River Flow 2010 - Dittrich, Koll, Aberle & Geisenhainer (eds) - © 2010 Bundesanstalt für Wasserbau ISBN 978-3-939230-00-7

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

M. McGrath & M. S. Altinakar
National Center for Computational Hydroscience and Engineering, University of Mississippi, Oxford MS, USA

E. Miglio
MOX, Dept. of Mathematics "F. Brioschi" Politecnico di Milano. Piazza Leonardo da Vinci, 32, 20133 Milano, Italy

ABSTRACT: A 2D first order upwinding finite-volume scheme is used on a regular orthogonal mesh in a numerical model to simulate dam break and dam breach flow from a reservoir. The discharge through the time-varying breach geometry is crucial to the accuracy of the resulting flood wave, yet the method of representing the breach evolution by changing the bottom elevation of individual cells within the vicinity of the dam makes the breach hydrograph and peak discharge highly dependent on the mesh size and dam orientation. To increase accuracy and reduce computational burden associated with refining the entire mesh, a quadtree local mesh refinement technique was used to better model the dam breach geometry. Simulation results from the model using a combined regular and quadtree mesh showed that a coarse mesh with local refinement can yield a good approximation of the discharge hydrograph obtained using a globally-refined mesh, with significant savings in computational time.

Keywords: Dam break, Flood, Quadtree, Mesh Refinement, Breach hydrograph

1 INTRODUCTION

1.1 *Purpose*

Two-dimensional numerical models are commonly used in solving shallow water equations for a problem of interest, such as a dam break releasing water into a floodplain. Numerical models solve governing equations for the variables of interest after discretizing the problem domain. In the case of a finite volume model for dam break, the variables of interest are flow depth and flow velocities, and the domain is discretized into cells. Each cell in the domain corresponds to a real-world area where the flow variables can be used for consequence analysis and emergency management planning operations stemming from damage caused by the flowing water. These analyses necessitate accurate results from the numerical model.

One of the easiest ways to discretize a domain is to divide it into equal-sized square cells, resulting in a regular orthogonal mesh. This type of mesh requires little preparation but has some drawbacks which will be discussed.

In the case of dam break flood modeling, the problem usually involves some reservoir or source of water, the dam, and terrain downstream of the dam over which the resulting flood propagates. One possibility is to separately compute the discharge through the dam breach using a dedicated program. Then the resulting hydrograph is used as a boundary condition in the 2D model.

If the bathymetry of the reservoir is known, one can also consider a computational domain which includes the reservoir, the dam, and the downstream area. In this case, the cells occupying the area of the dam are represented as topography and, thus, have a bottom elevation equal to the crest elevation of the dam. The gradual breaching of the dam is then represented by manipulating the elevation of the cells representing the dam according to a predefined schedule, which follows the evolution of the breach geometry.

The flow of water through the breach is of paramount importance to the resulting downstream flood. It is therefore important that it must be modeled as accurately as possible. When using a regular mesh, accurate representation of breach evolution can be a problem. Typical cell sizes for dam-break simulations using a regular mesh may be 50 m or larger. The evolution of breach geometry cannot be accurately represented with such large cell sizes.

It is of course possible to use a regular mesh with a higher resolution, but that would require smaller time steps and, thus, longer computational times due to the CFL condition.

Ideally, the breach discharge should be independent of the mesh size, and the evolution of the breach geometry should be correctly simulated even when using a coarse regular mesh dictated by other considerations, such as the size of the computational area.

Since refinement of the entire mesh imposes significant increases in computational burden, and because the dam breach progression requires small features to be modeled accurately, the capability to locally refine the mesh in the vicinity of the dam is a worthwhile endeavor. In the present study, this was accomplished using a version of quadtree mesh refinement.

1.2 *Quadtree method*

Quadtree refers to a hierarchical organization of data in which a single "parent" element is associated with four equal-size "child" elements that, when taken together, spatially represent the original parent element completely. Each child element may, in turn, have child elements, and so on, up to any practical level of refinement imposed by the modeler. The connectivity between parent and child is maintained, so communication is possible between different levels of the data structure.

In the current model, the quadtree structure is applied to cells in the two-dimensional mesh. Once the entire problem domain is discretized into cells with the original size, the ones near the areas of interest (dams) are refined until the smallest child cells are of the desired size. Refinement refers to the process of creating child cells within a given cell.

The quadtree structure has been implemented in a pre-existing verified and validated two-dimensional numerical code called CCHE2D-FLOOD (Altinakar et al. 2009a, b, and c). This uses an explicit, conservative, finite-volume, first-order upwinding scheme on a regular square mesh. A test case involving a dam breach was modeled with mesh sizes of 5m, 10m, 20m, and 40m. For the three largest mesh sizes, the breach was modeled first with no refinement and then with local refinement to a level of 5m near the dam. No refinement was made to the 5m mesh. Section 4 of this paper describes the results of these tests.

2 DESCRIPTION OF EXISTING 2D NUMERICAL CODE CCHE2D-FLOOD

A description of this model can be found in Ying (2003) and Altinakar et al. (2009a).

2.1 *Governing equations*

The conservative form of the two-dimensional shallow water equations is written as

$$\frac{\partial \mathbf{U}}{\partial t}+\frac{\partial \mathbf{F}}{\partial x}+\frac{\partial \mathbf{G}}{\partial y}=\mathbf{S} \tag{1}$$

where $\mathbf{U}$, $\mathbf{F}(\mathbf{U})$, $\mathbf{G}(\mathbf{U})$, and $\mathbf{S}$ represent the vector of conserved variables, the vector of fluxes in the x-direction, the vector of fluxes in the y-direction, and the vector of sources, respectively:

$$\mathbf{U}=\begin{bmatrix} h \\ hu \\ hv \end{bmatrix} \mathbf{F}=\begin{bmatrix} hu \\ huu \\ huv \end{bmatrix} \mathbf{G}=\begin{bmatrix} hv \\ huv \\ hvv \end{bmatrix} \tag{2}$$

$$\mathbf{S}=\begin{bmatrix} 0 \\ -gh\dfrac{\partial Z}{\partial x}-g\dfrac{u\sqrt{u^2+v^2}}{C^2} \\ -gh\dfrac{\partial Z}{\partial y}-g\dfrac{v\sqrt{u^2+v^2}}{C^2} \end{bmatrix} \tag{3}$$

where h = water depth, u = fluid velocity in the x-direction, v = fluid velocity in the y-direction, g = acceleration of gravity, Z = water surface elevation, and C =Chezy's coefficient. Integrating (1) over a cell and applying Green's theorem yields:

$$\begin{aligned}\mathbf{U}_{ij}^{n+1}=\mathbf{U}_{ij}^{n}&-\frac{\Delta t}{\Delta x_i}\left(\mathbf{F}_{i+1/2,j}-\mathbf{F}_{i-1/2,j}\right)\\&-\frac{\Delta t}{\Delta y_j}\left(\mathbf{G}_{i,j+1/2}-\mathbf{G}_{i,j-1/2}\right)\\&+\Delta t\mathbf{S}_{ij}\end{aligned} \tag{4}$$

where $F_{i+1/2,\,j}$, $F_{i-1/2,\,j}$, $G_{i,\,j+1/2}$, and $G_{i,\,j-1/2}$ are the fluxes at a cell's right, left, top, and bottom interfaces, respectively, and Δt is the time step value. A simple upwind method is used to calculate the intercell fluxes in x and y directions:

$$\mathbf{F}_{i+1/2}=\begin{bmatrix} (hu)_{i+k} \\ (huu)_{i+k} \\ (huv)_{i+k} \end{bmatrix} \mathbf{G}_{j+1/2}=\begin{bmatrix} (hv)_{j+k} \\ (huv)_{j+k} \\ (hvv)_{j+k} \end{bmatrix} \tag{5}$$

with

$$k = \begin{cases} 0 & if\ Q \geq 0 \\ 1 & if\ Q \leq 0 \end{cases} \quad (6)$$

The time step for this explicit scheme is governed by the CFL condition:

$$N_{CFL} = MAX\left[\frac{\Delta t}{\Delta x}(|u| + \sqrt{gh}), \frac{\Delta t}{\Delta y}(|v| + \sqrt{gh})\right] \leq 1 \quad (7)$$

2.2 *Wetting and drying treatment*

To simulate the flood waves propagating over initially dry land, the model must be able to deal with wetting and drying processes. The present model first solves for water depths (*h*) and then for discharges (*hu* and *hv*) from which the velocities (*u* and *v*) are calculated. The wetting and drying process is implemented by checking if the water depth computed is less than some minimal threshold. If the depth is beneath this threshold, the cell's velocities in both directions are set to zero. Once the cell's depth reaches the threshold value, the velocities are calculated as normal

2.3 *Procedure*

2.3.1 *Overview*

The existing model is quite flexible in that it reads bottom elevation values from a DEM prepared by common GIS software and implements dams and reservoirs through simple text input files that are easily edited to set up a simulation. The cells within a user-given area representing the dam are identified and set to the proper dam crest elevation, and the reservoirs are filled with water to the proper level. The user can define points at which to sample data at regular intervals, and can also define lines across which flow discharge is measured. Once all data have been read, the program calculates a time step based on the maximum signal speed of the water with regard to eq. (7) and then calculates fluxes for all cell interfaces in the domain according to eqs. (5-6). Using these fluxes, the program updates the flow values of depth and velocity in each cell with eq. (4). A new time step is then calculated from the updated water depth and velocity values. This procedure repeats until a desired simulation time has been reached.

2.3.2 *Dam breach progression*

A dam is defined by two points, a crest elevation, a width, and a breach geometry profile. The dam width is applied to the line from the first point to the second point, which forms a rectangular area. All cells whose centers lie inside this rectangle are initially given a bottom elevation equal to the dam crest elevation. The cells inside this rectangular area have their bottom elevations updated at each time step according to a breach geometry profile. A profile is a series of elevation changes along the length of the dam that define a "snapshot" of the geometry of the breach at a certain point in time. The geometry is interpolated both in space (using a single projected cell-center point between the user-given profile points) and in time (between different profile snapshots) for each cell under the dam.

3 IMPLEMENTATION OF QUADTREE METHOD

3.1 *Description of method*

3.1.1 *Data structures*

The data structures for the quadtree mesh were implemented similarly to the method described by Zheng (2008). Each cell in the quadtree data structure has an associated level of refinement. Cells in level 1 are the same size as the cells in the regular 2D mesh, while cells in each level greater than 1 have dimensions exactly half as long as the cells one level lower. A cell in the quadtree data structure has four nodes at its corners and four edges which can be shared by neighboring cells. Corner nodes are only used to keep position information and are not used in the computation. The four edges of a cell are of the same level of refinement as the cell itself. The procedure for refining a cell is as follows. A new node common to the four child cells at the center of the parent cell is added to the nodes list. Four new child cells are added to the cell list with one level higher than the parent cell. Four new interior edges with the same level as the child cells are added to the edge list. Each of these edges is shared by two of the new child cells. Then each of the four existing edges of the parent cell is checked to see if it already has child edges. If so, then the new child cells identify with the proper edge, and if not, the edge is also refined by one level. This is done to ensure that there is no duplication of edges, nodes, or cells, and that connectivity between parent and child exists at all times for both cells and edges. Once this is done, all four adjacent cells are checked to make sure that the difference in level of refinement does not exceed one, as the quadtree data structure requires that a cell have no more than two cells adjacent to it on a given side. The procedure of cell refinement is recursive, allowing this rule to be verified in neighbors of each refined cell in succession.

The quadtree method was implemented by constructing separate linked lists, one for cells of each level, one for edges, and one for nodes. A cell is treated as a record in the cell list with several fields including flow variables, position information, parent-child connectivity, and the connections with the four corresponding edges. Connectivity between parent and child cells, between parent and child edges, and between cells and edges is maintained with pointers. Cells of different levels are kept in separate lists to increase speed of computation. Figure 1 depicts the hierarchical structure of the data, with cells labeled C1 in the list of cells with refinement level 1, ones labeled C2 in the list with refinement level 2, and so on. A cell with no child cells is called a leaf cell, while a cell with child cells is called a stem cell. In Figure 1, cells C1* and C2* are stem cells and the rest are leaf cells.

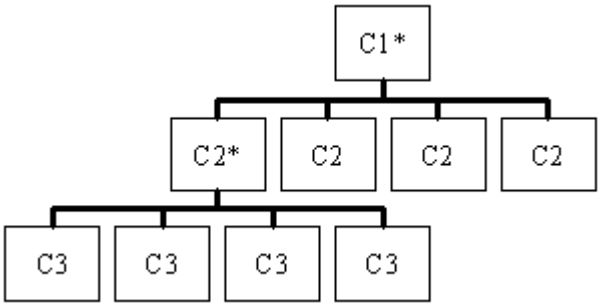

Figure 1. Depiction of hierarchy of quadtree data structure for cells

Edges are kept in a list that includes records of parent-child connectivity, flux values, and connectivity to the two adjacent cells. An edge with no child edges is called a leaf edge, while an edge with child edges is called a stem edge. In Figure 2, edges E1* ad E2* are stem edges and the rest are leaf edges.

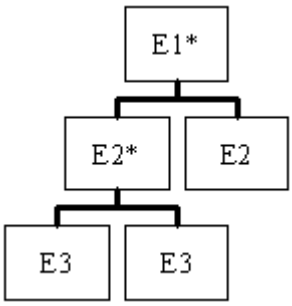

Figure 2. Depiction of hierarchy of binary tree data structure for edges.

Figure 3 shows the hierarchy of refinement. An edge of level E1 is associated with a cell in level C1. It can be seen that in the refined cell C1*, edge E1* will have two child edges in level E2. When a cell or edge is refined, the parent record is kept to maintain connectivity with the other elements of its level. It can be seen in Figure 3 that when all levels are stacked together, some cells and edges are collocated in space, but the computational procedure accounts for this.

3.1.2 *Computation*

The same numerical scheme from the original 2D model is applied to the quadtree model with some differences. Calculations are done to update the edge fluxes and cells with the highest level of

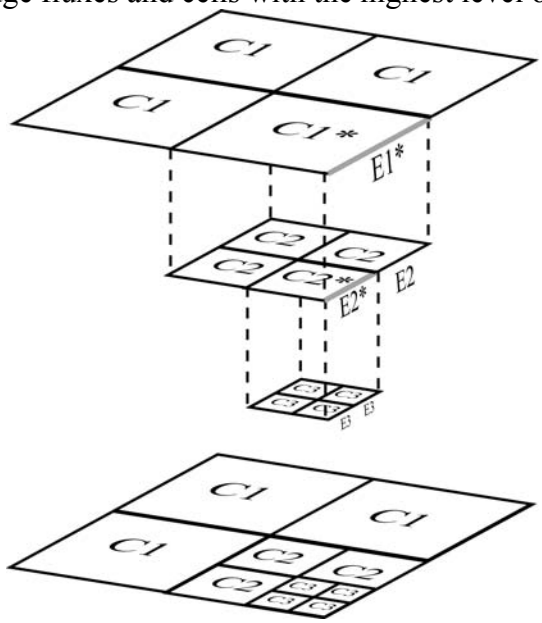

Figure 3. Depiction of various levels of refinement for cells.

refinement, and then progressively toward the edges and cells with lower levels of refinement. For leaf edges, the fluxes are calculated according to eqs. (5-6). For stem edges, the fluxes are calculated by summing the two child edge flux values. Once all edge fluxes have been calculated for the current level, flow variables at the cell centers are updated. If a cell is a leaf cell, it is updated according to eq. (4). If a cell is a stem cell, then its flow variables h, Z, u, and v are taken from the average of the child cells which are not dry. For instance, a cell with one dry child cell would take the average of the flow values from the remaining three. This prevents higher bottom elevations in dry cells from causing problems. Since a cell at a higher level of refinement is exactly half as large as its parent cell, the time step for each level is also half the value of the parent cell in order to maintain the same CFL value. This is represented as the following, where n is the level of the cell starting with level 1:

$$\Delta t_n = \frac{\Delta t_1}{2^{n-1}} \tag{8}$$

The calculation procedure is a nested loop where the smallest cells are calculated in the inner-most loop. To calculate any level, all the levels below it must first be calculated with its own time step.

Essentially each level is calculated and updated twice before moving to the next level.

The water surface slope source term seen in eq. (3) is applied by judging the upwind direction of the flow of water, which comes from looking at the flow direction in the cell's neighbors. In the case of the regular 2D mesh, a cell has only one neighbor on each side, and this term is easily calculated. However, with the quadtree mesh, a cell can have two neighbors on a side, so the water surface elevation used for the slope is the average of those two neighbor cells.

At the beginning of the calculation of the quadtree mesh, the flow variables of the level-1 leaf cells are copied from the regular 2D mesh. These cells exist in the quadtree mesh precisely for connectivity purposes, and their flow values are calculated only in the regular 2D mesh calculation. After the quadtree calculations have finished, the newly-computed flux values on the edges of these level-1 connection cells are copied back into the regular 2D mesh arrays. This way, the cell-center flow values in both regular and quadtree meshes are calculated using the current time step's fluxes, and the fluxes for the next time step are calculated using the current time step's cell-center flow values.

Initially it was assumed that the refinement process only needed to obey the rule requiring adjacent cells to differ by no more than one level. However, if a cell was adjacent to a mesh size change on both top and bottom, or on both left and right sides like in the top image of Figure 4, it led to instabilities stemming from the calculation of the water surface slope source term. The problem was diminished by also requiring that the difference between the maximum level and minimum level of refinement in all neighbors of a cell be less than or equal to one, as shown in the bottom image of Figure 4.

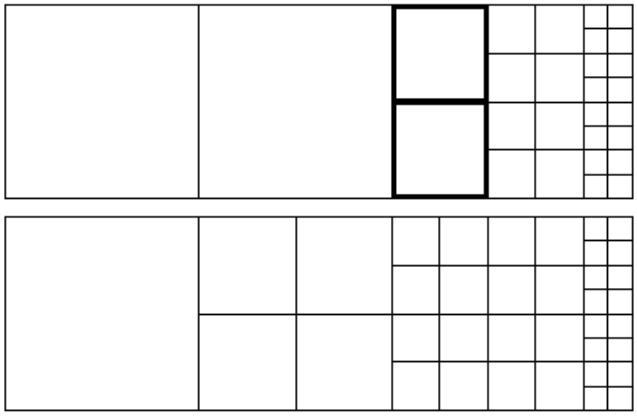

Figure 4. Top: Refinement level changes too rapidly for emphasized cells. Bottom: Improved mesh refinement.

3.2 *Application to dam cells*

The quadtree mesh refinement procedure is applied once at the beginning of a simulation. The needed level of refinement is determined by calculating the smallest distance between points in all profiles (above some minimal threshold) and equating that to the minimal cell size required to resolve it. This level of refinement is applied to the entire dam as a whole. Level-1 cells that are identified as being inside the dam are created within the quadtree data structure and then refined one level. The resulting child cells are then checked for being inside or outside the dam, and the procedure continues refining the new cells up to the needed level. The bottom elevation of any cell found to be inside the dam is initialized to the level of the dam's crest for the initial time step. It is possible that once a cell is refined, some of the child cells are inside the dam and some are outside. This refinement procedure is recursive, meaning that when refining a given cell, a neighboring cell is also refined if it differs by more than one level. Figure 5 shows the results of the refinement process for a typical dam. The figure on the left shows the level-1 cells identified as being underneath a section of a dam. The figure on the right shows the results after refining these cells to level 4.

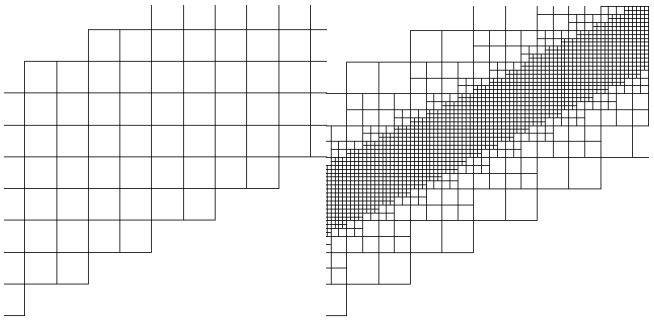

Figure 5. Cells underneath a dam identified (left) and refined to level 4 (right).

4 TEST CASES

4.1 *Simulation setup*

The model with quadtree refinement was validated against the existing model using a test case from Chauhan et al. (2008). The computational domain was a 1,000 m wide and 1,680 m long area with a flat bottom (see Figure 6). A 1,000 m wide and 1,280 m long portion of the computational domain was used to model a 19.35 m deep reservoir having a storage volume of 24,768,000 m^3 as in the test case of Chauhan et al. (2008). The remaining area downstream of this dam was initially dry. The dam had an 80 m wide section

in the middle that was modeled with the linearly progressing breaching sequence shown in Figure 7. The breaching started right at the beginning of each simulation. It took 2,520 s (42 min) to attain the final trapezoidal breach geometry which had a bottom width of 46 m and a top width of 80 m (side slopes approximately 0.9:1 horizontal:vertical).

The reservoir was allowed to drain through the breach into the downstream basin and freely exit the domain. The test case was first simulated using regular mesh sizes of 5 m, 10 m, 20 m, and 40 m without any refinement. Then a new series of simulations was performed with the same regular mesh sizes but with quadtree local refinement of the dam. Each simulation was carried out for 3,000 seconds, which was long enough to capture the falling end of the hydrograph after the breach reached its full size.

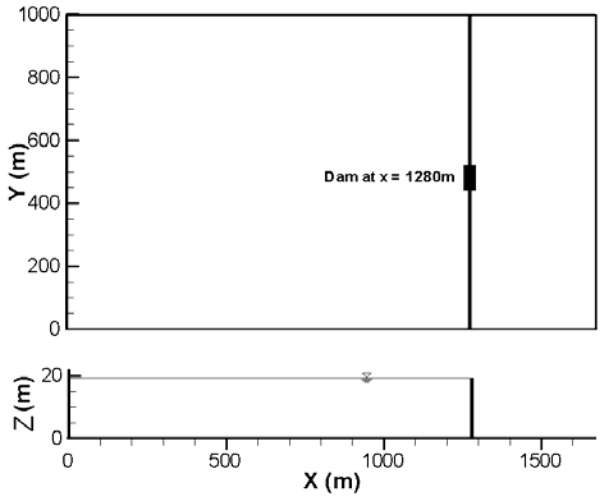

Figure 6. Representation of domain with dam at x=1280 m.

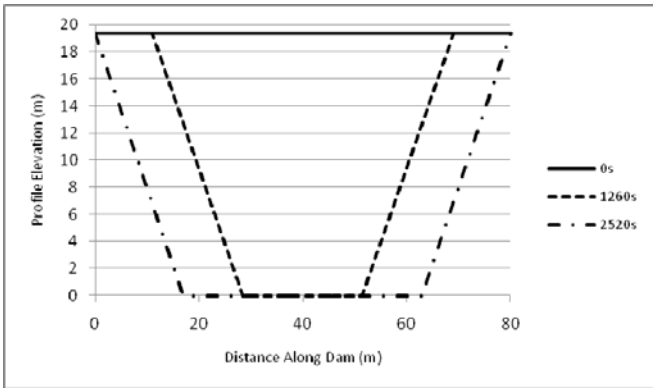

Figure 7. Dam breach geometry profiles.

4.2 *Results*

The simulations without mesh refinement yielded the discharge hydrographs shown in Figure 8. Based on the assumption that finer mesh resolutions are better able to capture the breach geometry sequence than the coarser meshes, the discharge hydrograph using the 5 m mesh was considered to be the most accurate, and other simulations were compared to it. Figure 8 clearly shows that the breach discharge hydrograph depends on the cell size used to represent the breach. The 80 m breach is represented with just two adjacent cells in the 40 m mesh, and this causes the peak discharge to be overestimated by more than 25% compared to the 5 m mesh hydrograph.

It is interesting to note that the differences between the results of the 10 m mesh and the 5 m mesh are small, which gives the notion that further refinement would yield little increase in accuracy.

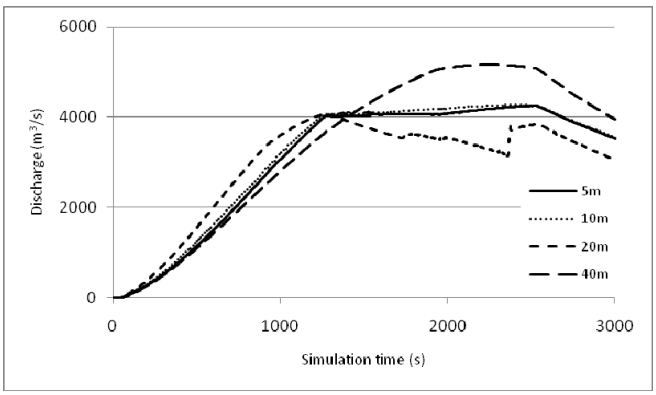

Figure 8. Dam breach discharge hydrographs for various mesh resolutions with no refinement.

A new series of simulations was also carried out using the three largest mesh sizes, but locally refining the mesh in the area of the breach using the quadtree method. The 40 m cells were refined three levels, 20 m cells were refined two levels, and 10 m cells were refined one level. In each case the smallest cells in the quadtree mesh were 5 m square. The discharge hydrographs for these three simulations with quadtree mesh refinement are plotted in Figure 9 together with the hydrograph simulated with the 5 m regular mesh over the entire computational domain.

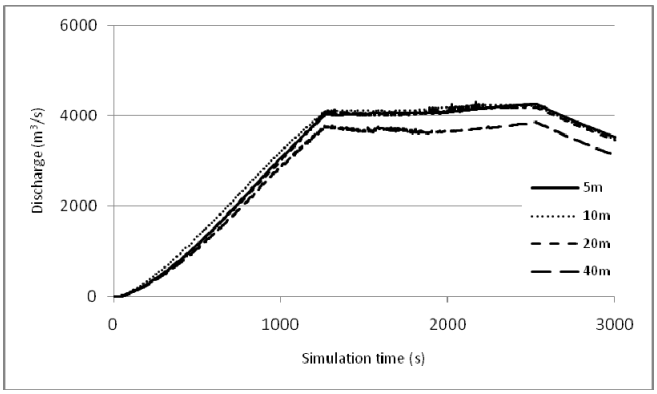

Figure 9. Dam breach discharge hydrographs for various mesh resolutions with refinement near the dam.

It is evident that the local quadtree refinement of the mesh at the breach area allows a computation

with larger mesh sizes to better approximate the hydrograph obtained from a 5 m unrefined mesh, which was assumed to be best available estimation of the breach hydrograph.

Figure 10 shows the differences in discharge values for each mesh size with refinement and without refinement along with the 5m mesh hydrograph plotted for reference. The 10 m mesh simulation matched the 5 m mesh simulation quite well, with and without refinement. This shows that the breach appears to be adequately modeled with 10 m cells.

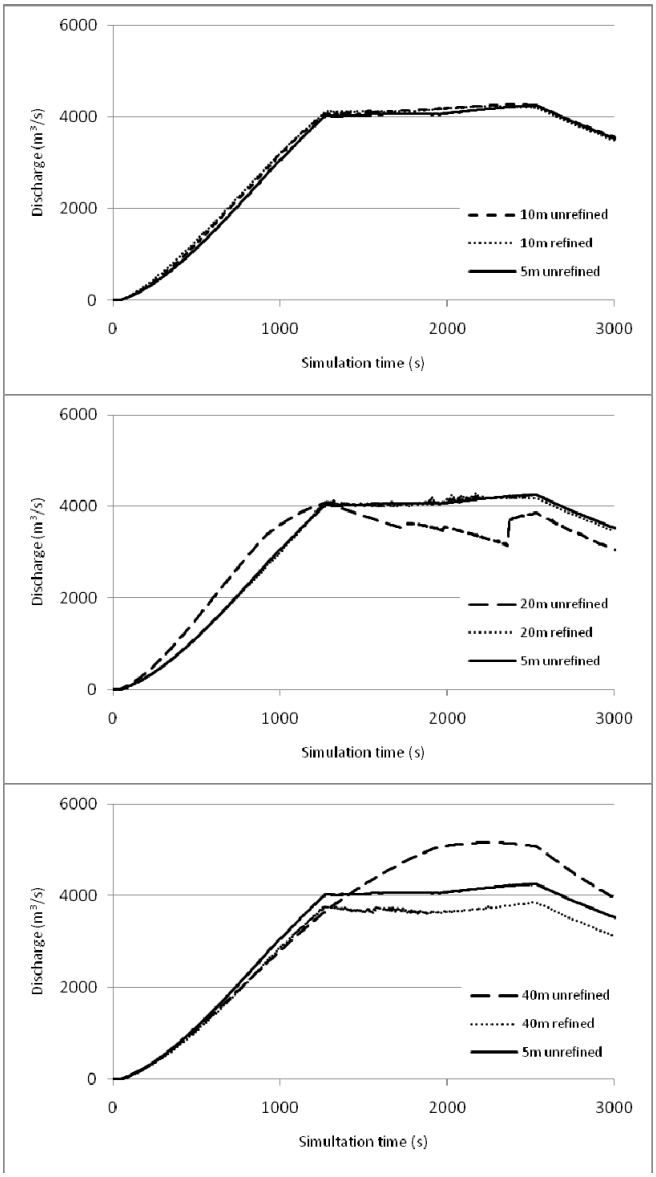

Figure 10. Breach discharge hydrographs comparing non-refined and refined representations of breach geometry.

The discharge of the reference hydrograph from the simulation with the 5m mesh had a peak flow rate of 4,259 m^3/s as shown in Table 1. Chauhan et al. (2008) described the same dam breach scenario and gave estimates of the peak flow rate in two ways. The first estimate of 6,908.8 m^3/s was obtained by using parameters given by Froehlich (1995a) and modeling the peak discharge with a program called DAMBRK. The second estimate used parameters given by Froehlich (1995b) to directly estimate the peak discharge of 3,637 m^3/s. Chauhan explains that using the parameters from Froehlich (1995a) for input into DAMBRK causes overestimates of the peak discharge. Thus, the discharge predicted by Froehlich (1995b) serves as a low estimate while the discharge predicted by DAMBRK serves as a high estimate for the expected results from the present model. Macchione (2008a, 2008b) also gives estimates for peak discharge parameters which result in a value of 7,737 m^3/s for the present case. Though the peak discharge estimated from the various sources varies considerably, the value predicted by the current model seems to agree with the general trend.

Table 1. Peak discharge flow rates calculated using CCHE2D-FLOOD results, DAMBRK results, and equations of Froehlich and Macchione.

Model	Peak discharge (m^3/s)
CCHE2D-FLOOD	4,258.7
DAMBRK	6,908.8
Froehlich	3,626.6
Macchione	7,737.0

The simulations were run one at a time on a desktop computer with a dual-core AMD microprocessor running at 3.01GHz to clock their CPU time. For the different cases, the CPU times to run 3,000 simulation seconds are given in Table 2. The coarser meshes had fewer total cells and calculated the fastest while the finer meshes took longer to calculate, as expected. It is remarkable to consider that the 20 m mesh with refinement, for instance, yielded a discharge hydrograph comparable to the 5 m unrefined mesh while requiring only 1.2% of the CPU time to compute with the current model. Obviously not all scenarios will demonstrate such drastic gains in computational efficiency as the simple cases shown here. In general, however, the larger the domain, the more efficiency there is to gain from using local mesh refinement as opposed to refining the mesh globally; that is, if the only goal is to capture the breach progression and resulting hydrograph accurately.

Table 2. Computation time for simulations with each mesh size with and without local refinement.

	CPU time (s)	
Coarsest mesh	With refinement	No refinement
40m	16.7	6.5
20m	73.6	67.3
10m	532.5	561.7
5m	N/A	6167.0

5 DISCUSSION AND CONCLUSIONS

A local mesh refinement technique utilizing quadtree data structures was implemented in an existing two-dimensional dam break flood model. This local refinement was used to more accurately represent the progression of dam breach geometry while allowing the mesh to remain coarse elsewhere in the domain. This resulted in reasonable agreement with simulations where the entire mesh was refined, but required only a fraction of the computational time. The peak discharge value through a given breach profile compared well with simulations carried out with the numerical model DAMBRK along with values predicted by Froehlich and Macchione.

However, the breach discharge hydrograph is not the only important factor to consider when choosing which size mesh to use for a simulation. Often there will be a certain size mesh below which the data does not exist.

The method presented could be quite beneficial in situations where many simulations are to be run to determine other simulation parameters, such as the approximate length of time required for the flood wave to reach a certain location or for determining an appropriate upstream reservoir elevation. The modeler is freed from the burden of setting up additional meshes since the quadtree local mesh refinement procedure is automatic.

It should also be noted that the application of mesh refinement is not a straightforward process, as quadtree can create some ambiguities. The previously-mentioned case of determining a parent cell's flow values when some child cells are dry is one such ambiguity. Another ambiguity stems from the condition that if a cell has an adjacent cell that is simultaneously dry and has a higher bottom elevation than the current cell's water surface, the velocity in that is set to zero direction in the current cell. However, if a cell is adjacent to a cell of higher refinement, and one of the adjacent refined cells fulfills the condition while the other does not, the situation becomes ambiguous. The current model does not set the velocity to zero in this case. However, these ambiguities did not have any measurable influence on the numerical results, at least for the test cases simulated so far.

It may be possible to better represent the breach progression at all levels of refinement by averaging the elevation of the breach geometry across the length of the each cell representing the dam as opposed to using only the single elevation value at the cell center. This is being implemented in a future version of the model.

REFERENCES

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.

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

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

Chauhan, S., Bowles, D., Anderson, L. 2008. Do current breach parameter estimation techniques provide reasonable estimates for use in breach modeling?. Dam Safety 2004, Proc., ASDSO 2004 Annual Conference, Sept. 26–30, 2004. Phoenix (CD-Rom). Association of State Dam Safety Officials, Lexington, Ky.

Froehlich, D.C. 1995a. Embankment dam breach parameters revisited. Proceedings of the 1995 ASCE Conference on Water Resrouces Engineering, San Antonio, Texas. August. pp. 887-891.

Froehlich, D.C. 1995b. Peak outflow from breached embankment dam. J. Water Resour. Plan. Manage. Div., Am. Soc. Civ. Eng., Volume 121, Issue 1, pp. 90–97.

Macchione, F. 2008a. Model for predicting floods due to earthen dam breaching. I: Formulation and evaluation. ASCE, J. Hydraul. Eng., Volume 134, Issue 12, pp. 1688–1696.

Macchione, F. and Rino, A. 2008b. Model for Predicting Floods due to Earthen Dam Breaching. II: Comparison with Other Methods and Predictive Use. ASCE, J. Hydr. Engrg. Volume 134, Issue 12, pp. 1697-1707

Miglio, E. and Altinakar, M.S. 2008. Representation of Linear Terrain Features in 2D Free Surface Models using Cut-Cell Boundary Method. Proceedings of River Flow 2008, International Conference on Fluvial Hydraulics, Cesme, Izmir, Turkey, September 3-5, 2008.

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.

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.