell becomes less than or equal to a very small value, say $\varepsilon = 10^{-9}$ m, the cell is assumed to be dry and the velocity components are set to zero. Otherwise, the velocities in the cell c can be computed from its conserved variables:

$$
u_c = \frac{hu_c}{h_c}
\qquad
v_c = \frac{hv_c}{h_c}
\tag{25}
$$

To prevent oscillations caused by the friction term, that part of the source term is linearized. Detailed below is the linearization of the friction term in the x-direction momentum equation, but the y-direction is handled in a similar fashion. Writing out the x-direction momentum equation from Equation (13) gives Equation (26) below.

$$(hu)_{ij}^{m+1} = (hu)_{ij}^{m} - \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}_{i+\frac{1}{2},j} - \boldsymbol{F}_{i-\frac{1}{2},j}\right) - \frac{\Delta t}{\Delta y}\left(\boldsymbol{G}_{i,j+\frac{1}{2}} - \boldsymbol{G}_{i,j-\frac{1}{2}}\right) + \Delta t\left(-g\left(h_{ij}^{m+1}\right)\frac{\left(u_{ij}^{m+1}\right)\left(n_{ij}\right)^2\sqrt{\left(u_{ij}^{m}\right)^2 + \left(v_{ij}^{m}\right)^2}}{\left(h_{ij}^{m+1}\right)^{\frac{4}{3}}}\right) + \Delta t\left(-g\frac{1}{2}\left(h_{i+\frac{1}{2},j} + h_{i-\frac{1}{2},j}\right)\left(\frac{z_{b_{down}} - z_{b_{up}}}{\Delta x}\right)\right) \tag{26}$$

The variable n is the manning roughness coefficient for the cell, with units of $\frac{m^{-\frac{1}{3}}}{s}$. Gathering the h_{ij}^{m+1} and u_{ij}^{m+1} terms in the friction term to the left hand side of the equation gives Equation (27):

$$(hu)_{ij}^{m+1} + (hu)_{ij}^{m+1}\Delta t\left(g\frac{\left(n_{ij}\right)^2\sqrt{\left(u_{ij}^{m}\right)^2 + \left(v_{ij}^{m}\right)^2}}{\left(h_{ij}^{m+1}\right)^{\frac{4}{3}}}\right) = (hu)_{ij}^{m} - \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}_{i+\frac{1}{2},j} - \boldsymbol{F}_{i-\frac{1}{2},j}\right) - \frac{\Delta t}{\Delta y}\left(\boldsymbol{G}_{i,j+\frac{1}{2}} - \boldsymbol{G}_{i,j-\frac{1}{2}}\right) + \Delta t\left(-g\frac{1}{2}\left(h_{i+\frac{1}{2},j} + h_{i-\frac{1}{2},j}\right)\left(\frac{z_{b_{down}} - z_{b_{up}}}{\Delta x}\right)\right) \tag{27}$$

Factoring $(hu)_{ij}^{m+1}$ gives Equation (28):

$$(hu)_{ij}^{m+1}\left(1+\Delta t\left(g\frac{\left(n_{ij}\right)^2\sqrt{\left(u_{ij}^m\right)^2+\left(v_{ij}^m\right)^2}}{\left(h_{ij}^{m+1}\right)^{\frac{4}{3}}}\right)\right)$$

$$= (hu)_{ij}^{m} - \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}_{i+\frac{1}{2},j} - \boldsymbol{F}_{i-\frac{1}{2},j}\right) - \frac{\Delta t}{\Delta y}\left(\boldsymbol{G}_{i,j+\frac{1}{2}} - \boldsymbol{G}_{i,j-\frac{1}{2}}\right) \tag{28}$$

$$+ \Delta t\left(-g\frac{1}{2}\left(h_{i+\frac{1}{2},j} + h_{i-\frac{1}{2},j}\right)\left(\frac{z_{b\,down} - z_{b\,up}}{\Delta x}\right)\right)$$

Finally, by dividing the term SF on both sides of Equation (28) gives Equation (29):

$$(hu)_{ij}^{m+1} = \left((hu)_{ij}^{m} - \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}_{i+\frac{1}{2},j} - \boldsymbol{F}_{i-\frac{1}{2},j}\right) - \frac{\Delta t}{\Delta y}\left(\boldsymbol{G}_{i,j+\frac{1}{2}} - \boldsymbol{G}_{i,j-\frac{1}{2}}\right)\right. \tag{29}$$

$$\left. + \Delta t\left(-g\frac{1}{2}\left(h_{i+\frac{1}{2},j} + h_{i-\frac{1}{2},j}\right)\left(\frac{z_{b\,down} - z_{b\,up}}{\Delta x}\right)\right)\right) * \frac{1}{SF}$$

where

$$SF = \left(1+\Delta t\left(g\frac{\left(n_{ij}\right)^2\sqrt{\left(u_{ij}^m\right)^2+\left(v_{ij}^m\right)^2}}{\left(h_{ij}^{m+1}\right)^{\frac{4}{3}}}\right)\right) \tag{30}$$

4. MODEL VERIFICATION

The verification of the HLLC numerical scheme is presented using three categories of common test cases. The first category is a simple dam break over a flat, dry, frictionless surface. The second category involves steady-state flow with different upstream and downstream boundary conditions on another flat, frictionless domain. The selection of boundary conditions in each test is designed to result in a subcritical, transcritical, or supercritical flow regime. The third category tests the so-called "C-Property" of the model: the tendency of a flat, still water surface to remain undisturbed over a complicated bed surface. If the water surface remains still after some amount of time, then it means the source terms properly balance the flux terms These test cases are presented as one-dimensional, but the second dimension for the 2D model is represented as being a single cell wide. The results of these verification tests are presented below.

A. Simple Dam Break

A one-dimensional or pseudo-two-dimensional dam break test case is commonly used to verify that the water surface elevation profile, velocity profile, and arrival time of the wave match the analytical results obtained from an exact Riemann solver. The domain, shown in Figure III-4, consists of a frictionless area 1,200m long with a dam at x= 500m, with one of two initial conditions. The first is with 10m of water upstream and dry downstream while the second is 10m of water upstream and 2m of water downstream. In both cases the water has no initial velocity, and the dam is removed instantaneously at the very beginning.

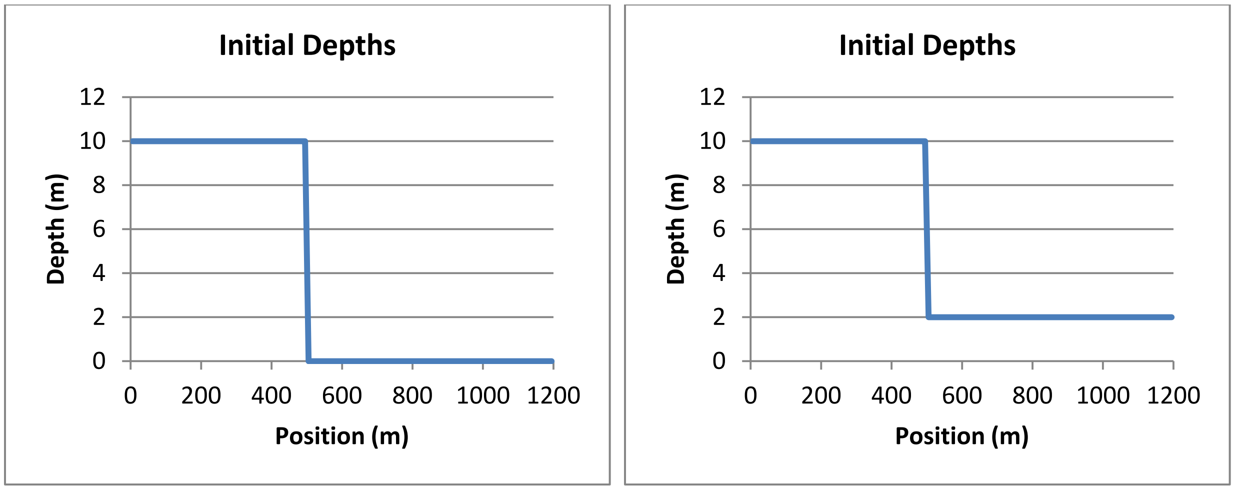


Figure III-4: Initial Depths. Left: Dry downstream. Right: 2m depth downstream

After 10 seconds, the velocity of the water with dry downstream conditions is shown in Figure III-5. Some numerical diffusion is present in the depth profile, but overall the results match rather well. The velocity profile shows that the tip of the wave has advanced farther than the exact solution to about 800m. The velocities in the leading edge of the wave remain high, but depths at these locations remain very small. This small error is likely exacerbated by the fact that there is no bed friction in this case. The velocity of water with wet downstream conditions is shown in Figure III-6. The depth profile again shows some numerical diffusion, but the velocity profile matches much better. Overall, these results show an acceptable match to the exact solution.

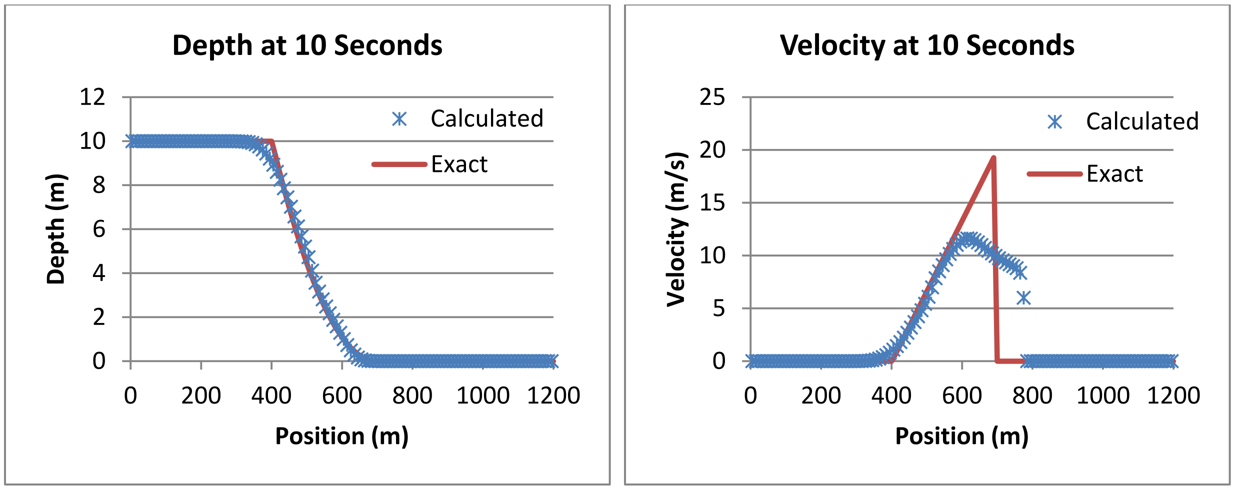


Figure III-5: Water depth and velocity 10s after dam break with dry downstream conditions

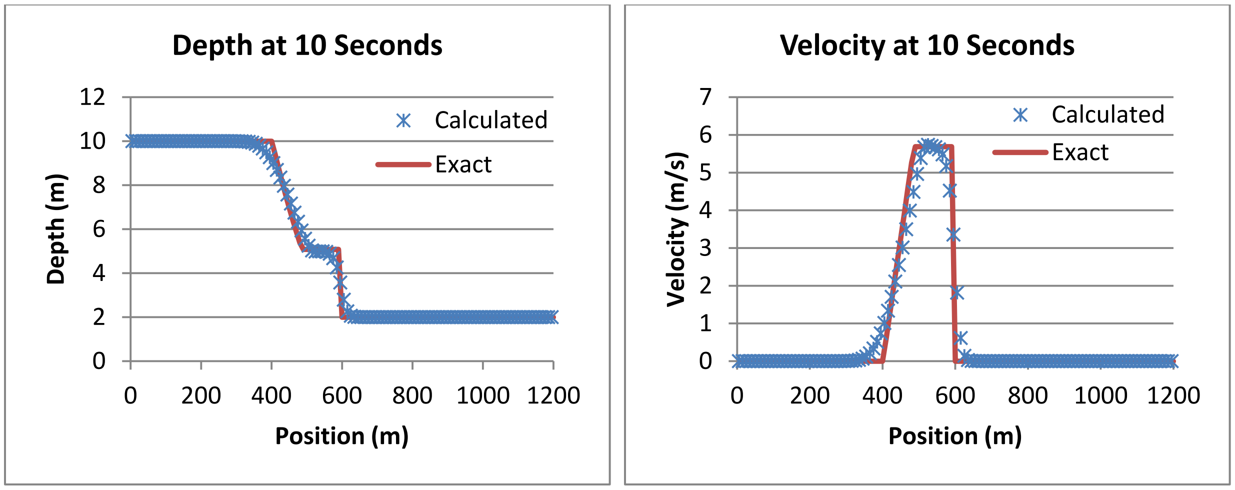


Figure III-6: Water depth and velocity 10s after dam break with wet downstream conditions

B. Flow regime tests

The second set of verification tests involve subcritical, transcritical, and supercritical flow regimes to verify the shock-capturing properties of the numerical scheme. For these tests, a frictionless domain 25m long has a 0.2m high bump centered at x= 10m. There are three types of upstream and downstream boundary conditions designed to give different steady-state solutions.

Figure III-7 shows the results of the first test case. The left or upstream side of the domain has a constant unit inflow discharge of 4.42 $m^3/s/m$ while the downstream boundary is set to a constant depth of 2.0m. This discharge causes the flow to remain subcritical throughout the domain. The bump causes a depression in the flow depths, but the cell-center discharge remains constant, save for a well-documented local deviation near the bump.

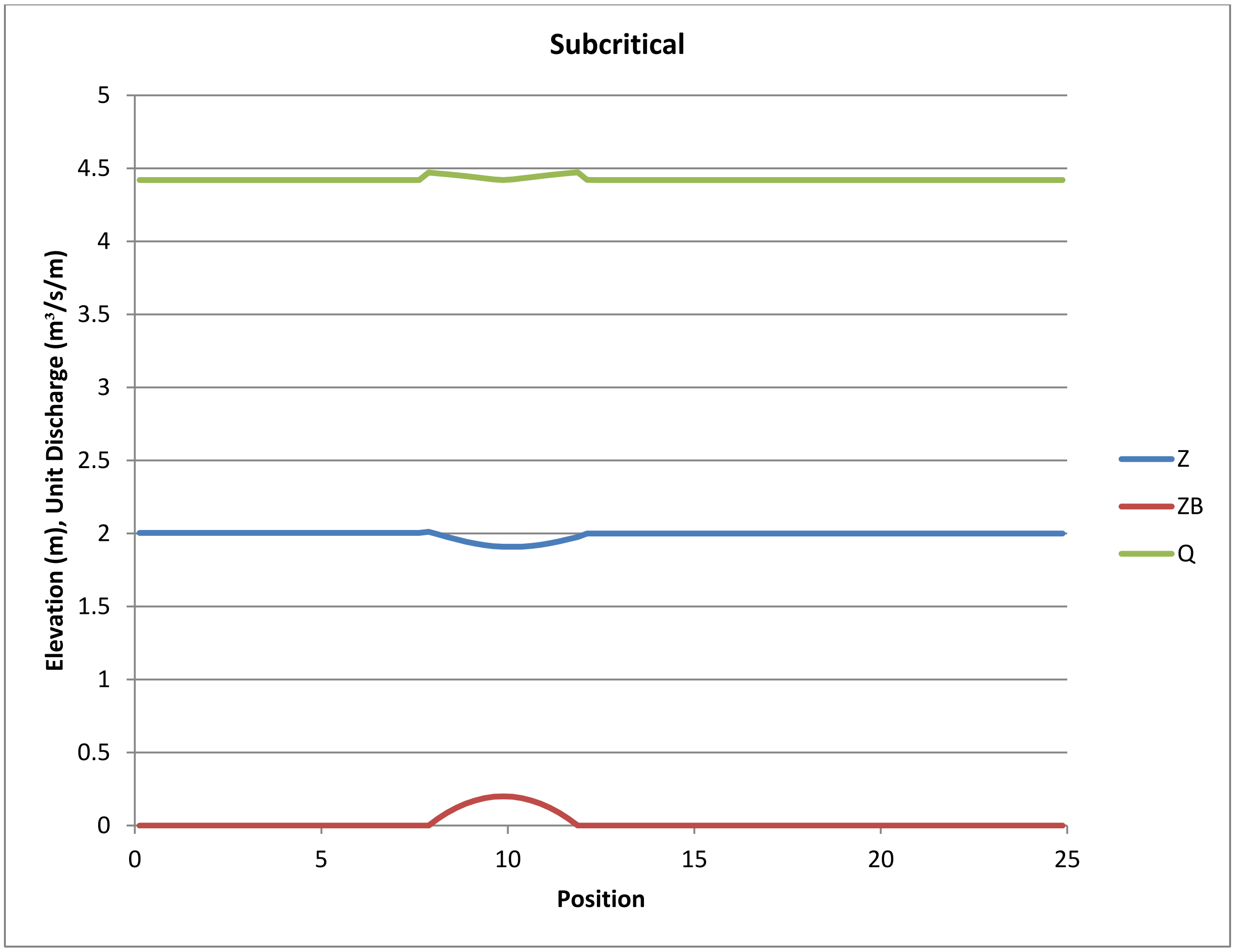


Figure III-7: Subcritical Flow.

Figure III-8 shows the second test simulation where the left or input unit discharge is 0.18 $m^3/s/m$, and the downstream depth is set to 0.33m. These conditions result in subcritical flow upstream of the bump, supercritical flow over the bump, followed by a hydraulic jump back to subcritical flow downstream of the bump. This is clearly shown in the plot of water surface in Figure III-8. Again, the cell-center discharge is constant except for the local deviation near the bump. A small bump in the water surface elevation is observed at the upstream edge of the bed bump, but this is commonly seen in other models with this test case as well.

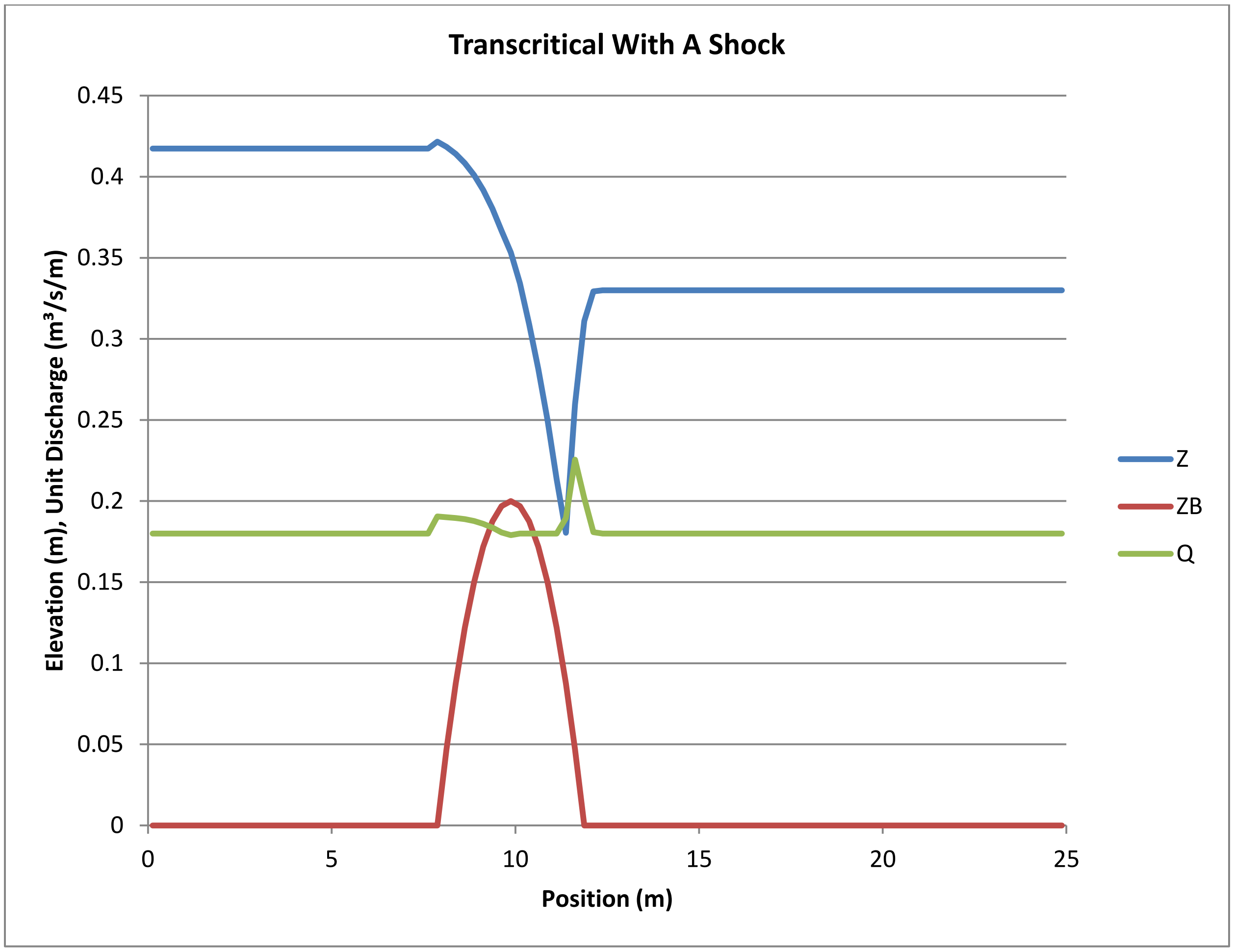


Figure III-8: Transcritical Flow with a Shock

Figure III-9 shows the third test case with an upstream unit discharge of 1.53 $m^3/s/m$, and an open boundary downstream. The water enters the domain in the subcritical regime, then passes through the critical depth over the bump, and remains supercritical until it exits. The cell-center discharge remains constant through the domain except for the local deviation near the bump. A small bump in water surface elevation is seen at the upstream edge of the bump in the bed, just as in the previous test case.

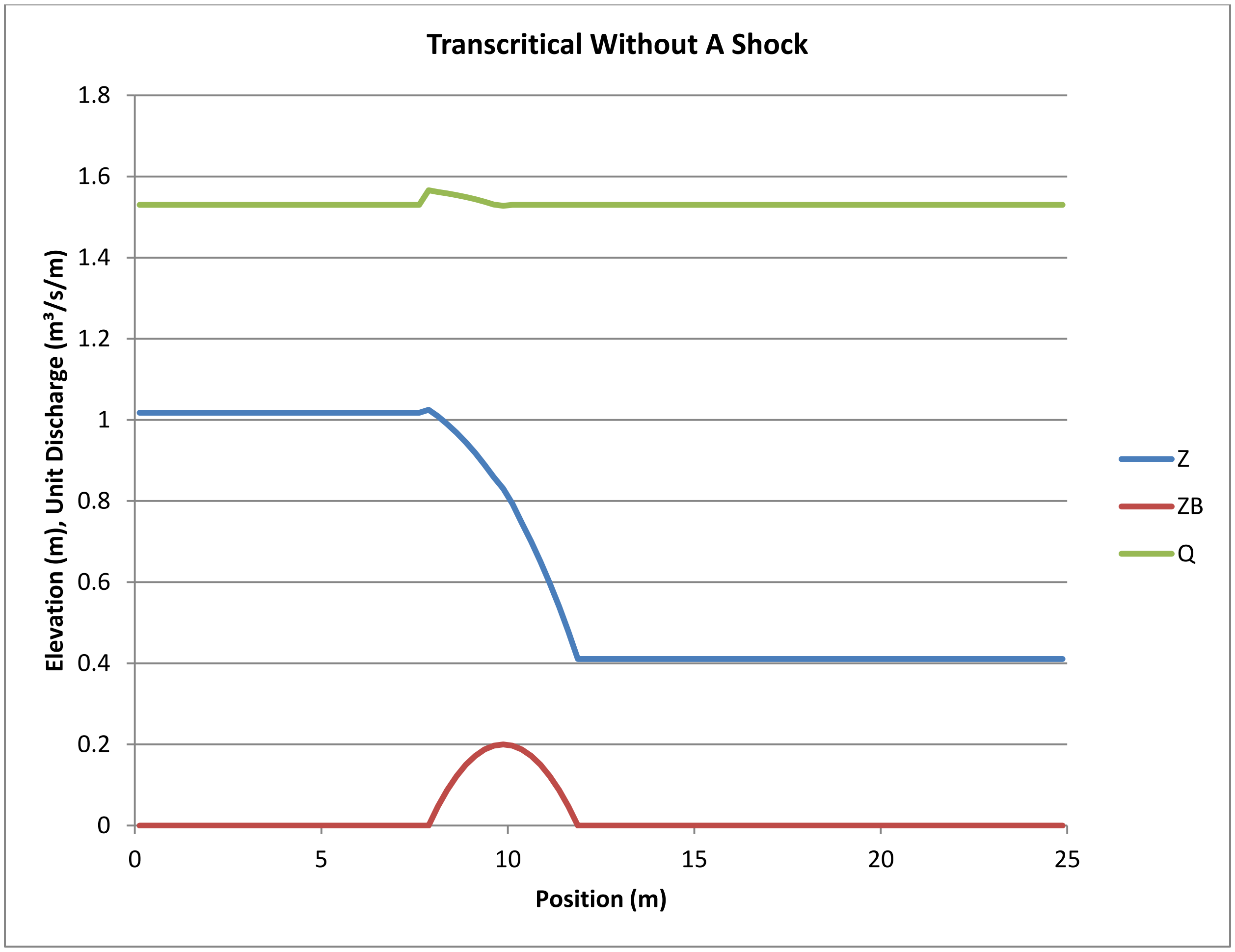


Figure III-9: Transcritical Flow Without Shock

C. Still-water Property

The final verification test shows that the source terms are properly balanced and that they generate no spurious oscillations in a flat water surface with sloping bed elevations. Figure III-10 shows a domain 50m wide with a complicated bed. The water depths over the domain were set such that when added to the bed elevation, the water surface elevation was everywhere equal to 10m. The simulation was run for 100 seconds with these water depths and no initial velocity. At the end of 100 seconds, the water surface remained undisturbed with no spurious velocities, thus showing that the source terms are properly balanced.

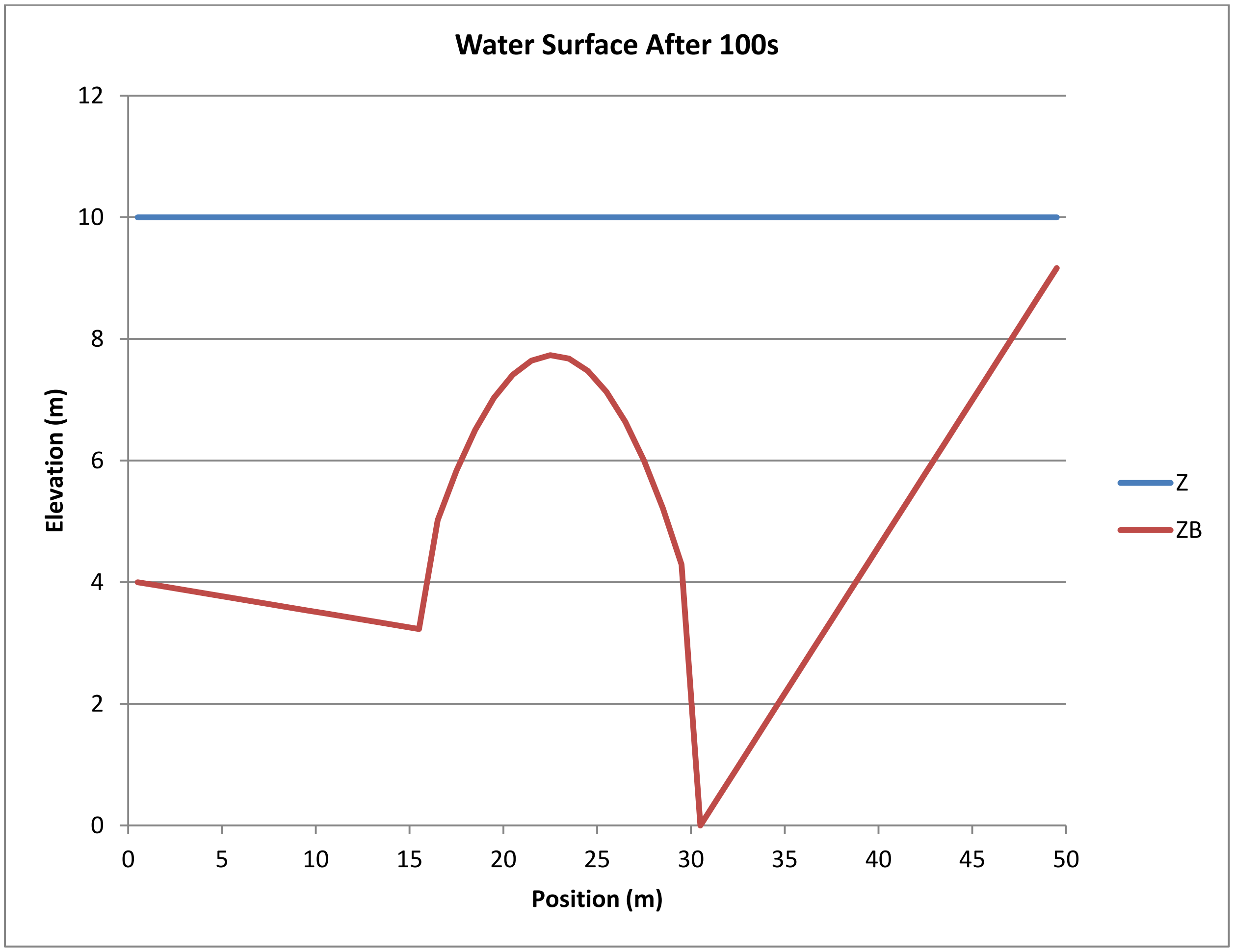


Figure III-10: Flat water surface with complicated bed.

5. DISCUSSION

The existing numerical flood model, CCHE2D-FLOOD, represents a collection of tools and procedures aimed at solving dam break and other flooding simulations. It represents a body of engineering knowledge and effort that allows flood inundation maps, discharge data, and other crucial information to be calculated quickly and with minimal setup required. It is for this reason that the current work was done within the existing framework utilizing its useful features. The enhancement of representing small features such as dam breach geometry using quadtree refinement required a superior numerical flux solver to be implemented. The HLLC numerical scheme was selected for this purpose and integrated into the existing model to replace the first-order upwinding scheme. This new scheme was verified to give correct results in a variety of different flow regimes and conditions. With a modern and capable numerical solver, the program is now ready to receive the quadtree local refinement feature.

IV. IMPLEMENTATION OF QUADTREE LOCAL REFINEEMNT

1. OVERVIEW

The basic method of computation in CCHE2D-FLOOD consists of a regular array of square cells that are all the same size. The flow values of water depth, x- and y-velocity, and the values of bed elevation, arrival time, and maximum depth record are stored at each cell center. As shown in Figure IV-1, computed values are updated by first sweeping over each of the edges shared between cells and calculating fluxes, then sweeping over each cell and using the edge flux values to update the flow values. The edges are independent, so as long as all the edges are calculated before updating the cell center values, they can be calculated in any order. This concept still applies in the way the quadtree method is implemented, only instead of sweeping over a two-dimensional array, the computational cells of various sizes are stored in a linked list. method allows different sized cells to be stored in separate lists while allowing parent cell and child cells to relate to each other using pointers. A diagram of this is shown in Figure IV-2. The "base-level" or "C1" cells in the quadtree mesh are the same size as those in the regular two-dimensional arrays. Once the sufficient and valid quadtree mesh is created, it is kept static.

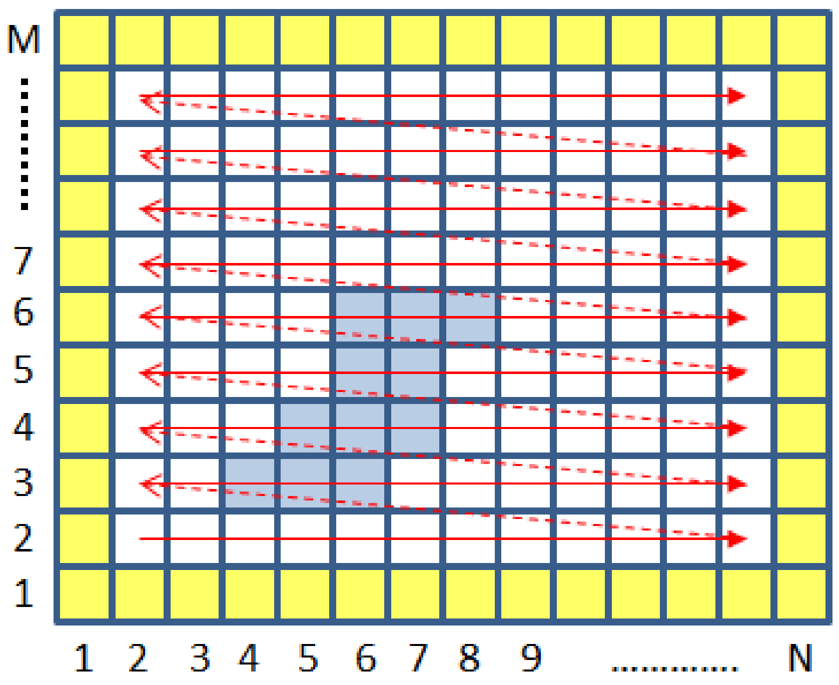


Figure IV-1: Basic computational sweep pattern. White cells are dry. Blue cells are wet. Yellow cells are ghost cells with which to calculate boundary conditions.

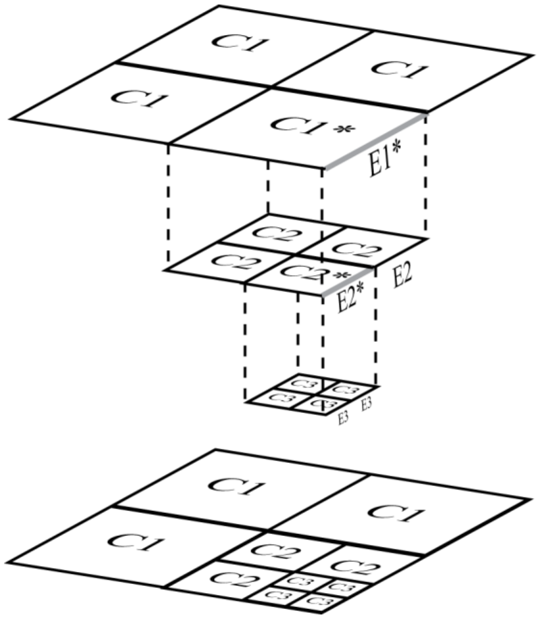


Figure IV-2: Hierarchy of cells in the quadtree data structure. The largest or coarsest cells are level C1. Higher levels of refinement are denoted by higher numbers. The flattened bottom image shows the various refined cells stacked together.

2. MESH CREATION

One of the critical aspects of the current implementation of the quadtree method is defining which areas need to be refined and by how much. The only limits to refinement are practical ones, where it would no longer be beneficial to refine an area because of diminishing returns in accuracy or in terms of added computational burden. To this effect, an upper limit on refinement level is set at 8, meaning that a base-level cell can turn into a maximum of 16384 cells at the maximum level. To put it a different way, a 100 meter cell refined to this level would in effect become a group 128 cell-lengths on an edge, each tiny cell being only 0.78125 meters wide. For most simulations, however, the refinement level may be kept much lower in order to minimize computational burden.

In the current work, only cells in the footprint of a dam are refined. However, other features requiring locally-small cell sizes could be refined using a similar method.

Each dam's maximum level of refinement is calculated from its breach geometry. There can be any number of breach profiles defining the progression of the breach, and each one corresponds to the pattern of elevations along the dam's length at a snapshot in time. Since the progression is assumed to progress linearly in time between adjacent profiles, only the user-given profiles need to be considered. To be able to accurately model the geometry of each profile, each one is analyzed to find the width of its smallest feature. Two or more successive data points that are not at the level of the crest are considered to be a feature of the profile. The width of the smallest feature for a dam's breach geometry is then taken as the limiting size. The refinement level is calculated using the formula below, requiring the smallest cell size to be able to fit 10 of their widths inside the smallest feature.

$$R_{dam} = CEILING(\frac{MAX\left(0, LOG\left(\frac{DX}{\frac{size_{feature}}{10}}\right)\right)}{LOG(2)} + 1) \tag{31}$$

Using the numbers from before, where DX= 100m and $size_{feature} = 10m$, the feature to be modeled with 10 cells, the required refinement level for the dam is calculated to be:

$$R_{dam} = CEILING\left(\frac{MAX\left(0, LOG\left(\frac{100}{\frac{10}{10}}\right)\right)}{LOG(2)} + 1\right) = CEILING\left(\frac{2}{LOG(2)} + 1\right)$$

$$= CEILING(7.64) = 8$$

That is, a refinement level of 8 is required to turn 100m cells into a size small enough to resolve a 1m feature, the same result as before.

Once the refinement level for each dam is computed, it is time to identify the actual cells that need to be refined. For each dam, its rectangular profile is projected onto the computational grid, as in Figure IV-3. Each base-level cell whose center is inside this rectangle is refined. The process of is repeated recursively for each of the new refined cells, which has already been done in the left image of the figure. It is possible that a base-level cell does not have its center inside the dam's rectangle, but has a neighbor that is inside, as in the orange cell in the left image of the figure. These cells are tested to see if any of their would-be refined cells would lie inside the dam if it were to be refined to the maximum level, as in the middle image. If so, then that base-level cell is refined, as in the right image in the figure, and the process is repeated recursively for its children.

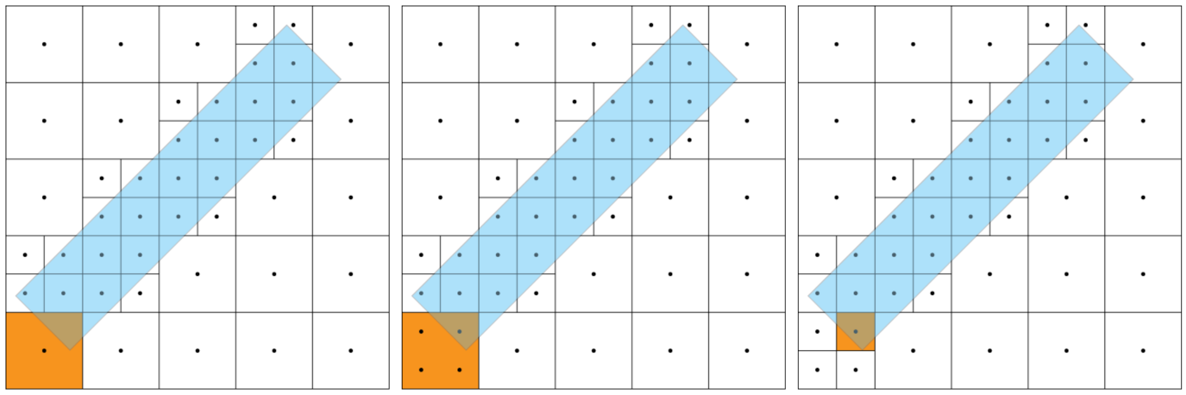


Figure IV-3: Refinement of cell adjacent to dam. Left: Orange cell's center is outside the blue dam, but an adjacent cell is inside the dam. Middle: Marked centers of cells if the orange cell were to be refined. Right: Top right child cell would be inside the dam, so the cell is refined.

When refining cells, a basic rule must be followed to maintain a valid quadtree mesh. The rule requires that adjacent cells differ by no more than one level of refinement. Put another way, each cell have at most two neighbors on a given side. If refining a cell causes one of its neighbors to violate this rule, then that neighbor must also be refined. Since the neighboring cell was refined, the rule must be enforced for that cell's neighbors as well, recursively, until the entire mesh is once again valid. This process is shown in Figure IV-4. The top image shows the orange cell selected for refinement. The second image shows the two new right-most child cells also being refined. After this step, in row 3, the cell on the right is still at the base level, and now has four neighbors to its left. Since this cell violates the quadtree refinement rule, it too must be refined. The final image shows a valid quadtree mesh.

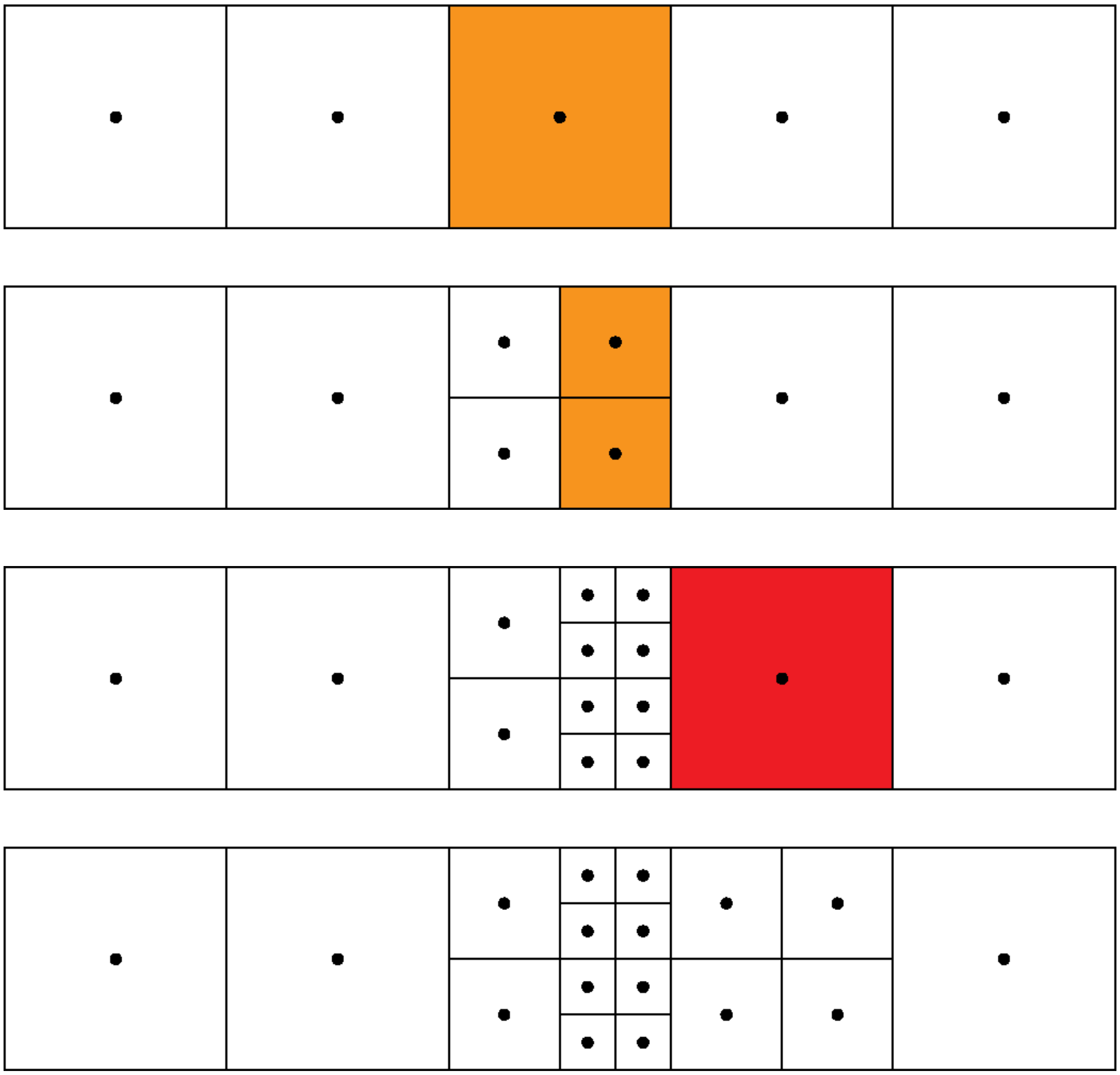


Figure IV-4: Quadtree refinement rule. Row 1: Orange cell is refined. Row 2: Orange cells are refined. Row 3: Red cell violates quadtree rule. Row 4: Final mesh.

However, this basic rule does not guarantee against rather sudden transitions between levels of refinement. Figure IV-5 shows that the cells highlighted in red in the top image have a base-level cell on their left and cells with level two refinement on their right. To avoid these sharp changes in cell size, an additional rule is imposed when refining cells: The maximum and minimum level of refinement amongst all neighbors of a given cell must be less than or equal to one. The red cells in the figure below have a neighbor to the left with level zero and cells to their right with level two, or a difference of two, thus violating the rule. The solution requires that the adjacent cell(s) with the lower refinement level must also be refined. This process is also repeated recursively until all cells in the quadtree mesh no longer violate the basic quadtree rule or the adjacency rule. The bottom image in the figure below shows the final mesh.

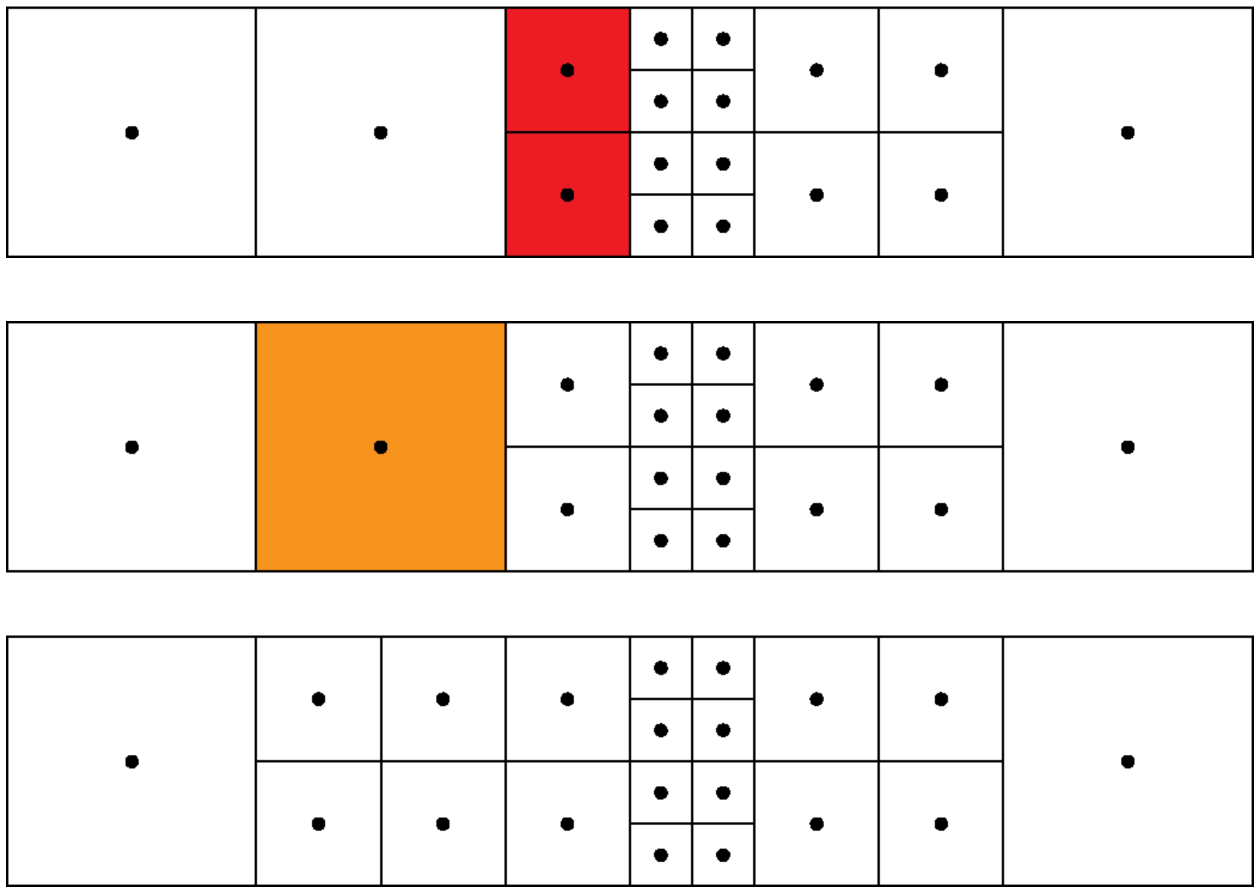


Figure IV-5: Quadtree adjacency rule. Row 1: Red cells violate the additional adjacency rule. Row 2: Orange cell must also be refined. Row 3: Final mesh.

3. DATA STRUCTURES

Each base-level cell is represented in the regular mesh by its position and flow information, including bed elevation, water depth, water surface elevation, and u- and v-velocities. These data are stored in two-dimensional arrays. The quadtree data structure consists of a one-dimensional linked-list, with each list element containing the position and flow information along with connectivity information. The connectivity information links each cell to its parent if it has one, to its four children, if it has them, and to its four edges of its same level. The edges are stored in a similar one-dimensional linked list containing position information, flux values, and connectivity information. The connectivity information for edges includes its parent edge if it has one, its two child edges if it has them, and the two cells of its same level on either side. Cells and edges are stored in a separate list for each refinement level to form a group. Figure IV-6 shows the hierarchy of cells descending from the cell marked C1* in Figure IV-2. Cells which have child cells are marked with a * symbol in the image and are called stem cells. Cells that have no child cells are called leaf cells.

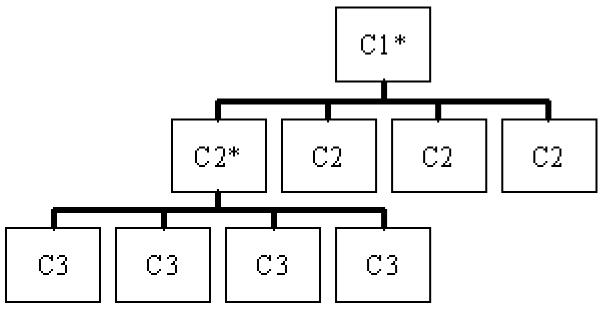


Figure IV-6: Hierarchy of cells.

Like cells, the edges have a hierarchy shown in Figure IV-7, with E1* corresponding to the same E1* in Figure IV-2. Similarly to cells, edges with child edges are called stem edges and those without are called leaf edges.

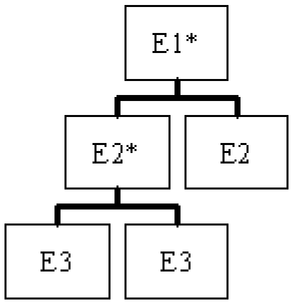


Figure IV-7: Hierarchy of edges.

4. MESH CONNECTIVITY

For computations using the quadtree local refinement method, the global 2D arrays of values are kept from the basic method. The local refinement mesh is an additional set of data structures that allows the differing-resolution cells to be calculated in succession. The quadtree mesh overlaps the basic mesh, as in Figure IV-8, where left and right images show the basic mesh on the left and the same area with the quadtree mesh superimposed on the right. The red and green cells in the left image are not computed in the basic mesh since their values are overridden by the cells in the quadtree mesh. Base-level cells in the quadtree mesh along the periphery of the refined area, highlighted in red in Figure IV-8, are duplicates of the corresponding cells in the basic mesh. These duplicates are only calculated in the quadtree mesh, since one of their edges has been refined. After calculation, their flow values are copied back to the basic mesh to facilitate communication between the two meshes.

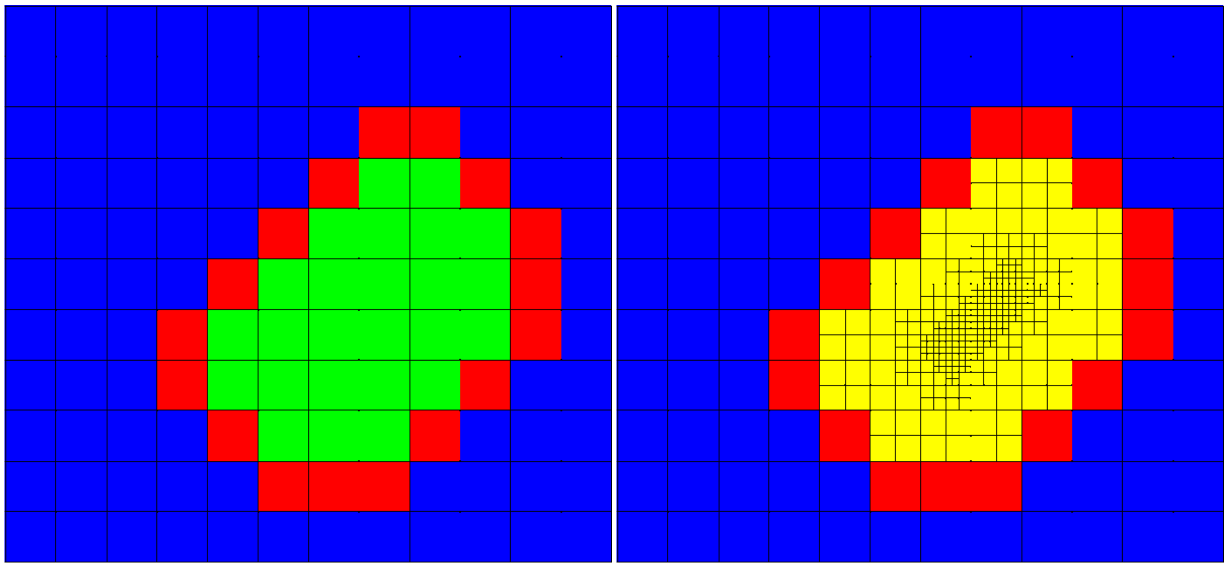


Figure IV-8: Connection between basic mesh and quadtree meshes. The left and right images show the same area. Left: Basic mesh. Red cells are shared with the quadtree mesh, and neither red nor green cells are computed. Right: Quadtree mesh. Red cells are shared with the basic mesh. Yellow cells have been refined. Both yellow and red cells are computed in this mesh.

5. NUMERICAL SCHEME MODIFICATIONS

At each time step, for each level of refinement, edges in the quadtree mesh are computed using the HLLC flux solver only if they are leaf edges. Similarly, cells are only computed from their edge fluxes if they are leaf edges. If a leaf cell is adjacent to a larger cell on one edge, then there is no problem. That cell will only see a single edge in that direction. However, if a cell is adjacent to two smaller neighbor cells or edges, then the situation is more complicated. Figure IV-9 shows the most complicated valid quadtree cell possible.

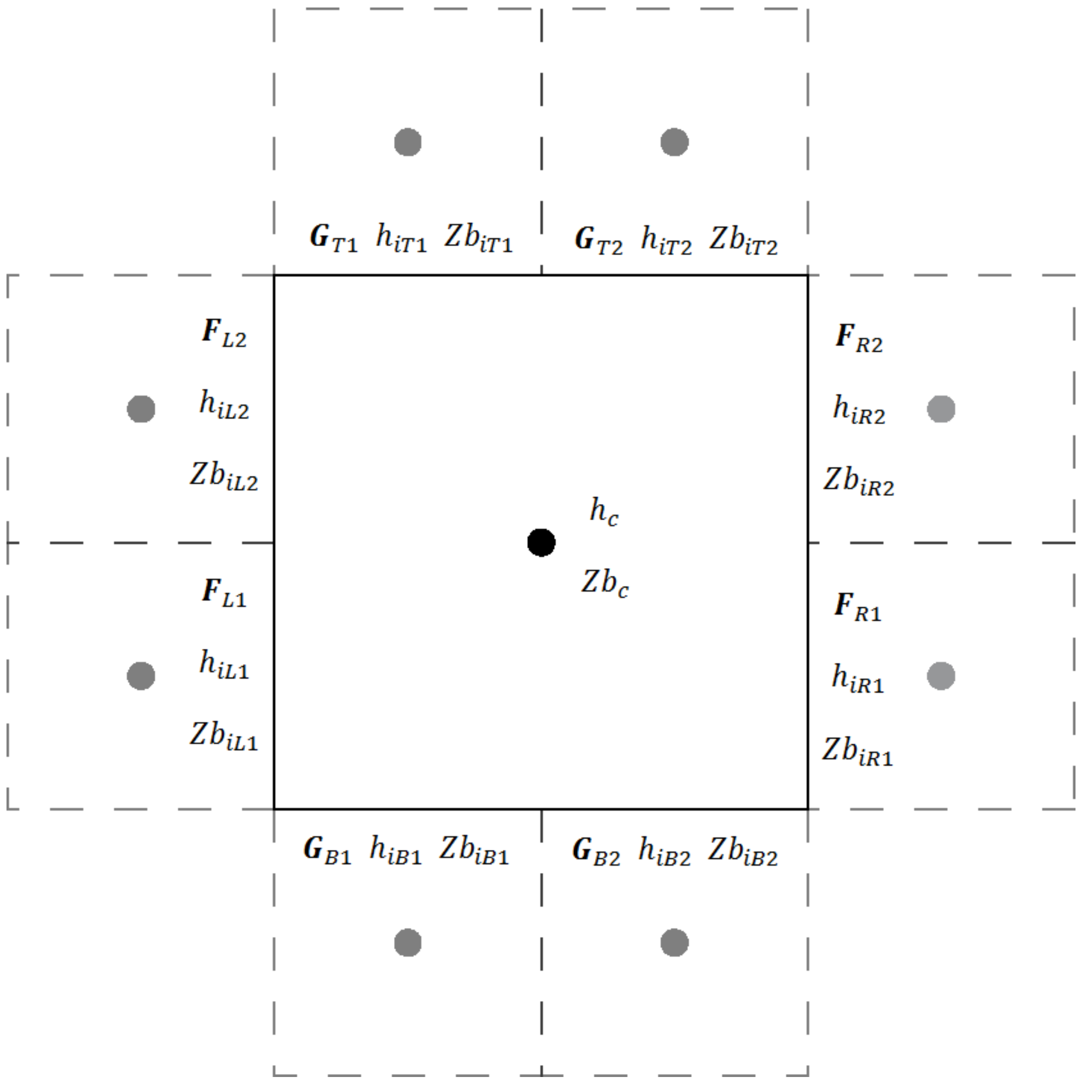


Figure IV-9: Locations of edge variables for computing quadtree cell. The bold F and G terms are vectors of fluxes with three components each.

Referring to Figure IV-9 and the previous chapters, the HLLC numerical scheme is modified as follows. If the edge in the given direction has the same level of refinement as the current cell, the flux in that direction is taken directly from the edge. If the edge in the given direction has a higher level of refinement, the flux is the sum of the two child edge fluxes, weighted by their half-size:

$$\boldsymbol{F}_L = \frac{1}{2}(\boldsymbol{F}_{L1} + \boldsymbol{F}_{L2})$$

$$\boldsymbol{F}_R = \frac{1}{2}(\boldsymbol{F}_{R1} + \boldsymbol{F}_{R2})$$

$$\boldsymbol{G}_T = \frac{1}{2}(\boldsymbol{G}_{T1} + \boldsymbol{G}_{T2}) \tag{32}$$

$$\boldsymbol{G}_B = \frac{1}{2}(\boldsymbol{G}_{B1} + \boldsymbol{G}_{B2})$$

Then Equation (2) in Chapter III is modified as such:

$$\boldsymbol{U}^{m+1} = \boldsymbol{U}^m - \frac{\Delta t_R}{\Delta x_R}(\boldsymbol{F}_R - \boldsymbol{F}_L) - \frac{\Delta t_R}{\Delta y_R}(\boldsymbol{G}_T - \boldsymbol{G}_B) + \Delta t_R \boldsymbol{S} \tag{33}$$

Where Δt_R is the time step for the given cell's level of refinement and $\Delta x_R = \Delta y_R$ is the cell size for the cell of the given refinement level.

Since the two neighboring cells on a given side may be any combination of wet and dry, the source terms must be handled in a different way than they are in the basic grid HLLC scheme. Here, the bed slope source term contribution from each neighboring cell is taken independently by way of the edges. The source term contribution from each of the neighboring cells to this one is calculated and stored during calculation of the edge fluxes, which is why these terms appear along with the fluxes in Figure IV-9.

The bed slope source term is calculated by summing the individual contributions from each of the neighboring edges. For example, for the bed slope source term in the x-direction:

Bottom-left quarter of the cell:

$$-\frac{1}{4}g\frac{1}{2}(h_{iL1}+h_c)\frac{Zb_c - Zb_{iL1}}{\frac{1}{2}\Delta x_c}$$

Top- left quarter of the cell:

$$-\frac{1}{4}g\frac{1}{2}(h_{iL2}+h_c)\frac{Zb_c - Zb_{iL2}}{\frac{1}{2}\Delta x_c}$$

Bottom-right quarter of the cell:

$$-\frac{1}{4}g\frac{1}{2}(h_{iR1}+h_c)\frac{Zb_{iR1} - Zb_c}{\frac{1}{2}\Delta x_c}$$

Top-right quarter of the cell:

$$-\frac{1}{4}g\frac{1}{2}(h_{iR2}+h_c)\frac{Zb_{iR2} - Zb_c}{\frac{1}{2}\Delta x_c}$$

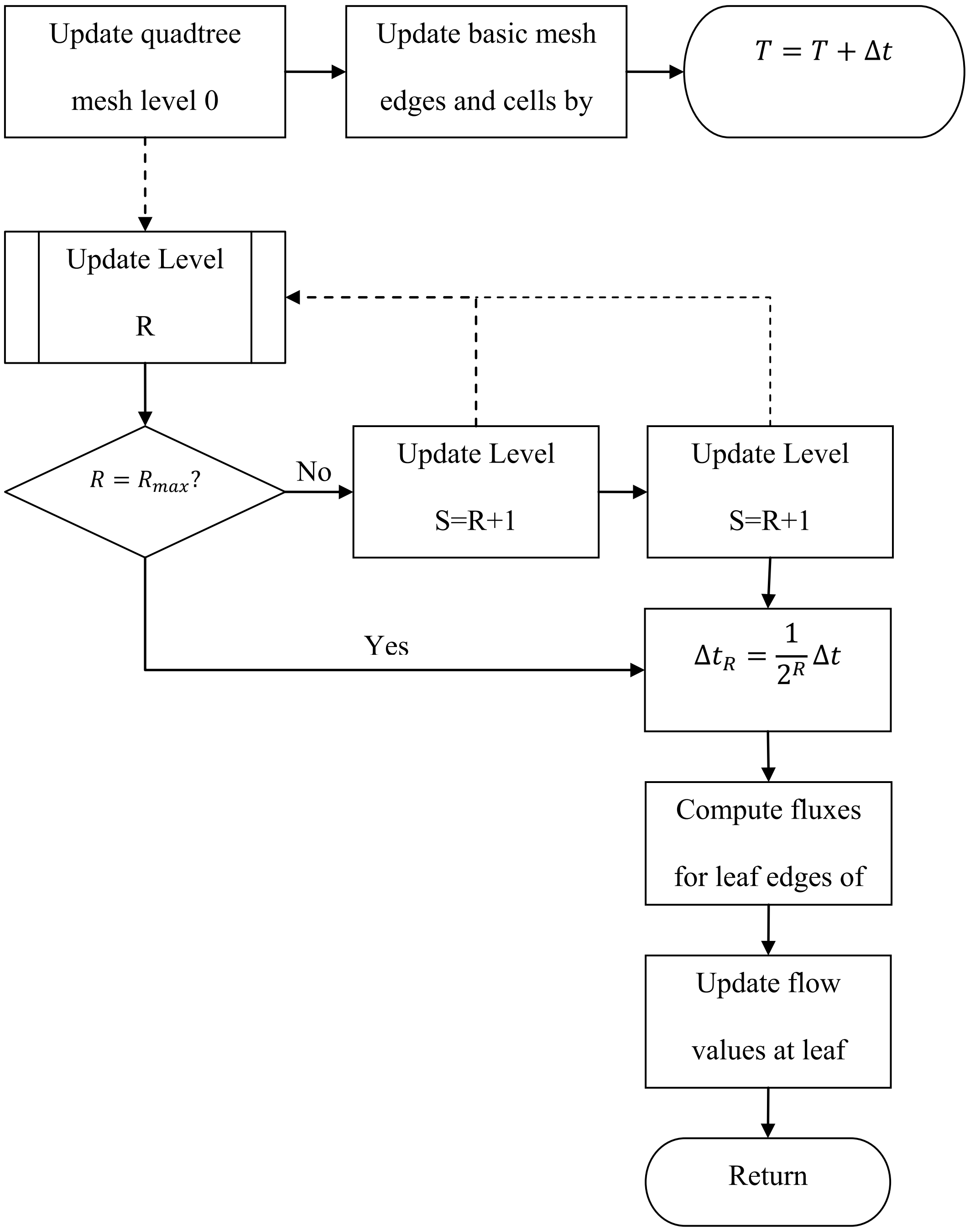
Grouping terms and rearranging yields the final form of the bed slope source term in the x-direction:

$$\begin{aligned}-\frac{1}{4}\frac{g}{\Delta x_R}[h_{iL1}(Zb_c - Zb_{iL1}) + h_{iL2}(Zb_c - Zb_{iL2}) + h_{iR1}(Zb_{iR1} - Zb_c) \\ + h_{iR2}(Zb_{iR2} - Zb_c) + h_c(Zb_{iR1} + Zb_{iR2} - Zb_{iL1} - Zb_{iL2})]\end{aligned} \tag{34}$$

The bed slope term in the y-direction is derived in a similar manner. This way of handling the bed slope source terms maintains the "C-property" or balance of the flux and source terms in still water. In the case where all bed elevations are flat, then the bed slope source term above reduces to zero, as expected. The bed roughness source term is calculated the same linearized fashion as the basic mesh.

6. COMPUTATIONAL ALGORITHM

Refined cells and edges are in lists according to their level of refinement, from R_0in the base-level, unrefined cells, to R_1, cells refined one level, and so on, up to R_{max}, the smallest cells with the highest level of refinement. The global time step is calculated from the maximum depth and the maximum velocity in all cells in both the basic and the quadtree meshes using the usual CFL condition in Equation (3) in Chapter III. This time step is divided in half for computing each level of refinement above the base-level R_0. Referring to the flowchart in Figure IV-10, the order of computations are as follows. First, the fluxes on the edges with the highest level of refinement $R_{max}$are calculated. These edge fluxes are used to update the cell-center values of the cells with that level of refinement. This first iteration brings those cells temporally half of the way to the next-lower level, R_{max-1}. Repeating this process a second time brings them temporally equal to the cells inR_{max-1}. Then the computation proceeds to compute one iteration of R_{max-1}, and so on, moving to the next-lower level once the current level has been computed twice. During each step, if a cell has children, then its values are not computed from its edge fluxes. Flow values for each of the 4 children are averaged to give the value in the parent cell. This reduces redundant calculation since the edges of a parent cell have already been computed by its children.



Update quadtree mesh level 0
Update basic mesh edges and cells by
$T = T + \Delta t$
Update Level R
$R = R_{max}?$
No
Update Level S=R+1
Update Level S=R+1
Yes
$\Delta t_R = \frac{1}{2^R}\Delta t$
Compute fluxes for leaf edges of
Update flow values at leaf
Return

Figure IV-10: Order of calculations

Once all the cells in the quadtree mesh have been computed, the flow values in the base-level cells are copied to their corresponding cells in the basic mesh, and the edges and cells in the basic mesh are computed as normal. Once both meshes have been updated by one global time step, the iteration is complete, and the process starts over until the end time is reached.

7. MEASURING DISCHARGE WITH OBSERVATION LINE

One useful feature in the basic CCHE2D-FLOOD model is the ability to query discharge across a given line segment. This feature has been extended to work in both the basic and quadtree meshes. Figure IV-11 shows an example observation line in green. The observation line is represented by a series of equally-spaced points, referring to the colored circles in the figure, and each sample point has an associated segment length, shown in red. Whether in the basic 2D mesh or the quadtree mesh, each point falls into a single cell, highlighted in orange. This cell determines the size of the interpolation cell, shown in blue. This interpolation cell's corners will touch four cells, shown in orange and yellow. The values of X- and Y- components of discharge for these four cells are used with bilinear interpolation to get the values at the sample point. These values represent a vector of discharge at the sample point. This vector is projected onto the observation line, and component normal to the line is multiplied by the length of the segment associated with that sample point. The positive and negative values of normal discharge are summed separately among all the sample points to give the total discharge across the observation line.

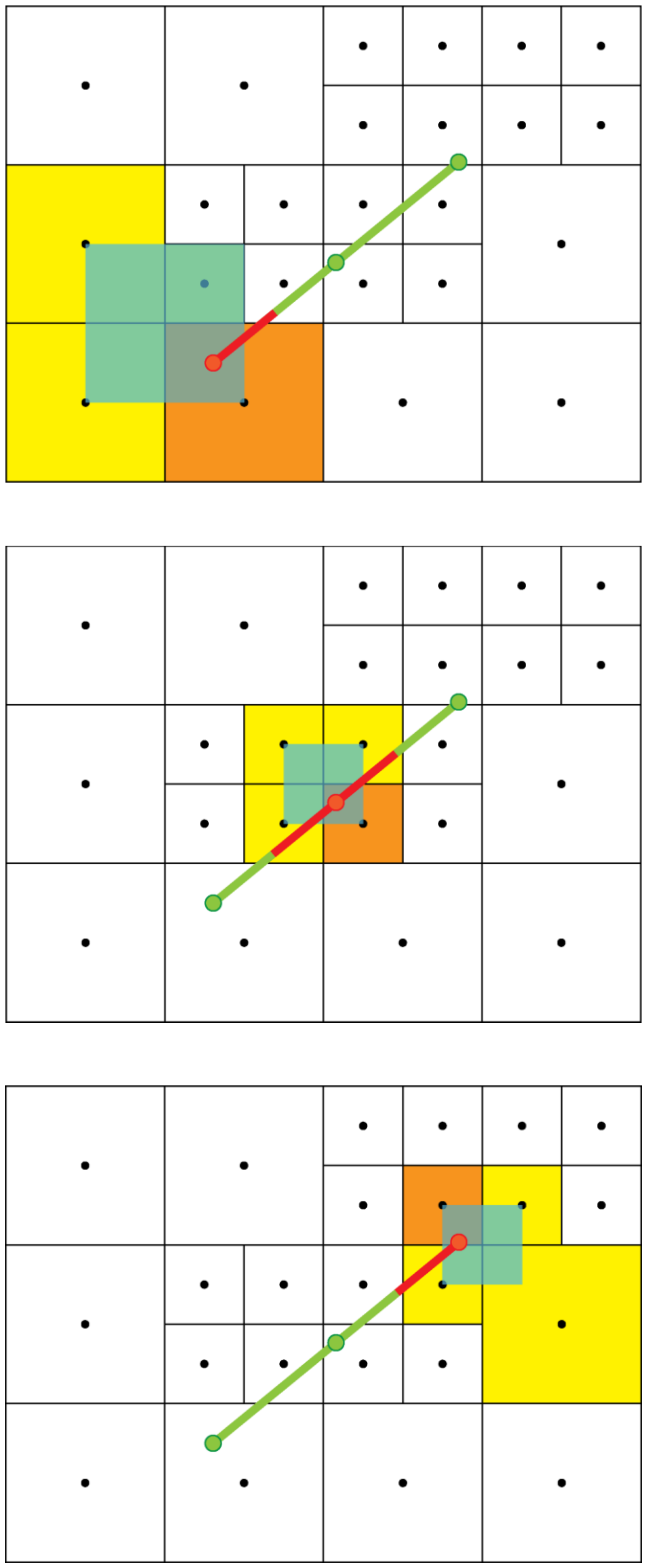


Figure IV-11: Observation Line. Top: First sample point. Center: Second sample point. Bottom: Third sample point.

8. SPECIAL SITUATIONS

Certain special situations can arise when refining cells. The cells in dam chosen for refinement can be adjacent to a reservoir or body of water. The bed elevations and initial water depths in the base-level cells are known only at the base resolution, so when these cells are refined, their values must be passed down to all their refined cells. This can be a problem because a base-level cell can be refined in such a way that some of the child cells are inside the dam and some are outside, as in Figure IV-12. The refined cells that are inside the dam take on the dam's crest elevation while the ones outside the dam retain the original bed elevation of the parent, as in the middle image of Figure IV-12. Those with the original bed elevation may be adjacent to a reservoir cell, but will inherit the dry initial condition from its dry parent, as in the right image in the figure. If these cells are not handled as part of the refinement process, then water from the adjacent cells will immediately rush into them in the first time step, causing sloshing in the reservoir before the dam begins to breach. This problem arises because the source of the input water depth data was not aware of the quadtree refinement structure. The solution is to recursively fill any refined cell with the water surface elevation of any of its wet neighbors. Thus the orange cells in Figure IV-12 would take their bed elevations from their parent cell, but they would receive the water surface elevation of the reservoir cells to their north or west.

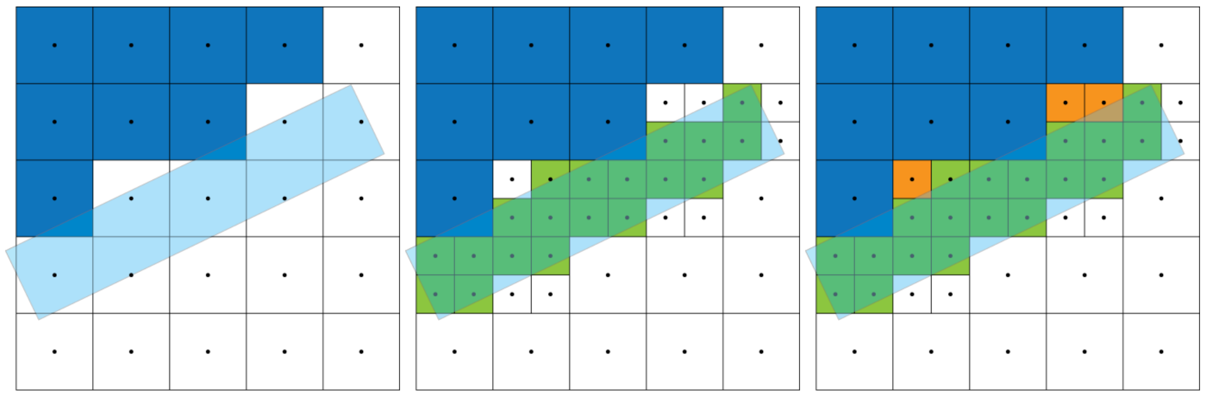


Figure IV-12: Illustration of problem when refining. Dark blue cells are in the reservoir while white cells are dry. The dam is the light blue rectangle. Green cells are refined cells in the dam. Orange cells highlight the cells that should have been part of the reservoir. Left: Dam is placed. Middle: Cells inside the dam are refined. Right: Orange cells are refined, but not part of the reservoir.

9. DISCUSSION

The quadtree local refinement method allows regions of larger cells to be replaced by a greater number of smaller cells. These refinements of larger cells into smaller cells allow modeling greater detail in the flow of water. The flow can be manipulated by altering the bed elevations of the refined cells, just like it is done in the regular mesh to model dam breach geometry. In addition, the discharges in the refined cells can also be probed and summed using observation lines. To accomplish quadtree local mesh refinement, regions of refinement must be selected via some means, and the desired level of refinement must be calculated. In the current work, the footprint of the dam is used to identify potential cells to be refined. The level of refinement is determined by the dam's given breach geometry profiles. During the refining process, the refinement algorithms are designed to obey certain rules to keep the mesh valid and ensure good results. These new processes allow a basic simulation definition to be automatically enhanced with quadtree local mesh refinement without additional input from the user. However, certain special situations arise when using this method, such as handling initial depths in refined cells near water bodies. In addition, the refinement process adds additional cells to the model which must be computed. This extra computational burden is the tradeoff for being able to resolve smaller features without having to refine the entire domain. In the next chapter, the accuracy and speed of the model will be tested to determine the costs and benefits of using the quadtree local refinement method.

V. RESULTS

The CCHE2D-FLOOD program with the quadtree local mesh refinement enhancements was tested by modeling different configurations of a progressive dam breach. The goal was to determine if different configurations of the same ideal model match each other at various global cell sizes with various levels of refinement. If the results given by these simulations are acceptable, then it is reasonable to believe that other configurations and other simulations will also give acceptable results. The accuracy and computational costs of simulations using quadtree refinement vs. global refinement are presented for these tests, followed by a discussion of the results.

1. SIMULATION SETUP

The ideal model presented in these results consists of a 128m by 128m domain, in which a 64m by 64m square reservoir is bounded by barriers on 3 sides and a dam on the fourth side. Two extra barriers 64m apart extend from the dam's side of the reservoir until they meet the edge of the domain. The domain edges are open boundaries to allow water to flow out. Referring to Figure V-1, the reservoir, barriers, and dam are aligned in one of three orientations: vertical or zero degrees of rotation, rotated 20 degrees clockwise, and rotated 45 degrees clockwise. These three ideal models are modeled using four global mesh sizes: 1m, 2m, 4m, and 8m. The dam in the 2m global mesh size simulation is modeled using unrefined 2m cells and using one level of refinement. The dam in the 4m global mesh size simulation is modeled using unrefined 4m cells, using one level of refinement (2m cells), and using two levels of refinement (1m cells). The dam in the 8m global mesh size simulation is modeled using unrefined 8m cells, using one level of refinement (4m cells), using two levels of refinement (2m cells), and using three levels of refinement (1m cells). In addition, there is another group of simulations using a 512m by 512m domain with a vertically-aligned 128m by 512m reservoir and dam with the same levels of refinement above. Table 1 summarizes each group in the 40 total simulations. The first column describes the global cell size and the highest level of refinement for the locally-refined cells. For example, "1mL0" is the simulation using 1m cells with no refinement, and "8mL3" is the simulation with the global cell size equal to 8m with three levels of refinement.

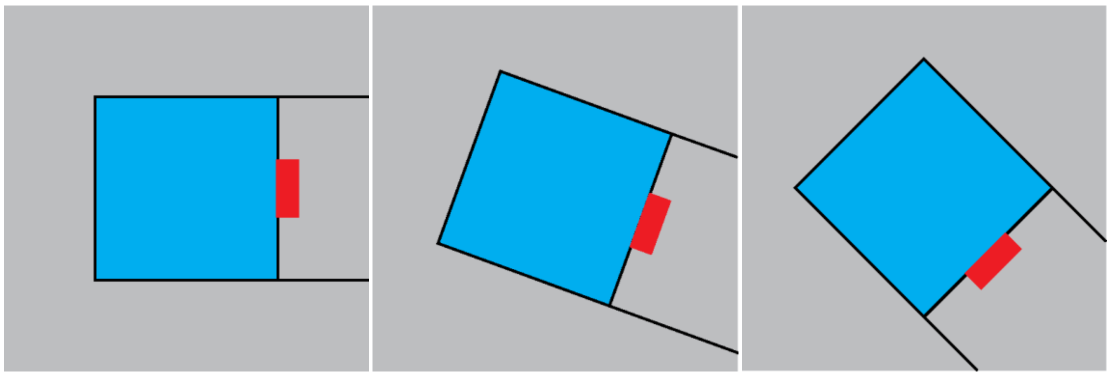


Figure V-1: Ideal Domain for Left: 0 degrees rotation, middle: 20 degrees rotation, right: 45 degrees rotation. Blue is reservoir area. Red is the dam. Black is barriers. Gray is background.

Table 1: Simulation Information. Four groups of these simulations are done: 1. 64mx64m reservoir with 0 degree orientation. 2. 64mx64m reservoir with 20 degree orientation. 3. 64mx64m reservoir with 45 degree orientation. 4. 128mx512m reservoir with 0 degree orientation

Simulation	**Cell Size (m)**	**Refinement Level**	**Smallest Cell Size (m)**
1mL0	1	0	1
2mL0	2	0	2
2mL1	2	1	1
4mL0	4	0	4
4mL1	4	1	2
4mL2	4	2	1
8mL0	8	0	8
8mL1	8	1	4
8mL2	8	2	2
8mL3	8	3	1

In each case the barriers are represented by raising the bottom elevation of whole cells to a level above the reservoir. Care has been taken to ensure that the position of the barriers is such that the reservoir contains approximately the same area at all resolutions. Similarly, the dam width is chosen as 8m to ensure that it is modeled the same in every case. Each orientation and resolution combination is shown in Figure V-2.

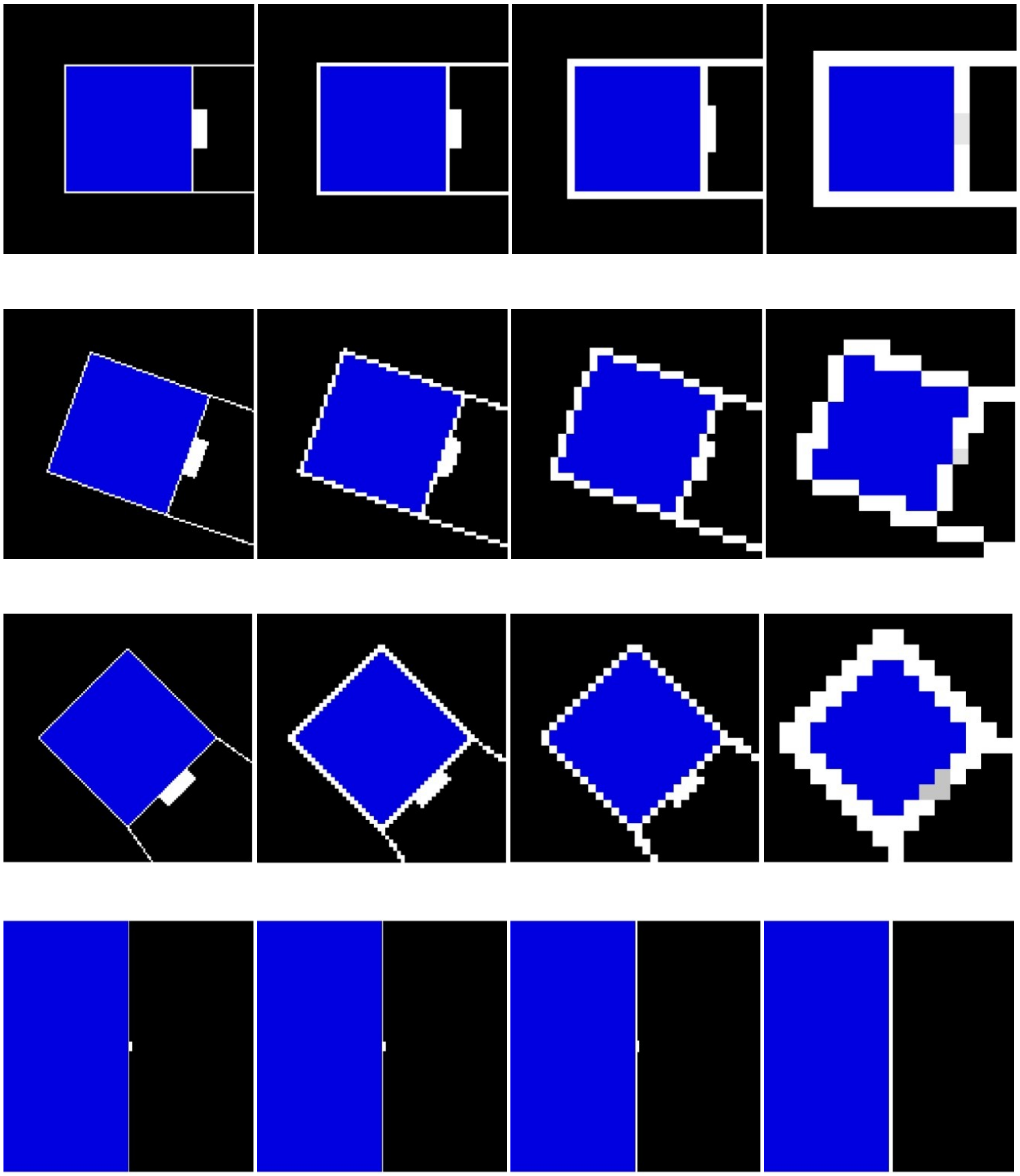


Figure V-2: From left to right in a row: 1m resolution, 2m resolution, 4m resolution, 8m resolution. Row 1: 0 degree orientation. Row 2: 20 degree orientation. Row 3: 45 degree orientation. Row 4: Larger simulation with 0 degree orientation. Blue color indicates reservoir area. White indicates barriers and dams. Black indicates background areas.

An observation line is placed along the centerline of the dam in each simulation, giving a measure of the total discharge flowing through the breach at every second. Each simulation is run for 800 seconds model time. The wall time, CPU time, and number of iterations are recorded for each simulation along with a raster file of water depths outputted every 100 seconds.

2. BREACH CHARACTERISTICS

The dam breach is the same in each of the 40 test cases. The dam profile changes from the initial, intact dam height to the final breach geometry in 300 seconds. Elevations of cells within the dam are interpolated linearly in both time between successive breach profiles and their position between points in the profile. The final breach geometry has a top width of 16m and a bottom width of 8m, giving an average width of 12m. The depth of the breach is 5m, the same as the height of the dam, and the side slopes are 4H:5V. The breach geometry at every 100 seconds is plotted in Figure V-3. The size and formation time of this breach are selected to allow modeling at the various cell sizes in the cases described, but are typical for the reservoir-dam pair chosen here for testing. The progression of the breach geometry starts at the level of the dam crest, then the full width of the breach is brought down to its final profile over the 300 second duration. However, actual dam breaches usually follow a different progression, where a smaller-width gap progresses downward and then expands to its full width. Representing more realistic dam breach progressions can be done in the future. Figure V-4 shows an example of the progression of breach geometry for the 0 degree orientation at 100 second intervals for the highest level of refinement. Figure V-5 shows the same for the 20 degree orientation. Figure V-6 shows the same for 45 degree orientation.

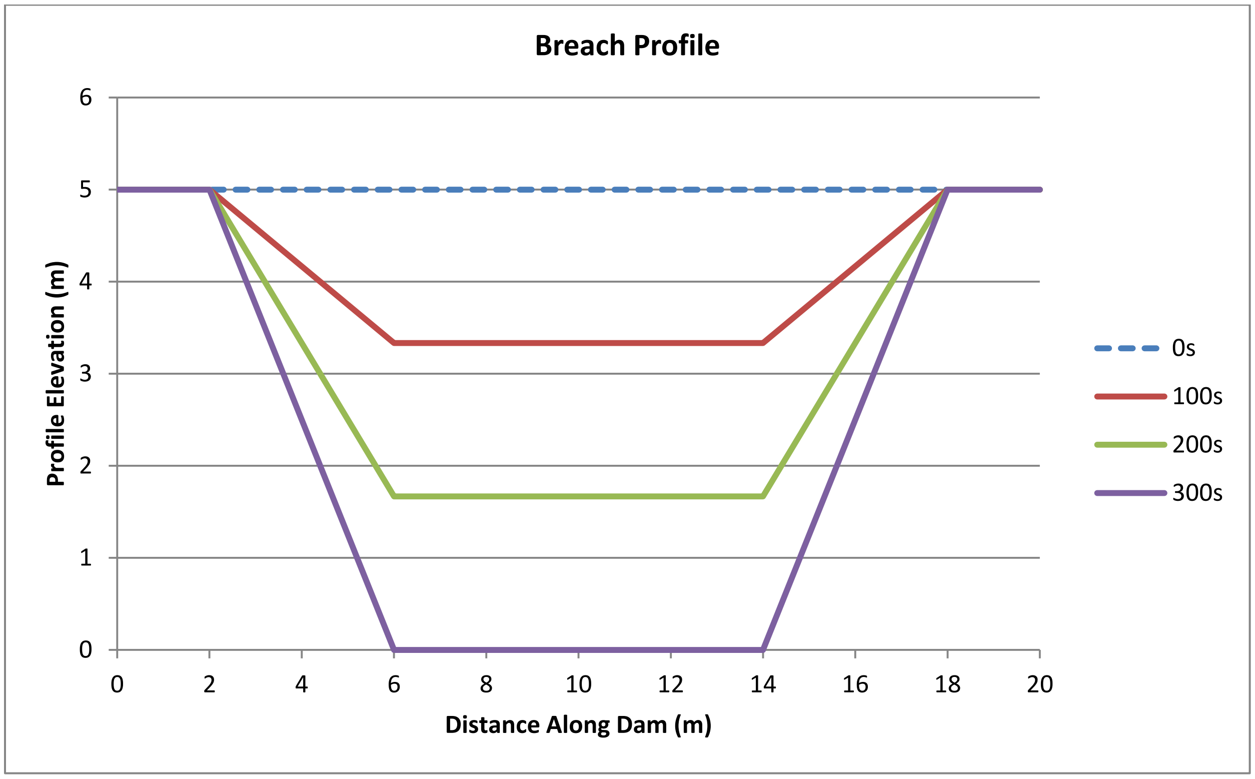


Figure V-3: Progression of Breach Geometry

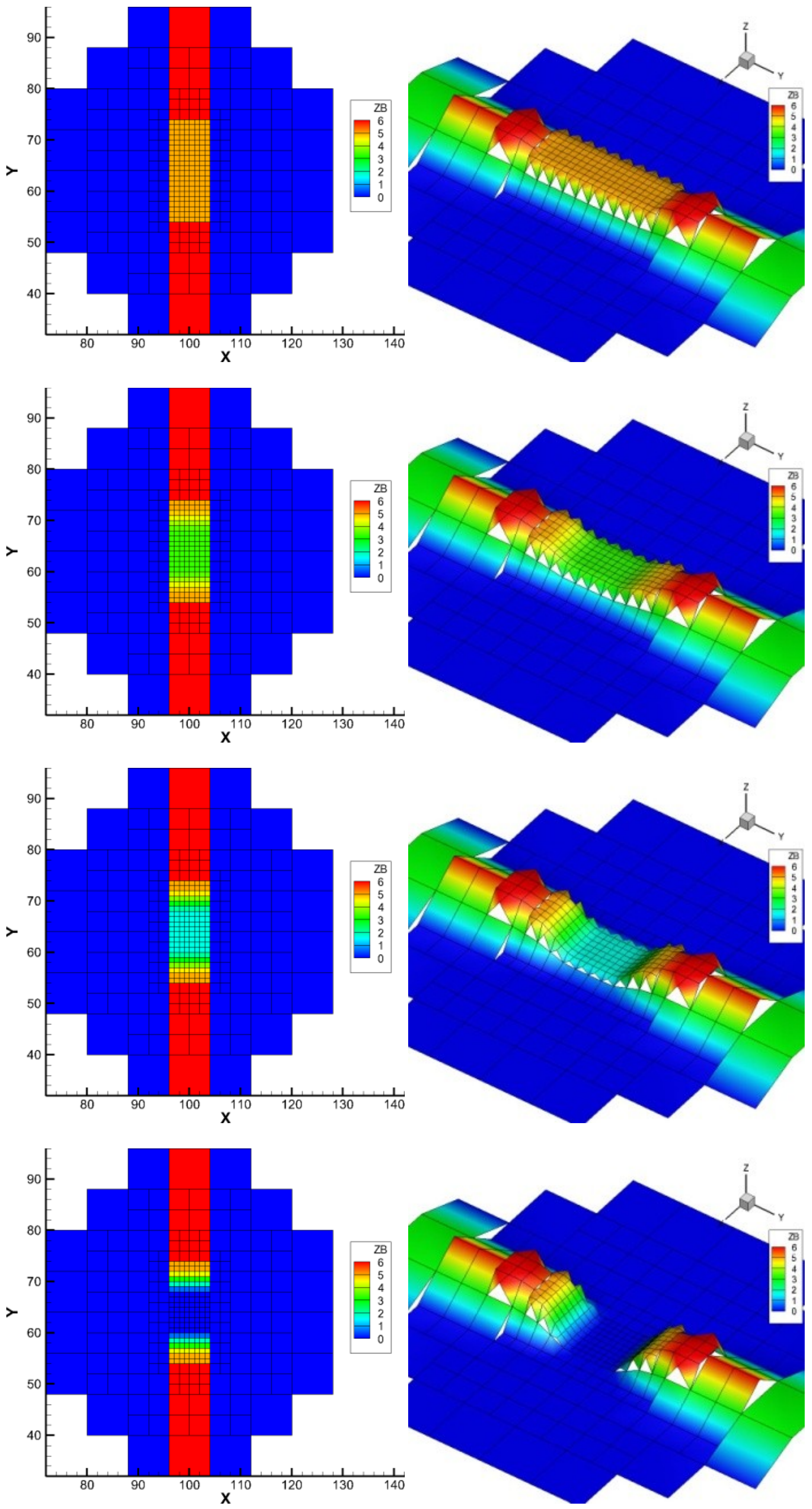


Figure V-4: Progression of breach geometry for 0 degree orientation. Left: Plan view. Right: 3D View looking upstream. Row 1: Initial geometry at 0 seconds. Row 2: 100 seconds. Row 3: 200 seconds. Row 4: 300 seconds, final geometry.

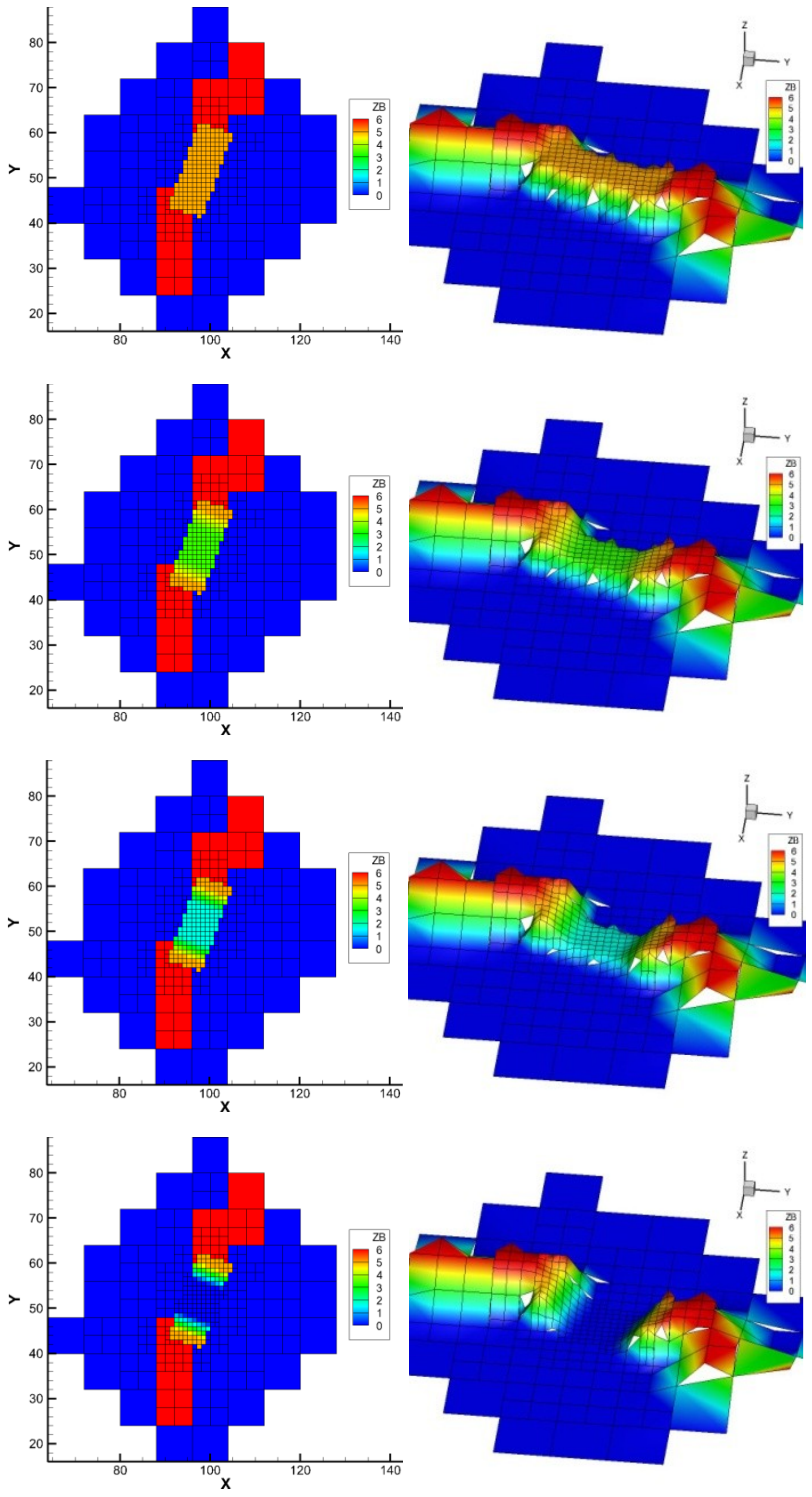


Figure V-5: Progression of breach geometry for 20 degree orientation. Left: Plan view. Right: 3D View looking upstream. Row 1: Initial geometry at 0 seconds. Row 2: 100 seconds. Row 3: 200 seconds. Row 4: 300 seconds, final geometry.

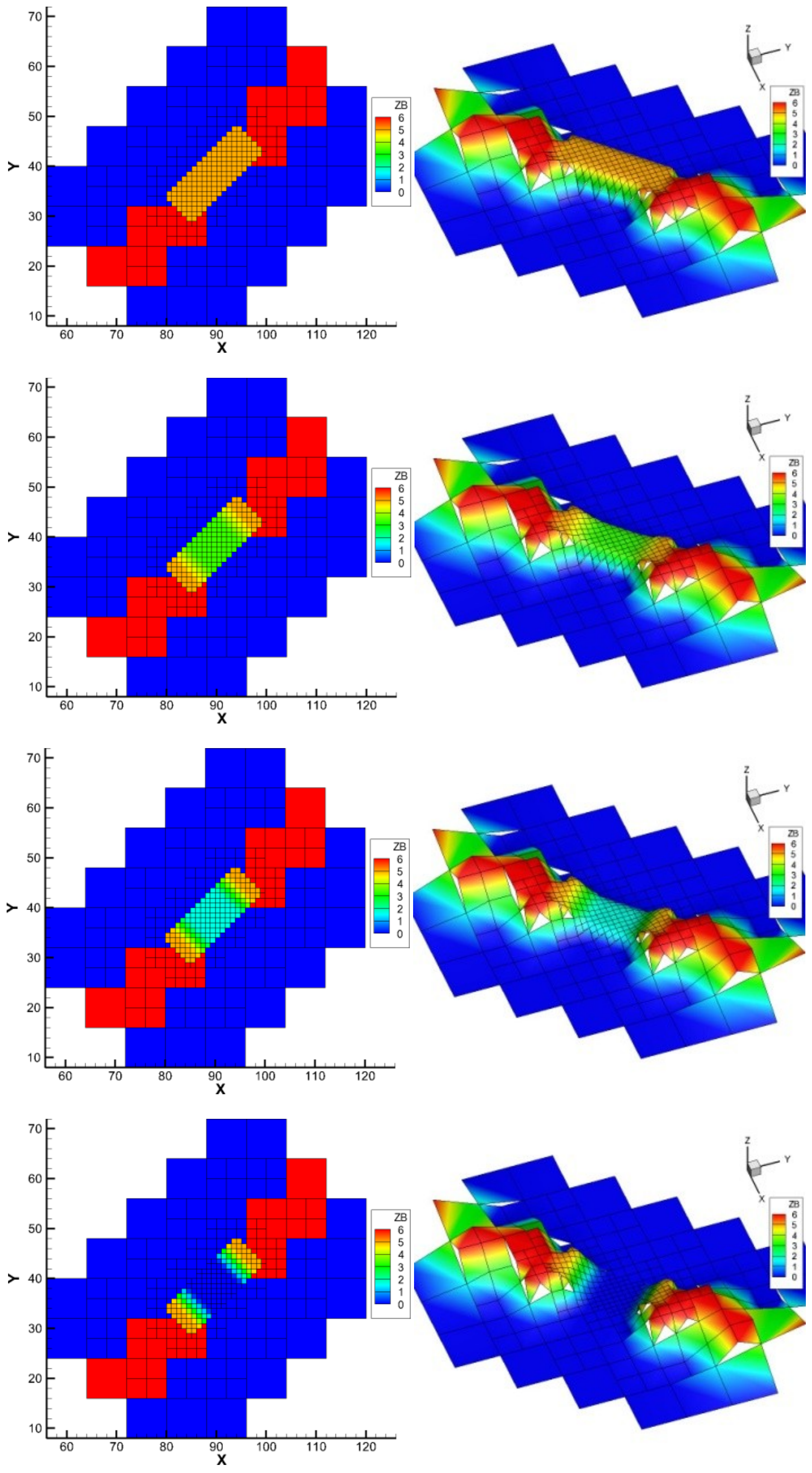


Figure V-6: Progression of breach geometry for 45 degree orientation. Left: Plan view. Right: 3D View looking upstream. Row 1: Initial geometry at 0 seconds. Row 2: 100 seconds. Row 3: 200 seconds. Row 4: 300 seconds, final geometry.

The orientation of the refined section of the dam in the 20 and 45 degree cases slightly overlaps with the cells in the boundary definition that do not participate in the breaching process. The positions of the barriers around the reservoir are originally placed in the domain by imposing high elevations in those cells at the global resolution. Even if some of those cells are later refined, the higher elevations may remain and slightly interfere with the flow through the dam breach. However, only the edges of the dam overlap these areas, and the breach geometry has at least 2 meters of buffer on each end that remains at the original dam height. In future work, the extent to which these barrier cells interfere with the flow near the dam cells will be determined.

Since the breach geometry is the same for all test cases, the resolution of the finest cells in the simulation determine how many of them are used to model the breach. Table 2 shows the number of finest-cell widths that can fit within the 16m breach geometry. The orange-colored values highlight the simulations where the refinement level and global cell size result in only a few cells representing the breach. The yellow value is the worst-case among the simulations, with only 2 cells able to fit in the breach width.

Table 2: Number of finest-resolution cells that can fit within the dam breach width for each simulation. Highlighted values show combinations with a small number of cells.

Simulation	**Cell Size (m)**	**Refinement Level**	**Smallest Cell Size (m)**	**Breach Width (m)**	**Number of finest-cell widths in breach**
1mL0	1	0	1	16	16
2mL0	2	0	2	16	8
2mL1	2	1	1	16	16
4mL0	4	0	4	16	4
4mL1	4	1	2	16	8
4mL2	4	2	1	16	16
8mL0	8	0	8	16	2
8mL1	8	1	4	16	4
8mL2	8	2	2	16	8
8mL3	8	3	1	16	16

3. SIMULATION RESULTS

For each orientation, 10 simulations were run using various global cell sizes and levels of refinement. A set of example results for each orientation is shown in Figure V-7.

The cases with 1m global resolution without refinement are considered to give the correct discharge through the breach: we consider these to be the control simulations. Figure V-8 shows the discharge vs. time plots for the control simulations where the global cell size is 1m and no refinement is done. Table 3 shows the peak discharges for these control simulations with percent error calculated relative to the 0 degree case. These are considered to be the "correct" answers to which other simulations in their group are compared. All three simulations give similar results, with their peak discharge matching within 10% of each other.

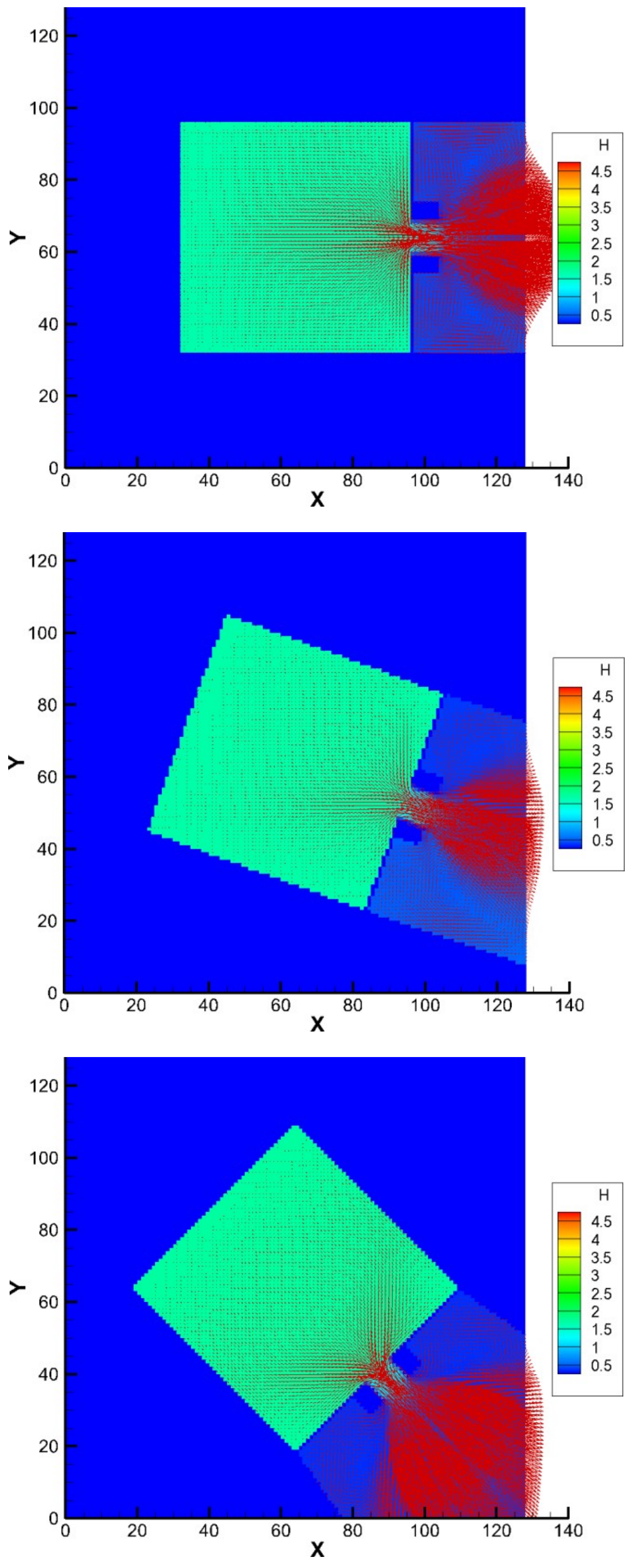


Figure V-7: Water depths and red velocity vectors of simulations at 400 seconds. Top: 0 degree rotation case. Middle: 20 degree rotation case. Bottom: 45 degree rotation case.

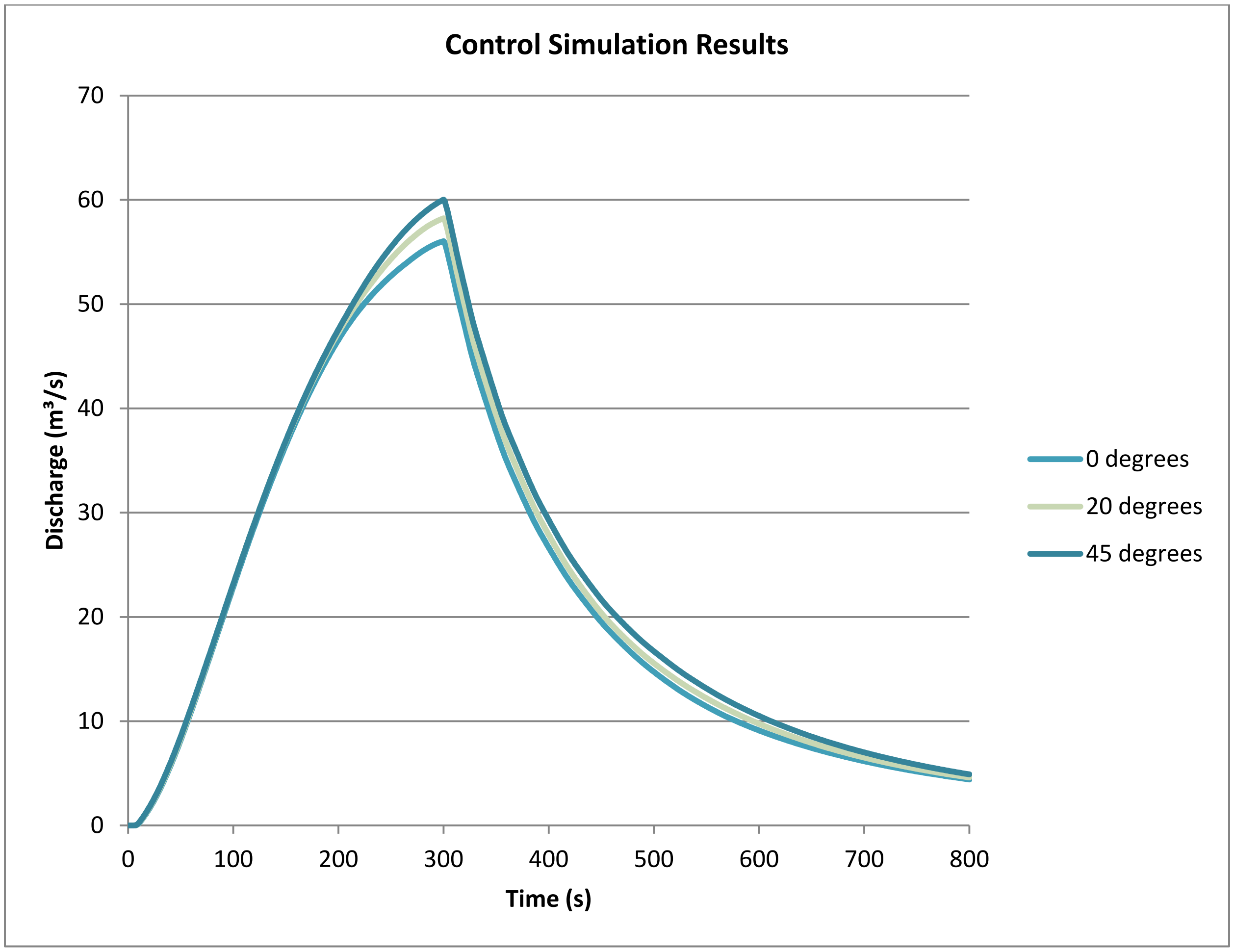


Figure V-8: Discharge vs. time for 1m unrefined simulations

Table 3: Peak Discharges for control simulations

	Maximum Discharge (m^3/s)	**% Error**
0deg	56.0	0.0
20deg	58.2	3.9
45deg	60.0	7.1

For the simulations with 0 degree orientation, the discharges vs. time are plotted in Figure V-9 with corresponding maximum or peak discharges given in Table 4. Results of simulations with 20 degree orientation are plotted in Figure V-10 with peak discharges in Table 5. Results of simulations with 45 degree orientation are plotted in Figure V-11 with discharges in Table 6. In the tables, the percent error is calculated relative to the peak discharge in their 1mL0 case. In addition, the 8m and unrefined 4m simulation results are highlighted to show their higher relative error.

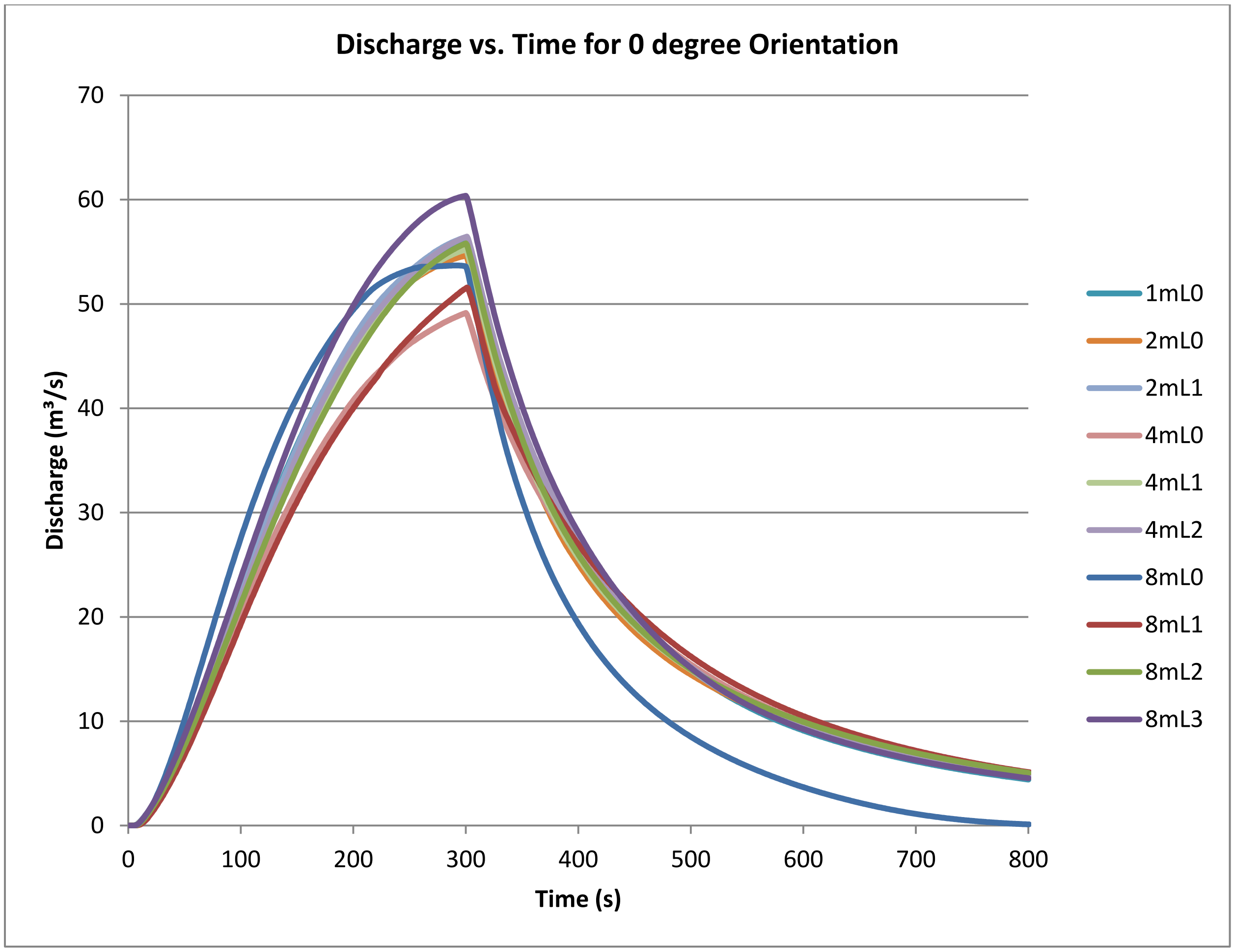


Figure V-9: Discharge vs. Time for 0 degree Orientation Simulations

Table 4: Peak Discharges for 0 degree cases

	Maximum Discharge (m^3/s)	**% Error**
1mL0	56.0	0.0
2mL0	54.6	-2.5
2mL1	56.4	0.7
4mL0	49.1	-12.3
4mL1	55.2	-1.5
4mL2	56.5	0.7
8mL0	53.7	-4.2
8mL1	51.6	-8.0
8mL2	55.8	-0.4
8mL3	60.4	7.7

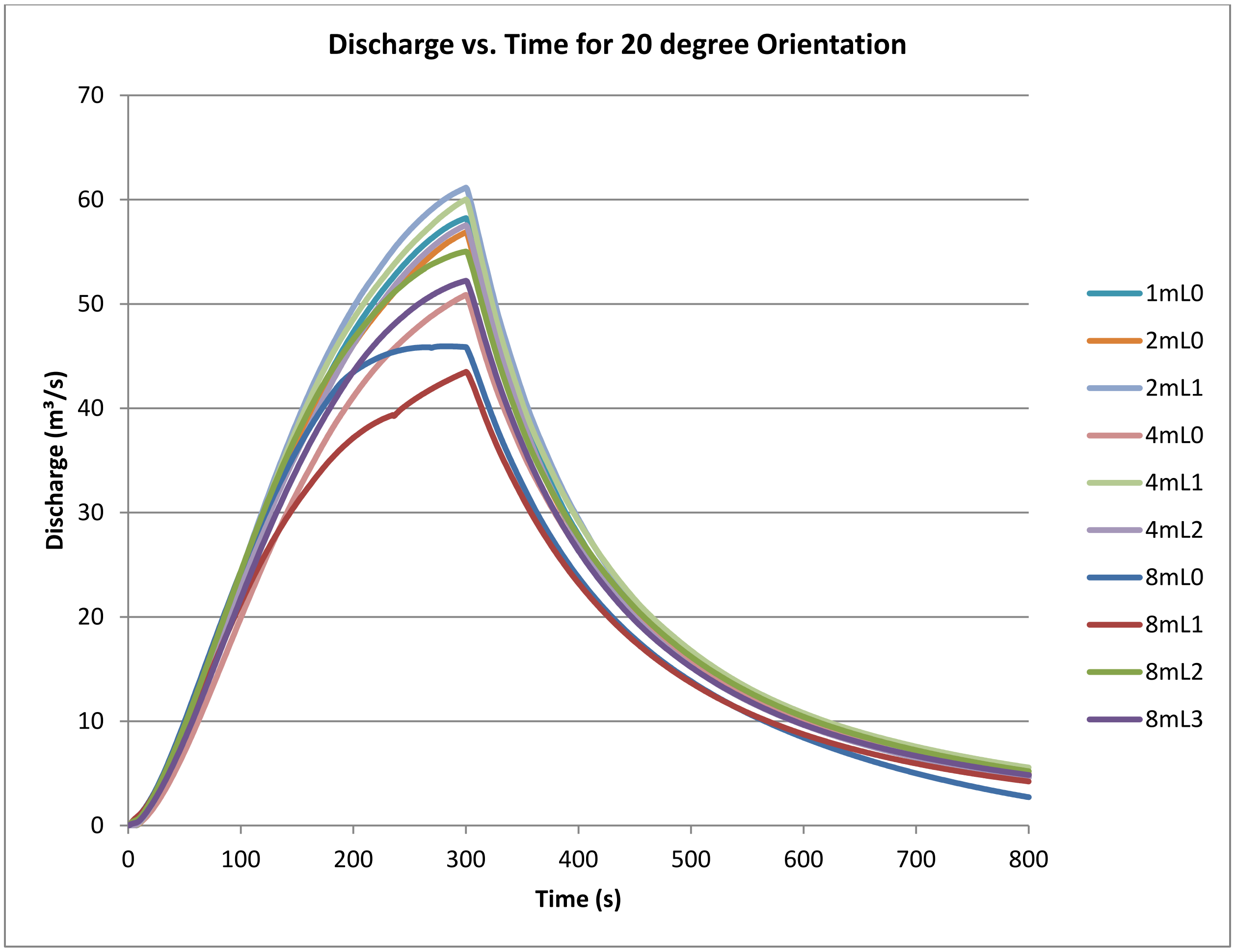


Figure V-10: Discharge vs. Time for 20 degree Orientation Simulations

Table 5: Peak Discharges for 20 degree cases

	Maximum Discharge (m^3/s)	**% Error**
1mL0	58.2	0.0
2mL0	56.9	-2.3
2mL1	61.2	5.0
4mL0	50.9	-12.6
4mL1	60.0	3.1
4mL2	57.6	-1.2
8mL0	45.9	-21.1
8mL1	43.5	-25.3
8mL2	55.0	-5.5
8mL3	52.2	-10.3

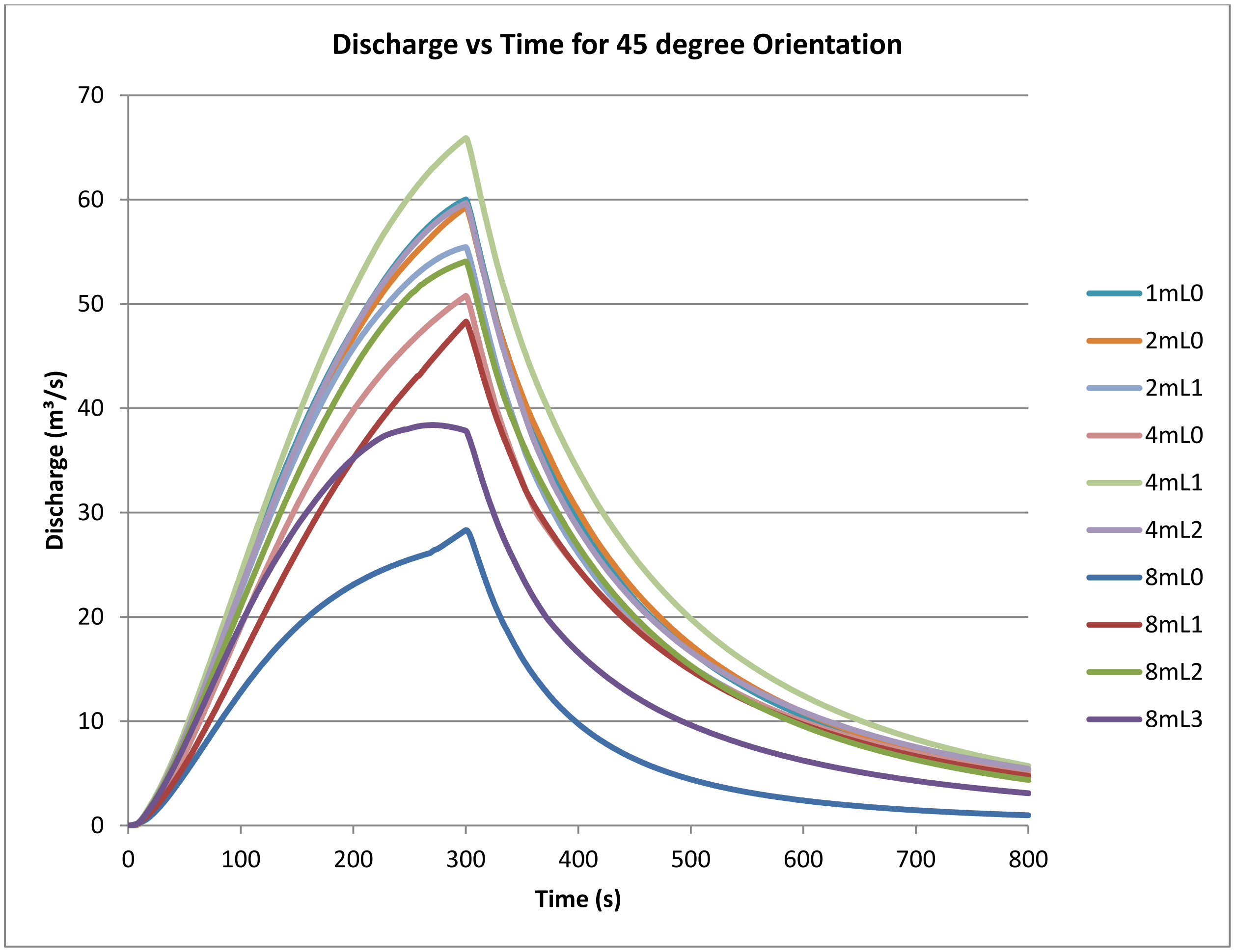


Figure V-11: Discharge vs. Time for 45 degree Orientation Simulations

Table 6: Peak Discharges for 45 degree cases

	Maximum Discharge (m^3/s)	**% Error**
1mL0	60.0	0.0
2mL0	59.3	-1.2
2mL1	55.5	-7.6
4mL0	50.8	-15.4
4mL1	65.9	9.8
4mL2	59.7	-0.6
8mL0	28.3	-52.8
8mL1	48.3	-19.5
8mL2	54.1	-9.9
8mL3	38.4	-36.0

These plots show that the discharge peaks after 300 seconds, after the dam breach has reached its full size. The reservoir empties quickly after the peak.

In the 0 degree case, the 2mL1 simulation has lower error in the peak discharge than the 2mL0 simulation. In the 20 degree and 45 degree cases, the opposite is true, showing that for those particular simulations, a higher level of refinement did not improve the results.

In all cases, the 4m global cell size simulations with higher levels of refinement had smaller errors in peak discharge than the ones with lower levels of refinement. In some cases the error changed sign between refinement levels, but overall the errors were reduced.

In all three orientations, the 8mL2 simulations had lower errors than the 8mL0 and 8mL1 simulations in their groups. However, the 8mL3 simulation in each orientation actually had a higher error than the 8mL2 simulation, suggesting that higher levels of refinement may not correlate with higher accuracy. Clearly there is room for future work to determine if refining to higher levels continues to increase accuracy, or if at some point the higher levels introduce more errors to counter the increase.

In each of these groups, the 8m simulations and the 4m unrefined simulations have the highest percent error relative their controls and are thus the least accurate. Removing these simulation results from each of the three orientation groups and combining the 15 remaining simulations gives the results in Figure V-12. This figure shows that the remaining cases more closely match each other.

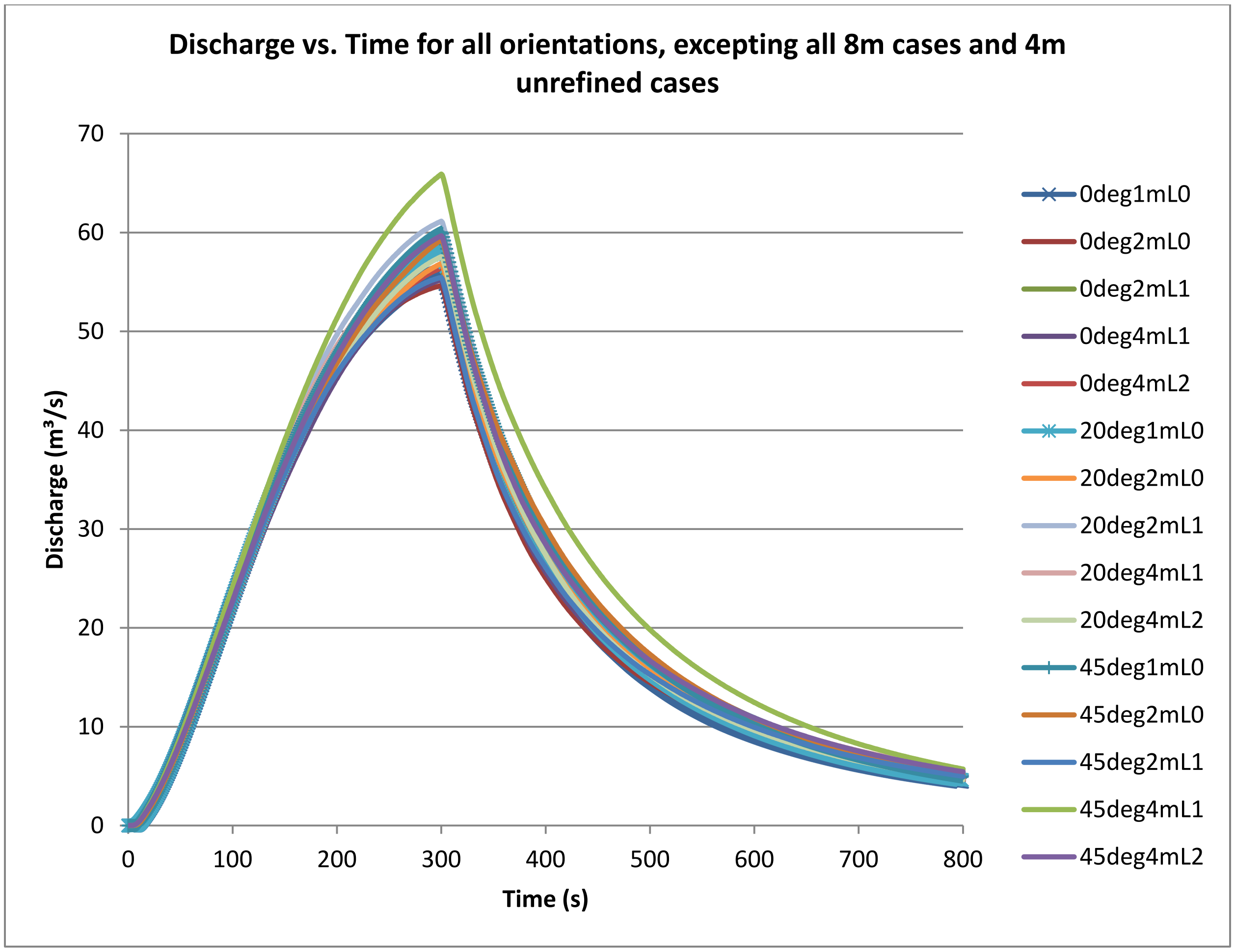


Figure V-12: Discharge vs. time for all cases except 8m cases and 4m unrefined cases

4. PERFORMANCE

Each simulation was run, one after the other, with no other major processes running on the machine. The machine was a desktop running Microsoft Windows 7 with a quad-core Intel i7 870 CPU and 14 GB of RAM. Table 7, Table 8, and Table 9 show the relevant timing information for the 0 degree, 20 degree rotated, and 45 degree rotated simulation groups, respectively. CPU time is a measure of the computational effort spent by all cores of all processors used by a single computer process. CPU time is less-influenced by other processes running on the machine, drive lag times, etc. than wall time. The wall time is a measure of real-world duration of when the process begins to when the process ends as though measured by a clock, regardless of interruptions or competition for computer resources from other running processes. The number of iterations is the total number of time steps required to reach the full 800 seconds of simulation. The "QT cells" and "Leaf cells" columns give the total number of cells in the quadtree mesh and the total number of leaf cells in the quadtree mesh, respectively. The Speedup of each simulation is the CPU time taken by the 1m simulation divided by the CPU time taken by the given simulation. The theoretical speedup is the cube of the relative difference in global cell size compared to the 1m simulation.

Table 7: Simulation Characteristics for 0 degree Simulations

	CPU time (s)	**Wall time (s)**	**Iterations**	**QT Cells**	**Leaf Cells**	**Global Cell size (m)**	**Speedup**	**Theoretical Speedup**
1mL0	91.45	92	26365	-	-	1	1.0	1
2mL0	11.23	12	13070	-	-	2	8.1	8
2mL1	12.98	13	13606	256	216	2	7.0	8
4mL0	1.53	2	6391	-	-	4	59.8	64
4mL1	1.79	2	6874	96	84	4	51.0	64
4mL2	2.79	3	6922	324	256	4	32.7	64
8mL0	0.17	1	2340	-	-	8	532.9	512
8mL1	0.28	1	3420	38	34	8	325.7	512
8mL2	0.50	1	3499	132	106	8	183.2	512
8mL3	1.56	2	3554	420	326	8	58.6	512

Table 8: Simulation Characteristics for 20 Degree Simulations

	CPU time (s)	Wall time (s)	Iterations	QT Cells	Leaf Cells	Global Cell size (m)	Speedup	Theoretical Speedup
1mL0	94.83	95	26512	0	0	1	1.0	1
2mL0	11.84	12	13221	0	0	2	8.0	8
2mL1	13.81	14	13895	300	250	2	6.9	8
4mL0	1.48	1	6353	0	0	4	64.0	64
4mL1	1.93	1	7015	99	85	4	49.0	64
4mL2	3.23	3	7055	401	315	4	29.4	64
8mL0	0.22	1	2858	0	0	8	434.2	512
8mL1	0.31	1	3472	42	37	8	304.0	512
8mL2	0.53	1	3519	140	112	8	178.8	512
8mL3	1.78	2	3666	491	379	8	53.3	512

Table 9: Simulation Characteristics for 45 Degree Simulations

	CPU time (s)	Wall time (s)	Iterations	QT Cells	Leaf Cells	Global Cell size (m)	Speedup	Theoretical Speedup
1mL0	98.00	99	26940	0	0	1	1.0	1
2mL0	11.97	12	13276	0	0	2	8.2	8
2mL1	13.88	14	13955	282	237	2	7.1	8
4mL0	1.54	2	6439	0	0	4	63.5	64
4mL1	2.04	2	7182	104	88	4	48.0	64
4mL2	3.14	4	7104	354	278	4	31.3	64
8mL0	0.20	1	2943	0	0	8	483.2	512
8mL1	0.62	1	3595	44	39	8	157.1	512
8mL2	0.56	1	3550	158	126	8	174.5	512
8mL3	1.54	1	3542	446	345	8	63.5	512

The CPU and wall time for the simulations follow the expected order, with 8m simulations taking less time than the 4m simulations which take less time than the 2m simulations, which in turn take less time than the 1m simulations. Simulations with higher levels of refinement are slightly slower than the simulations with lower levels of refinement in the same global cell size class. Simulations with a larger global cell size take fewer iterations than simulations with smaller global cell size due to the larger time step. The speedup of each simulation is approximately in the same order of magnitude as the theoretical case, except for the 8m refined to 3 levels simulations. However, these domains are so small, and the simulations finish so quickly, that a significant portion of the CPU time may have been spent doing auxiliary tasks such as writing raster files, computing and writing to disk the observation line discharges, etc. For these 30 simulations we see a general trend in the CPU time, but the group of simulations described in the next section is a better representative. Rather than to compare simulation times, the simulations above were designed to measure and compare the discharge flowing through the breach to ensure the results are correct.

5. LARGER DOMAIN CASE

The final batch of simulations involve a square domain 512m by 512m with a vertical barrier dividing the domain in two. The same dam and breach as in the previous simulations is situated vertically in the center as shown in the bottom row of Figure V-2 above. In these simulations, the domain and reservoir are much larger than in the previous simulations, so the computational time is expected to be longer and more differentiated between various resolutions. Figure V-13 shows the discharge vs. time for these simulations. Table 10 shows the peak discharges for these cases. The unrefined 8m simulation shows discharges higher than all over simulations and is thus the least accurate, as expected.

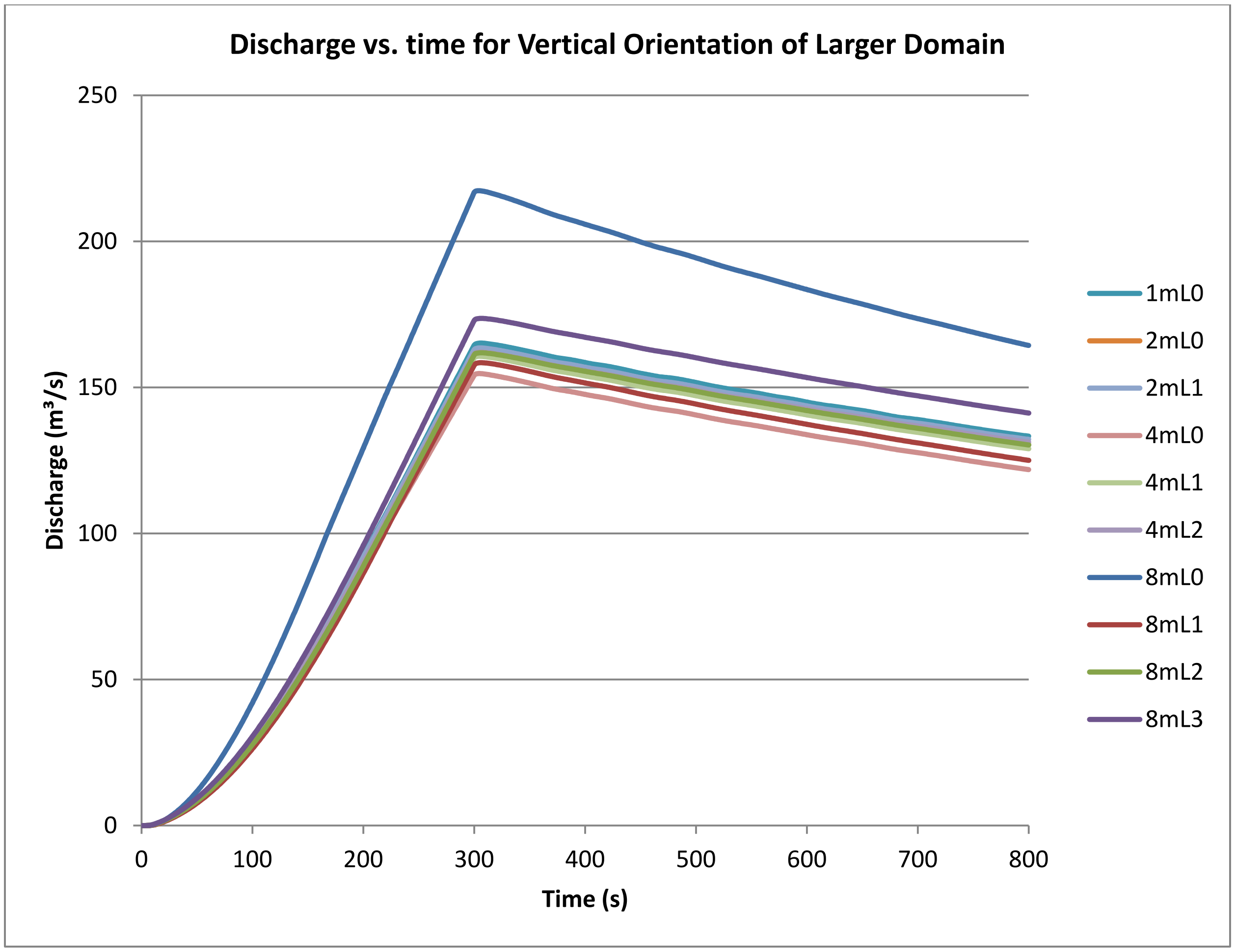


Figure V-13: Discharge vs. time for simulation of larger domain with 0 degree orientation

Table 10: Peak Discharges for larger domain simulations

	Maximum Discharge (m^3/s)	**% Error**
1mL0	165.2	0.0
2mL0	162.0	-1.9
2mL1	163.5	-1.0
4mL0	154.8	-6.3
4mL1	160.7	-2.7
4mL2	162.1	-1.9
8mL0	217.3	31.6
8mL1	158.5	-4.1
8mL2	161.9	-2.0
8mL3	173.6	5.1

One of the goals of local mesh refinement is to attain the same discharges through the dam breach with a coarser global cell size and finer resolution for the breach as with using a smaller global cell size. Figure V-14 compares the breach discharges for the four test cases using 1m for their finest cell resolution. These results agree quite well, with the 8m simulation with 3 levels of refinement being slightly less accurate, as in the cases in the previous section.

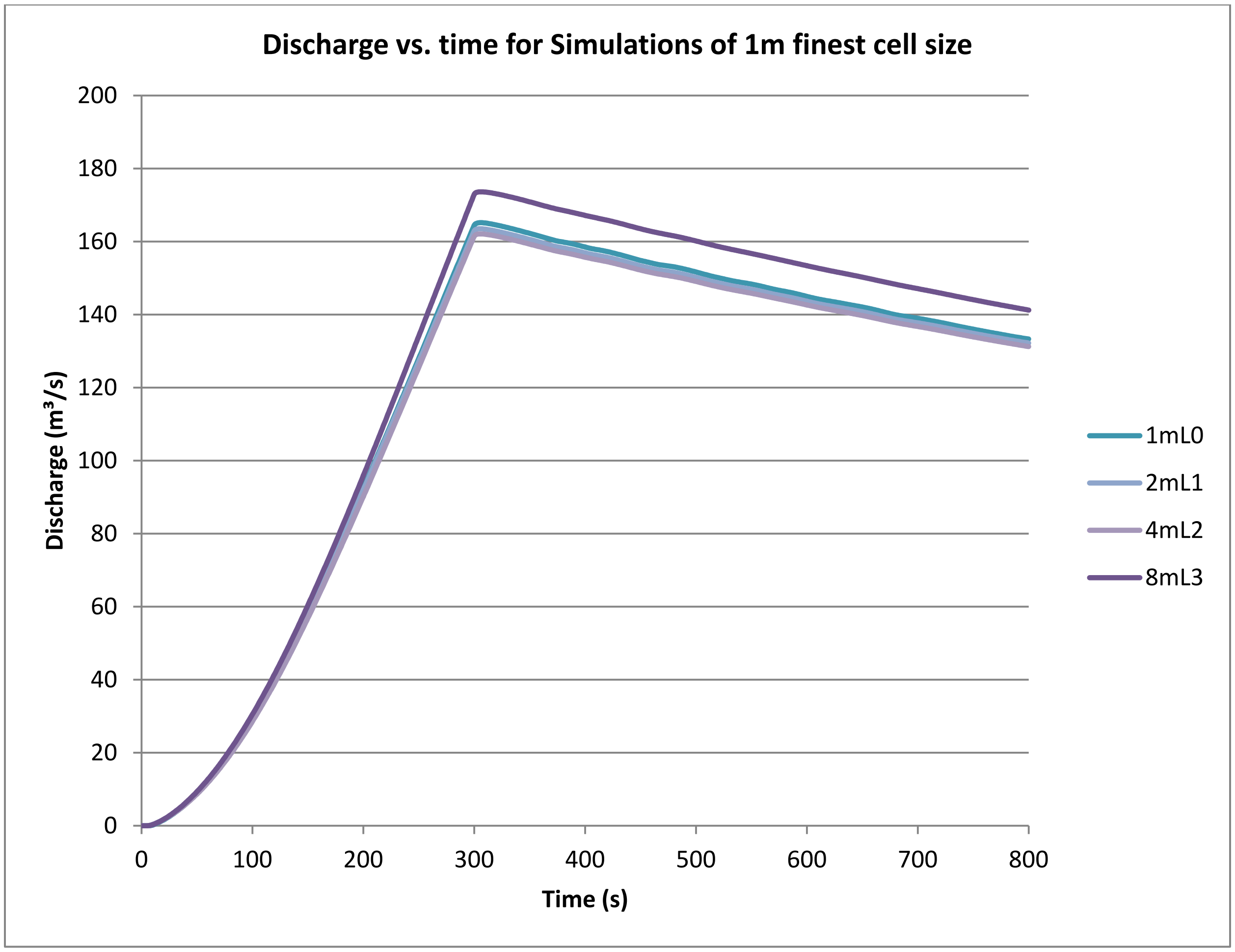


Figure V-14: Discharge vs. time of larger domain simulation comparing simulations where the finest-resolution cells were 1m.

Figure V-15 shows the discharge vs. time of the best cases, where the 8m simulations and the unrefined 4m simulation are excluded. These results also agree quite well.

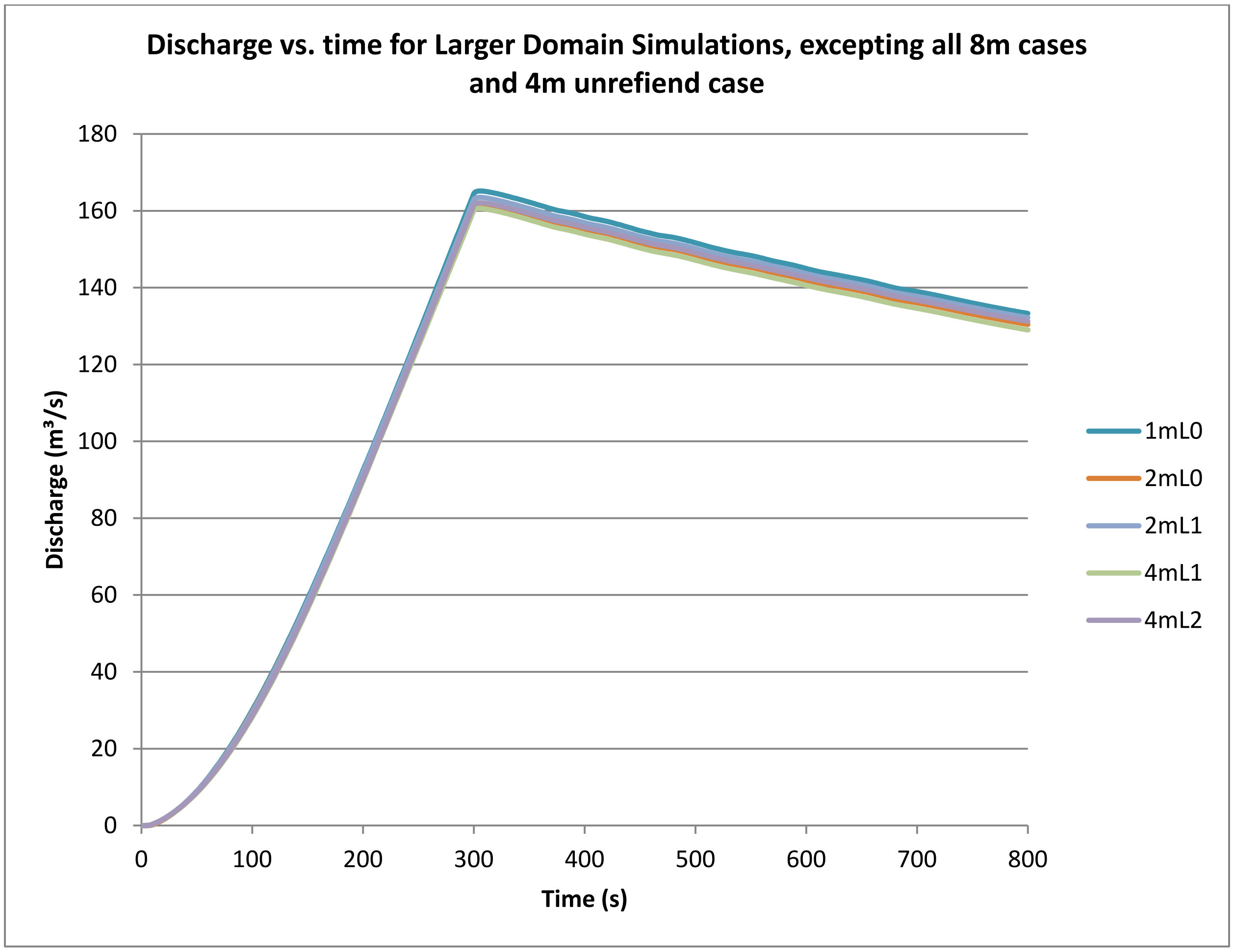


Figure V-15: Discharge vs. time of larger domain simulation for all cases except 8m cases and 4m unrefined case.

6. PERFORMANCE OF LARGER DOMAIN CASE

Table 11 shows the relevant simulation information for the larger domain case run on the same machine mentioned above. Like the smaller cases in the previous sections, the a general trend is present in the CPU time taken by these simulations. The 1m global cell size simulation takes the longest, with each successive doubling of the global cell size decreases the CPU time by a factor of about 8. These results agree with the theoretical speedup much better than the smaller test cases since the domains are large enough to spend a proportionally-larger amount of the time calculating the hydrodynamics and a proportionally-smaller time doing auxiliary tasks.

Table 11: Simulation Characteristics for Larger Domain Simulations

	CPU time (s)	**Wall time (s)**	**Iterations**	**QT Cells**	**Leaf Cells**	**Global Cell size (m)**	**Speedup**	**Theoretical Speedup**
1mL0	3409.00	3114	44039	-	-	1	1.0	1
2mL0	413.92	414	21719	-	-	2	8.2	8
2mL1	422.31	423	22035	256	216	2	8.1	8
4mL0	48.06	49	10494	-	-	4	70.9	64
4mL1	50.20	50	10860	96	84	4	67.9	64
4mL2	52.57	52	11012	324	256	4	64.8	64
8mL0	5.93	6	4904	-	-	8	575.1	512
8mL1	6.36	6	5238	38	34	8	535.6	512
8mL2	6.65	6	5419	132	106	8	513.0	512
8mL3	8.53	8	5612	420	326	8	399.5	512

The cube root of the speedup may be considered a measure of the computational burden brought on by the change in cell size. Plotting this for each of the 10 simulations vs. its cell size along with a line of theoretical maximum speedup is shown in Figure V-16. Some of the simulations exhibit a speedup greater than the theoretical maximum. The simulations with larger cell sizes have fewer total cells and a larger time step than the 1m case, and these are the only two factors that contribute to the theoretical speedup. Other factors could be coming into play, such as better CPU cache access for the simulations with fewer, larger cells.

Another trend to note is that for a given global cell size, the simulations with higher levels of refinement take longer than simulations with lower levels of refinement. This is due to the additional burden of calculating the additional, refined cells at their smaller time steps. For instance, even though the domain is large compared to the dam, the 8mL3 case has 420 quadtree cells plus 4,096 8m base cells, or about 10% more cells than the 8mL0, unrefined case. In addition, the quadtree cells of higher refinement level must be calculated multiple times for each global time step, adding to the expected result of slowing down the simulation vs. the cases without refinement. Despite the added computational burden of even the highest levels of refinement, the simulations with larger global cell sizes still vastly outperform the simulations with smaller global cell sizes.

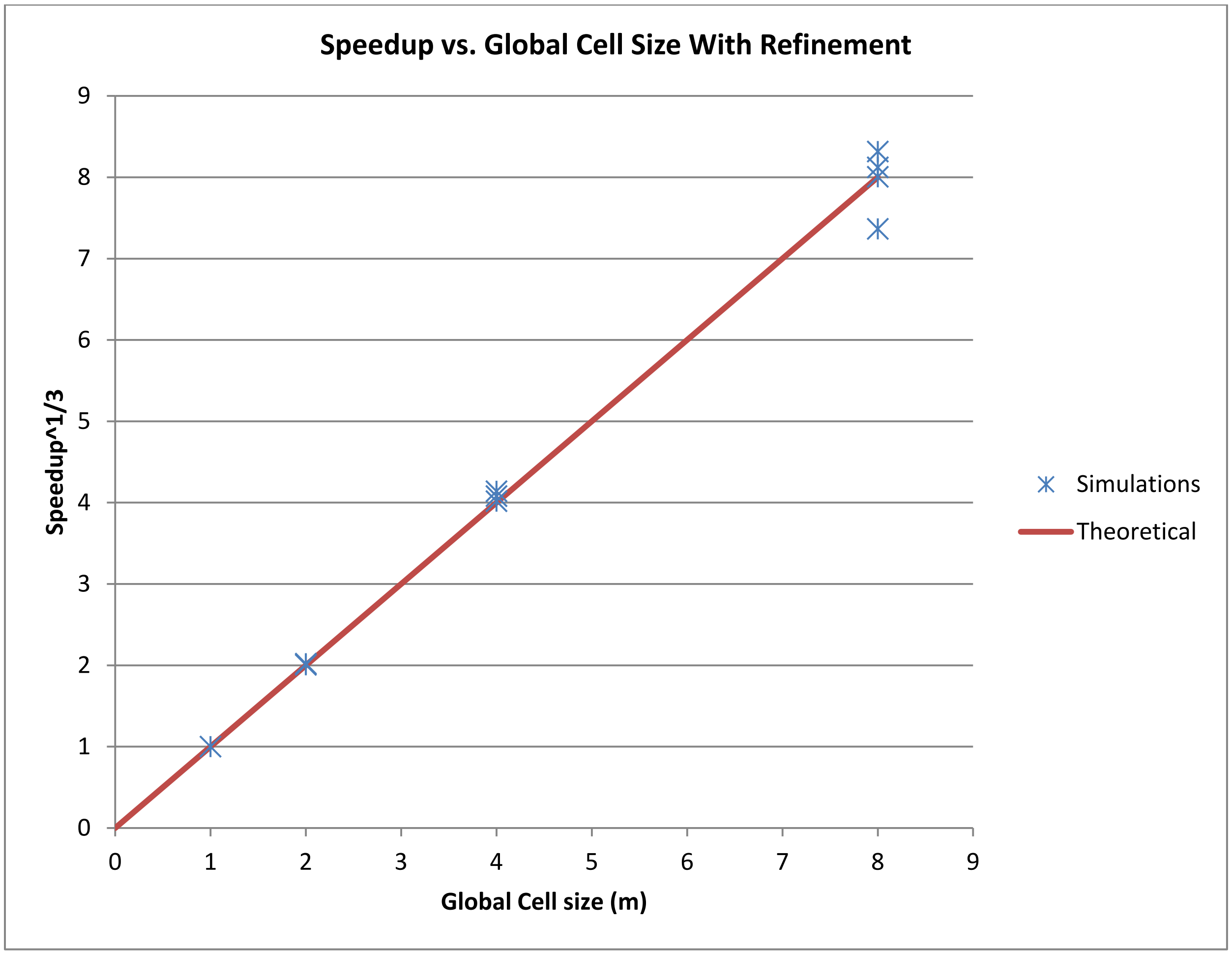


Figure V-16: Cube root of speedup vs global cell size for larger domain simulations. Data points are shown at cell sizes of 1m, 2m, 4m, and 8m. The line of theoretical maximum speedup is shown as the red line.

7. DISCUSSION

The goal of this chapter was to compare discharges through a dam breach using different orientations of an ideal reservoir and using different resolutions with local refinement. Orientations of 0 degrees, 20 degrees, and 45 degrees were each simulated using 1m cells with no refinement, 2m cells with no refinement, 2m cells with one level of refinement, 4m cells with no refinement, 4m cells with one level of refinement, 4m cells with two levels of refinement, 8m cells with no refinement, 8m cells with one level of refinement, 8m cells with two levels of refinement, and 8m cells with three levels of refinement. In addition, a larger domain and reservoir were simulated using the same dam and breach as above to get a more accurate portrayal of the CPU time required to finish the simulations. In the smaller domain cases, the results of 0 degree, 20 degree, and 45 degree rotations agreed with each other. With a given orientation, the results using larger global cell sizes refined to higher levels tended to disagree more with the control simulations. The CPU time required by simulations with larger global cell sizes tended to change with the cube root of the relative cell size. For simulations with a given global cell size, the ones with higher levels of refinement took longer to calculate than the ones with lower levels of refinement, but in all cases, they were faster than the next smaller level of global cell size.

There are two ways of thinking about local refinement. The first is when considering a given global mesh size, local refinement allows representing smaller features. Sometimes the data with the best available cell size is not enough to resolve the dam breach. In these cases, local refinement makes the simulation possible, regardless of computation speed. The other way of thinking is that given a size of feature to be modeled, the domain can be coarsened to some degree while maintaining the definition of that feature. This allows the same simulation to be run faster with local refinement. In the current work, local refinement is tested via the second way of thinking. This is acceptable because the domain outside the dam breach and reservoir are unimportant in the current cases. We assume that the domain can be coarsened without losing any important features. In real world dam break simulations, there will be a limit to the coarsening of the global cell size due to other features besides dam that need to be modeled. The issue of selecting an optimal cell size given these restraints is outside the scope of this thesis.

What can be determined from these results is that the discharges through the dam breach for various orientations and various levels of refinement match reasonably well with their respective control case. Ideally, since all simulations modeling the smaller domain were also modeling the same reservoir, dam, and breach, they should give the same results. However, differences in cell sizes and issues inherent to the numerical solver lead to some differences in the measured discharges. The differences between unrefined cases of different orientations are about as large as the differences between refinement levels in a given orientation. Therefore we can conclude that the computational model using local refinement gives results with the same level of accuracy as the original computational model without local refinement. In terms of accuracy, the quadtree method seems to give reasonable results up to three levels of refinement. Future work will determine if there is a limit where increasing the refinement level gives no improvement to the results. The test cases shown here demonstrate the local refinement technique and show that it gives reasonably accurate results.

It is expected that real world dam break simulations will require a much larger domain than the ones presented in this chapter. The speedup gained from using local refinement would be even more profound in these cases since the area to be refined would be proportionally smaller than the unrefined part of the domain. These cases would be expected to achieve close to the theoretical maximum speedup of 8-fold for each doubling of the global cell size. The best results in the simulations above come from refinement levels up to level 2. Future work will determine if simulations with further refinement with larger base cells follow the trends found here. If the accuracy of a selected refinement level is acceptable, then the speedup gained by coarsening the rest of the domain makes local refinement well worth it. With further work, the code could be optimized to take advantage of modern CPU architecture, such as more efficient data structures and cache access, vectorization, and parallel computation.

VI. CONCLUSIONS

The goal in this work is to present the reader with an introduction to the necessity of capable, stable, fast, reliable, and easy to set up dam break flood modeling, and to present an existing tool that has been upgraded to better fulfill these needs. With aging infrastructure and increases in populations in inundation zones, the consequences of a dam break flood can be a large and evolving threat. The problem is compounded by the large number of dams in the United States and the inadequate number and quality of emergency action plans. This shows the need for more research and better tools to address this problem. Recent advances in technology such as geographical information systems, collections of large data sets, and numerical methods, along with ever-improving hardware availability have made advanced models possible. The push is for more- more simulations with larger domains and faster results. This requires more advanced models that are easier to set up and give accurate results as quickly as possible.

The existing CCHE2D-FLOOD model contained numerous features designed to meet the goals of numerical modelers. However, the ease of using a basic DEM as the computational mesh limited it in the size of features it could model while still producing results quickly. In order to represent small features such as dam breach geometry, the modeler would have to reduce the size of the entire mesh, and only if the data was even available. In addition, the penalty of having a small cell size for the entire domain was required if the feature was to be modeled. While useful, this model showed a clear need for a remedy to this problem. A first step was to replace the older first-order upwinding numerical scheme with a more robust HLLC solver. This new solver has been verified to give correct results in a variety of test cases.

The efforts described in this manuscript were to address this problem by decoupling the global cell size from the size of features that could be resolved by using quadtree local mesh refinement. This method involves replacing one larger cell by four smaller ones with half the size in both x- and y-dimensions. These refinements can be done anywhere in the domain, and with only practical limits on refinement level. Specifically, the projection of a dam's crest line and width are used to define which cells need to be refined. The refinement level is determined by the size of the features in the definition of the dam's breach geometry. With special considerations to initial conditions, the resulting quadtree mesh is coupled with the basic, regular two-dimensional mesh and solved simultaneously. This allows the quadtree mesh to be generated only where needed while the regular mesh is kept at its original resolution. The methods, algorithms, and data structures for solving for the flow variables in time with the quadtree mesh have been presented.

Quadtree local refinement is useful because it enables the representation of small features without the need to refine the entire mesh. If the global mesh size is fixed, then local refinement allows the features to be represented at all where before they were not. Alternatively, local refinement allows features to be modeled at the original resolution while the rest of the domain is made coarser with larger cells. Each doubling of the global cell size has a theoretical speedup of eight, so this method allows much faster simulation where only a small area is required to have higher resolution.

The results of simulations utilizing local refinement are presented in Chapter 5. An ideal reservoir and dam with three different orientations were modeled using different cell sizes and with different levels of refinement. Even in the models using a coarse global cell size with three levels of refinement, the errors in peak discharge relative to the globally-refined cases were usually less than 10%. This is comparable to the differences between the peak discharges of the unrefined cases of different orientations. These results show that for the given simulations, using quadtree local mesh refinement for representing dam breach geometry gives results about as accurate as the model with global refinement.

An example of the benefits of using local refinement to decouple dam breach size form the global cell size are shown in Figure VI-1. The simulation CPU time required to simulate a reservoir and dam at the smallest global cell size without refinement is the unit time shown in the figure. Doubling the global cell size to 2 and refining one level takes about 1/8 as long to run. Quadrupling the global cell size to 4 reduces the computational cost by about another factor of 8, and so on. The larger the number of cells in the coarsened domain relative to the number of cells in the refined area, the closer the gains will approach to the theoretical.

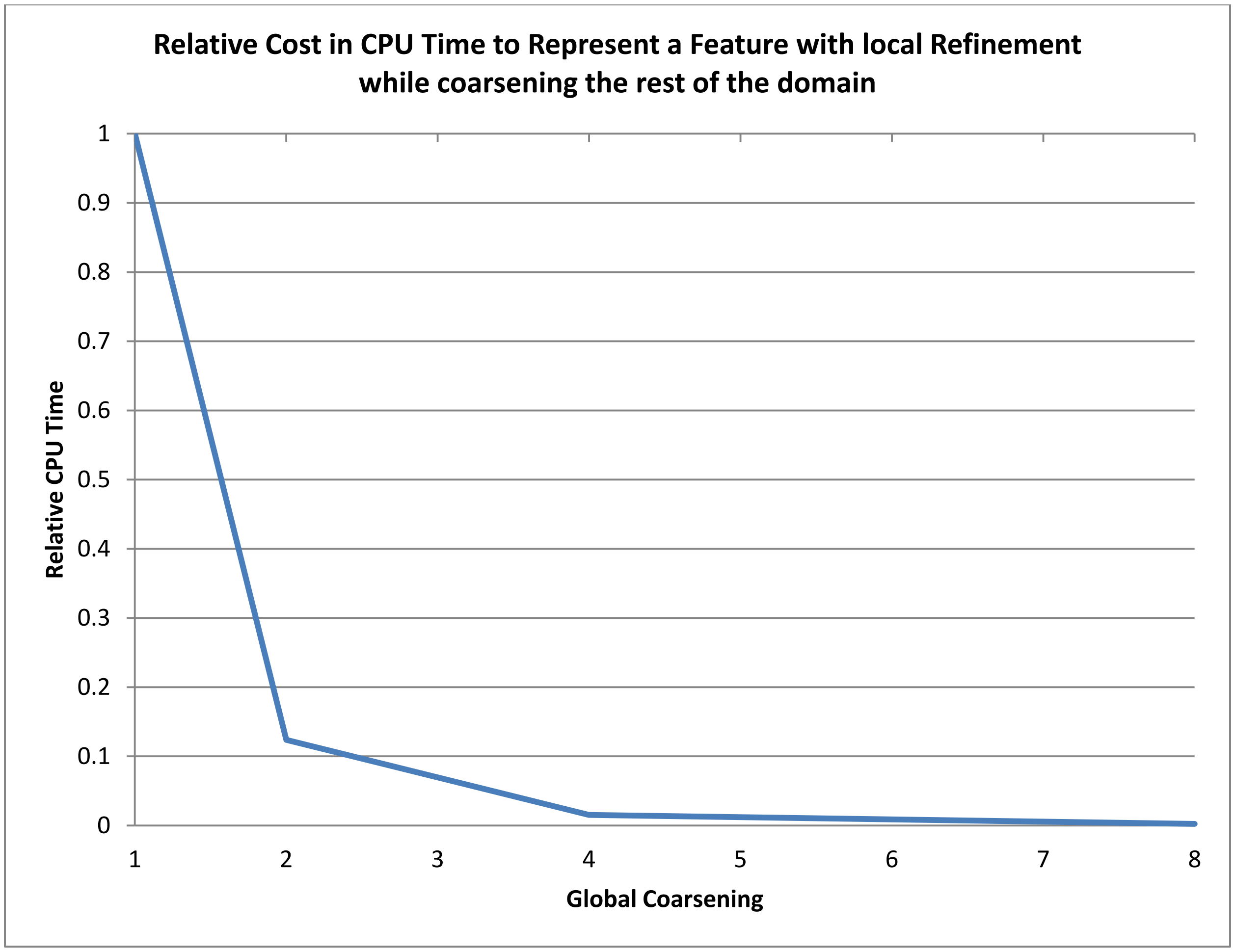


Figure VI-1: Relative Cost in CPU Time to Represent a Feature with local Refinement while coarsening the rest of the domain

Further enhancements can be made to the numerical model. Dam breach geometry is not the only feature that can be modeled with quadtree refinement. In fact, this method is useful for representing nearly any feature, such as areas of steep gradients, areas where greater output detail is required, etc. It is also possible that bed elevations of lower and higher resolution cells can come from different input sources. Future work can be done to include the definition of these features in the program. In addition, modifying the underlying data structures to use arrays instead of linked lists will allow the data to be indexed. This will simplify parallelization of the code and result in further speedups. Until then, the current implementation of quadtree local mesh refinement accomplishes the intended goals with good results. The model is featureful, robust, accurate, fast, and simple to set up. Quadtree local mesh refinement is another useful tool in helping the model meet the requirements of modern dam break problems.

Bibliography

Altinakar, M.S., Matheu, E.E, and McGrath, M.Z. 2009a. New Generation Modeling and Decision Support Tools for Studying Impacts of Dam Failures. Dam Safety 2009, Proc., ASDSO 2009 Annual Conference, Sept. 27-October 1, Hollywood, FL (CD-Rom). Association of State Dam Safety Officials, Lexington, Ky.

Dae-Hong Kim, Yong-Sik Cho, Yong-Kon Yi, Propagation and run-up of nearshore tsunamis with HLLC approximate Riemann solver, Ocean Engineering, Volume 34, Issues 8–9, June 2007, Pages 1164-1173, ISSN 0029-8018, 10.1016/j.oceaneng.2006.07.001.

Graf, W.H. and M.S. Altinakar (2002): Fluvial Hydraulics : Flow and transport in channels of simple geometry; J. Wiley & Sons, Ltd, Chichester, GB, 3rd Ed. (1st Ed. 1998, 2nd Ed. 2000).

Godunov, S.K. 1959. Finite difference methods for numerical computation of discontinuous solutions of the equations of fluid dynamics. Mathematicheskii Sbornik, 47 (3), 271-306.

Harten, A., Lax, P., and van Leer, B. 1983. On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. SIAM Rev., 25, pp. 35–61.

LeVeque, R.J. (1999): "Numerical Methods for Conservation Laws"; Birkhäuser Verlag, Basel, Switzerland.

LeVeque, R.J. (2002): "Finite Volume Methods for Hyperbolic Problems"; Cambridge Texts in Applied Mathematics Series, #31, Cambridge University Press

McGrath, M.Z., Altinakar, M.S., and Miglio, E. (2010): "Representation of Dam-Breach Geometry on a Regular 2-D Mesh Using Quadtree Local Mesh Refinement "; Proceedings of the International Conference on Fluvial Hydraulics (River Flow 2010), September 8-10, Braunschweig, Germany.

Ying, X., Wang, S., Khan, A. 2003. Numerical Simulation of Flood Inundation Due To Dam and Levee Breach. Proceeding of ASCE World Water & Environmental Resources Congress 2003 (CD-ROM), Philadelphia, USA, June.

Ying, X., Khan, A.A., and Wang, S.S.Y. (2004): "An Upwind Conservative Scheme for Saint Venant Equations"; ASCE-JHE 130(10), pp. 977-987.

Qiuhua Liang, Alistair G.L. Borthwick, Adaptive quadtree simulation of shallow flows with wet–dry fronts over complex topography, Computers & Fluids, Volume 38, Issue 2, February 2009, Pages 221-234, ISSN 0045-7930, 10.1016/j.compfluid.2008.02.008.

Raphael Finkel and J.L. Bentley (1974). "Quad Trees: A Data Structure for Retrieval on Composite Keys". Acta Informatica 4 (1): 1–9.

Roe, P.L. 1981. Approximate Riemann solvers, parameter vectors, and difference schemes. Journal of Computational Physics. 43: 357-372.

Toro, E.F. (1999): "Riemann Solvers and Numerical Methods for Fluid Dynamics"; 2nd Ed., Springer-Verlag, Berlin, Germany.

Zheng, H., Shu, C., Chew, Y. 2008. An object-oriented and quadrilateral-mesh based solution adaptive algorithm for compressible multi-fluid flows. Journal of Computational Physics, v.227 n.14, pp. 6895-6921, July.

VITA

Education:

Master of Science in Engineering Science Anticipated in Spring 2013

Emphasis: Computational Hydroscience

University of Mississippi

National Center for Computational Hydroscience and Engineering

Advisor: Dr. Mustafa Altinakar

Bachelor of Science in Mechanical Engineering 2007

University of Mississippi

Awards and Honors:

- Phi Kappa Phi
- Tau Beta Pi Engineering Honor Society
- National Society of Collegiate Scholars
- University Network Summit 2009 student abstract competition winner
- Undergraduate Dean's List, cum laude with grade point average 3.71

- Senior Design competition winner

Employment:

National Center for Computational Hydroscience and Engineering University, MS

Graduate Research Assistant 2007-Current

Responsible for development and improvement of a two dimensional dam-break flood code called CCH2D-FLOOD, teaching short courses, development of highly-parallel flood model using GPGPU technology

Publications:

Altinakar, M.S., McGrath, M.Z., Oner, Y., and Ger, M. (2013). "Simulation of Supercritical Flow in an Open Channel Bend Using Cut-Cell Boundary Method". Accepted for publication in: Proceedings of the 6th International Perspective on Water Resources and Environment (IPWE 2013), Jan 7-9, 2013, Izmir, Turkey.

Altinakar, M.S., McGrath, M.Z., Ramalingam, V.P., Singh, J., Ding, Y., and Inman, M.(2012): "Advances in 2D Flood Modeling and Mapping Technology for Dam and Levee Safety"; Proceedings of the 10th International Congress on Advances in Civil Engineering (ACE 2012), October 17-19, 2012, Ankara, Turkey.

Altinakar, M.S., McGrath, M.Z., Matheu, E.E., Ramalingam, V.P., Seda-Sanabria, Y. Breitkreutz, W., Oktay, S., Zou, J.Z., Yezierski, M. (2012): "Validation of Automated Dam-Break Flood Simulation and Assessment of Computational Performance"; Proceedings of ASDSO Dam Safety 2012 Annual Conference, September 16-20, Denver, CO.

Altinakar, M.S., Matheu, E., McGrath, M.Z., Ramalingam, V.P., and Zou J.Z. (2012): "Development and Implementation of Web-Based Dam-Break Flood Inundation Analysis Capability"; Proceedings of the 2012 U.S. Society on Dams (USSD) Conference, 23-27 April 2012, New Orleans, LA.

Altinakar, M.S. and McGrath, M.Z. (2012): "Parallelized Two-Dimensional Dam-Break Flood Analysis with Dynamic Data Structures", ASCE-EWRI, 2012 World Environmental & Water Resources Congress, May 20-24, 2012, Albuquerque, NM.

Kuiry, S.N., Ding, Y., McGrath, M.Z. and Altinakar, M.S. (2012): "A Two Dimensional Finite Volume Model for Tide and Storm Surge Predictions", ASCE-EWRI, 2012 World Environmental & Water Resources Congress, May 20-24, 2012, Albuquerque, NM.

Altinakar, M., Matheu, E., Ramalingam, V., McGrath, M., and Zou J. (2011): "Web-Based Implementation of Fast Dam-Break Flood Modeling Capabilities"; Proceedings of ASDSO Dam Safety 2011 Annual Conference, September 25-29, 2011, Washington, D.C.

McGrath, M.Z., Altinakar, M.S., and Miglio, E. (2010): "Representation of Dam-Breach Geometry on a Regular 2-D Mesh Using Quadtree Local Mesh Refinement "; Proceedings of the International Conference on Fluvial Hydraulics (River Flow 2010), September 8-10, Braunschweig, Germany.

Altinakar, M.S., McGrath, M.Z., Ramalingam, V.P. , and Omari, H. (2010): "2D Modeling of Big Bay Dam Failure in Mississippi: Comparison with Field Data and 1D Model Results"; Proceedings of the International Conference on Fluvial Hydraulics (River Flow 2010), September 8-10, Braunschweig, Germany.

Altinakar, M.S., McGrath, M.Z. and Ramalingam, V.P. (2010): "An Enhanced Two-Dimensional Numerical Model for Simulating Floods Due to Dam and Levee Break/Breaching"; Proceedings of the 9th International Conference on Hydroscience and Engineering (ICHE 2010), August 2-5, Chennai, India.

Altinakar, M.S., Matheu, E. E., and McGrath, M.Z. (2009): "New Generation Modeling and Decision Support Tools for Studying Impacts of Dam Failures"; Proceedings of ASDSO Dam Safety 2009 Annual Conference, September 27-October 1, 2009, Hollywood, FL.

Altinakar, M.S., McGrath, M.Z., Ozeren, Y., and Miglio, E. (2009): "Two-Sided Cut-Cell Boundary Method for Simulating Linear Terrain features and 1D Stream Flows on a 2D Rectangular Mesh"; Proceedings of the 33rd International IAHR Biennial Congress, August 9-14, 2009, Vancouver, Canada.

Altinakar, M.S., McGrath, M.Z., Ozeren, Y., and Miglio, E. (2009): "Representation of Linear Terrain Features in a 2D Flood Model with Regular Cartesian Mesh"; Proceedings of the 2009 World Environmental & Water Resources (EWRI) Congress, ASCE, May 16-23, 2009, Kansas City, MO.

Altinakar, M.S., McGrath, M.Z., Ozeren, Y., and Omari, H. (2008): "Modeling and Risk Analysis for Floods due to Failure of Water Control Infrastructures"; Proceedings of the International Symposium on Uncertainties in Hydrologic and Hydraulic Modeling, Oct 15-20, 2008, Montreal, Canada.

Altinakar, M.S., McGrath, M.Z., Fijołek, E.K., and Miglio, E. (2008): "Risk and Vulnerability Studies for Water Infrastructures Using a GIS-Based Decision Support System with 2D Numerical Flood Modeling"; Proceedings of the 8th International Conference on Computational Hydroscience and Engineering, (ICHE 2008), September 8-12, 2008, Nagoya, Japan.

Conferences Attended:

- International Conference on Fluvial Hydraulics 2010, Braunschweig, Germany.

Presentation titled "Representation of Dam-Breach Geometry on a Regular 2-D Mesh Using Quadtree Local Mesh Refinement."

- University Network Summit 2011, Washington, D.C. Poster presentation titled "An Enhanced 2D Numerical Model for Simulating Dam/Levee Break Floods."
- Mid-South Annual Engineering and Sciences Conference 2011, Memphis, TN. Presentation titled "Numerical Modeling of Big Bay Dam Failure."
- Association of State Dam Safety Officials Dam Safety 2011, Washington, D.C. Presentation titled "Web-based Implementation of Fast Dam-Break Modeling Capabilities "
- Association of State Dam Safety Officials Dam Safety 2012, Denver, CO.
- International Conference on Hydroscience and Engineering 2012, Orlando, FL. Presentation titled "Parameter-Based Estimation of Unknown Reservoir Bathymetry for 2D Dam-Break Modeling."

Short Courses Instructed:

- Vicksburg, MS. May 3-4, 2010. Course titled "Two-Day Short Course on DSS-WISE Software. Two-Dimensional Dam/Levee Break Flood Modeling, Flood Danger Mapping and Consequence Analysis."
- Association of State Dam Safety Officials Dam Safety 2012 Denver, CO. September 20, 2012. Course titled "Dam Break Flood Simulation: Doing It Faster and Simpler!"

- Vicksburg, MS. April 14-16, 2013. Course Titled, " Dam Break Flood Simulation: Doing It Faster and Simpler!""

Community and Hobbies:

- Personal interest in lifelong learning, especially related to technology
- Linux, free and open source software
- Member of Saint Anthony Hall social fraternity including community service activities