XVIII International Conference on Water Resources
CMWR 2010
J. Carrera (Ed)
©CIMNE, Barcelona, 2010

REPRESENTATION OF DAM-BREACH GEOMETRY ON A REGULAR 2-D MESH USING QUADTREE LOCAL MESH REFINEMENT

M.S. Altinakar*, **M.Z. McGrath*** **and E. Miglio**◇

*National Center for Computational Hydroscience and Engineering,University of Mississippi, Oxford, MS, USA
e-mail: {altinakar,mzmcgrat}@ncche.olemiss.edu, web page: http://www.ncche.olemiss.edu/

◇MOX, Dept. of Mathematics, Politecnico di Milano
Milano, Italy
e-mail: edie.miglio@polimi.it, web page: http://mox.polimi.it

Key words: Dam-break; Flood; Quadtree; Mesh Refinement; Breach hydrograph.

Summary. 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 changing breach geometry is crucial to the accuracy of the resulting flood wave, yet the method of representing the breach 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.

1 INTRODUCTION

Two-dimensional numerical models are very useful 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. Using a mesh of this type 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 water will flow. It is possible to separately compute the flow of water from the reservoir through the dam breach and apply the resulting hydrograph as a boundary condition to the 2D model. Another possibility, and the one used in the current model, is to model the reservoir, dam breach progression, and the downstream inundation directly at the same time. In this case the elevation of terrain is given for each cell in the domain and some value of water depth is assigned to the cells occupying the area of the reservoir. The cells occupying the area of the dam itself are given a bottom elevation equal to the crest elevation of the dam. This approach results in some problems because each cell can have only a single value for each variable, including bottom elevation. Typical cell sizes for calculation range from about 10m to 100m square, which means that features smaller than this size may not be represented accurately. Even if sufficient data is available to refine the entire mesh, halving the cell size doubles the number of cells in both x- and y- directions and halves the time step in an explicit scheme. This yields a four-fold increase in the total number of cells and doubles the number of time steps to calculate, resulting in an eight-fold increase in computational burden. The flow of water through the breach of a dam is of paramount importance to the resulting downstream flood. However it can also be difficult to model with a mesh that is coarser than the geometry of the breach itself, especially when the dam is not aligned with the axes of the mesh. Ideally, accurate discharge through the breach in a dam should be obtained despite the ability of the mesh to resolve small breach geometry features. 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 ability to locally refine the mesh in the vicinity of the dam is a worthwhile endeavor. This was accomplished using a version of quadtree mesh refinement.

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.

The quadtree structure has been implemented in a pre-existing verified and validated two-dimensional numerical code called CCHE2D-FLOOD ([1]-[2]). 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 THE EXISTING 2D NUMERICAL CODE

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 $\mathbf{U} = [u,\, hu,\, hv]^T$, $\mathbf{F}(\mathbf{U}) = [hu,\, huu,\, huv]^T$, $\mathbf{G}(\mathbf{U}) = [hv,\, huv,\, hvv]^T$ and $\mathbf{S} = [0,\, -gh\frac{\partial Z}{\partial x} - g\frac{u\sqrt{u^2+v^2}}{C^2},\, -gh\frac{\partial Z}{\partial y} - g\frac{v\sqrt{u^2+v^2}}{C^2}]^T$ 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; moreover h is the water depth, u and v are the velocities, g is the gravity acceleration, Z is the water surface elevation and C is the Chezy coefficient.

The discretization reads as follows

$$\mathbf{U}_{ij}^{n+1} = \mathbf{U}_{ij}^{n} - \frac{\Delta t}{\Delta x_i}(\mathbf{F}_{i+1/2,j} - \mathbf{F}_{i-1/2,j}) - \frac{\Delta t}{\Delta y_i}(\mathbf{G}_{i,j+1/2} - \mathbf{G}_{i,j-1/2}) + \Delta t \mathbf{S}_{ij} \tag{2}$$

where $\mathbf{F}_{i+1/2,j}$, $\mathbf{F}_{i-1/2,j}$, $\mathbf{G}_{i,j+1/2}$ and $\mathbf{G}_{i,j-1/2}$ are the fluxes on the cell faces; a simple upwind method is adopted to compute the intercell fluxes. A more detailed description of the algorithm can be found in [3].

2.1 Procedure

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.

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 THE QUADTREE METHOD

The data structures for the quadtree mesh were implemented similarly to the method proposed by [4]. 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 in the previous level. 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. A cell with no child cells is called a leaf cell, while a cell with child cells is called a stem cell.

3.1 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 refinement, and then progressively toward the edges and cells with lower levels of refinement. For leaf edges, the fluxes are calculated as usual; 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 equation (2). 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.

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 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, 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.

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 1 shows the result of the refinement process for a typical dam.

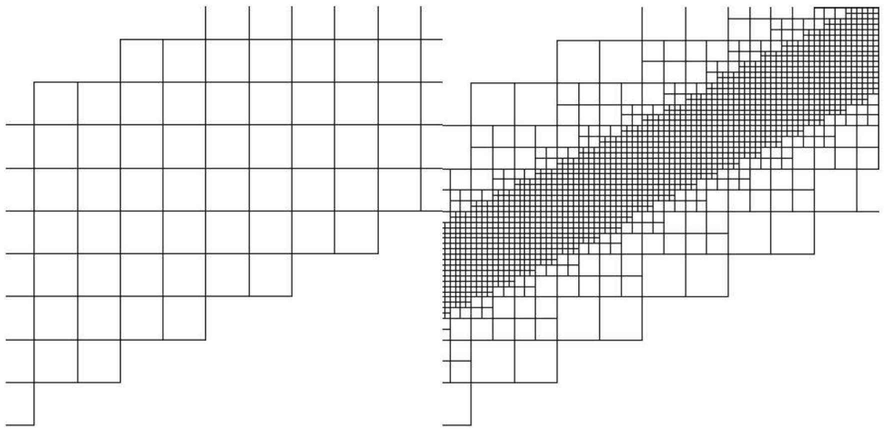

Figure 1: Cells underneath a dam identified (left) and refined to level 4 (right).

4 TEST CASE

The model with quadtree refinement was validated against the existing model using a test case from [5]. The computational domain was a $1000m$ wide and $1680m$ long area with a flat bottom. A $1000m$ wide and $1280m$ long portion of the computational domain was used to model a $19.35m$ deep reservoir having a storage volume of $24768000m^3$. The remaining area downstream of this dam was initially dry. The dam had an 80-meter wide section in the middle that was modeled with a linearly progressing breaching sequence. The breaching started right at the beginning of each simulation. It took 2520 seconds (42 min) to attain the final trapezoidal breach geometry which had a bottom width of $46m$ and a top width of $80m$ (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 scenario was simulated using regular mesh sizes of $5m$, $10m$, $20m$, and $40m$ per cell without refinement and again with the same regular mesh sizes in addition to local refinement near the dam. In each case the simulation lasted 3000 seconds, which was enough to fully model the breach widening and begin to capture the falling end of the hydrograph.

The simulations without mesh refinement yielded the discharge hydrographs shown on the left of Figure 2. 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 $5m$ mesh was considered to be the most accurate, and other simulations were compared to it. Figure 2 clearly shows that the breach discharge hydrograph depends on the cell size used to represent the breach. The $80m$-breach is represented with just two adjacent cells in the $40m$ mesh, and this causes the peak discharge to be overestimated by more than 25% compared to the $5m$ mesh hydrograph. It is interesting to note that the differences between the results of the $10m$ mesh and the $5m$ mesh are small, which

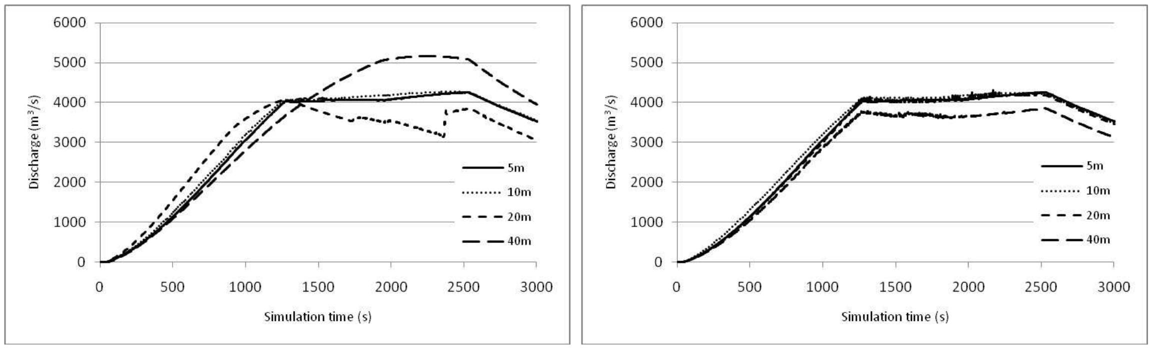

Figure 2: Dam breach discharge hydrographs for various mesh resolutions with no refinement (left) and with refinement near the dam (right).

gives the notion that further refinement would yield little increase in accuracy.

A new series of simulations were 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 $40m$ cells were refined three levels, $20m$ cells were refined two levels, and $10m$ cells were refined one level. In each case the smallest cells in the quadtree mesh were $5m$ square. The discharge hydrographs for these three simulations with quadtree mesh refinement are plotted on the right part of Figure 2 together with the hydrograph simulated with the $5m$ regular mesh over the entire computational domain.

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 $5m$ unrefined mesh, which was assumed to be best available estimation of the breach hydrograph.

5 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 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

[1] M.S. Altinakar, M.Z. McGrath, Y. Ozeren and E. Miglio. 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.

[2] M.S. Altinakar, M.Z. McGrath, Y. Ozeren and E. Miglio. 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.

[3] X. Ying and S. Wang and A. Khan. Numerical Simulation of Flood Inundation due to Dam and Leave Breach. *Proceeding of ASCE World Water & Environmental Resources Congress 2003*, (2003).

[4] H. Zeng and C. Shu and Y. Chew. An object-oriented and quadrilateral-mesh based solution adaptive algorithm for compressible multi-fluid flows. *J. Comput. Physics*, **227**, 6895–6921, (2008).

[5] S. Chayhan and D. Bowles and L. Anderson. Do current breach parameter estimation techniques provide reasonable estimates for use in breach modeling? *Dam Safety 2004, Proc., ASDSO 2004 Annual Conference*, (2004).