A Two Numerical Modeling for Urban Flood with Adaptive Mesh Refinement Method

AMR モデルによる洪水氾濫の数値解析

By

Wentao Gong

A Dissertation

Submitted to the Division of Field Environment Engineering

Hokkaido University

In Partial Fulfilment of the Requirements

for the Master Degree

February 2021

Supervised by Prof. Toshiki Iwasaki

Content

Abstraction

A Two-dimensional Numerical Modeling for Urban flood with Adaptive mesh refinement

AMR モデルによる洪水氾濫の数値解析

Division of Field Environment Engineering　Water Disaster and Environmental Research Laboratory　Wentao Gong
Keywords：adaptive mesh refinement, urban flood, computational time, CIP

1. Background

O'Donnell et al.,(2020) reported that local and national flood risks have increased since 2008 and are expected to increase further in future owing to climate change, urbanization, reductions in urban green spaces and deteriorating urban water management infrastructure. DESA,U et al.,(2018) reports that, by 2050, 68% of the world's population is expected to reside in cities. So managing urban flood risk is one of the key global challenges of the twenty-first century with future flood risk.

However, urban flood modeling is prohibitive expensive in computational memory and running time by using the conventional way .

Adaptive mesh refinement method Proposed by Berger and Oliger (1984) can refine and coarse the grid to use high resolution where it is needed or interested, which in return can save the computational cost in some way.

2. Previous Research

MacNiece et al.(2000,2004) presented a toolkit called PARAMESH. It is a set of Fortran 90 routines designed to enable Adaptive Mesh Refinement(AMR) for a parallel distributed memory machine. The user can develop his own-solver based on the PARAMESH with AMR and parallel computing.

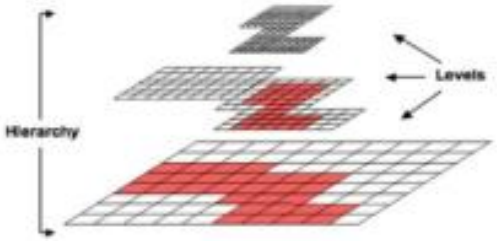


Figure 1. Block-structured AMR grid containing a hierarchy of nested refinement levels. (from MacNeice et al.2000)

Hu et al., (2018) developed a 2D control-volume and finite element flood model using adaptive unstructured mesh technology.

3. Objective

My objective is to develop a two-dimensional numerical modeling for urban flood with AMR and Finite difference method, and evaluate its accuracy and efficiency compared with the fine grid method.

4. Target City and Elevation Model

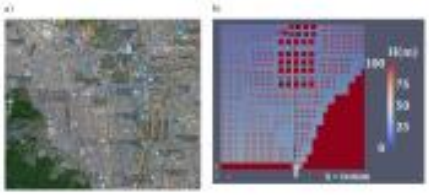


Figure 2. a) The center of Sapporo city, Japan. b) elevation data of the objective area (Morikawa (2019)).

The proposed flood model with AMR is applied to a case of flooding in Sapporo, Japan. Fig.1a shows that the objective area is highly urbanized by a lot of buildings. The elevation's resolution is 10m in each direction. In this elevation data, the different spatial density of building is modeled based on the aerial photograph (i.e., Fig.1a). When refinement or de-refinement is taken place in the computation, this elevation data is interpolated into every node of the computational grid by weighted averaging as,

$$\eta = \frac{\Sigma z / r}{\Sigma 1 / r} \tag{1}$$

5. Computational Model

5.1 Governing Equations

In the Cartesian coordinate system, Non-conservative shallow water equations are given as follows:

Continuity equation:

$$\frac{\partial h}{\partial t} + \frac{\partial hu}{\partial x} + \frac{\partial hv}{\partial y} = 0 \tag{2}$$

Momentum equation in x direction for non-viscous fluid:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -g\frac{\partial h}{\partial x} - g\frac{\partial \eta}{\partial x} - \frac{\tau_x}{\rho h} \tag{3}$$

Momentum equation in y direction for non-viscous fluid:

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -g\frac{\partial h}{\partial y} - g\frac{\partial \eta}{\partial y} - \frac{\tau_y}{\rho h} \tag{4}$$

Where x and y are the spatial coordinate components in the Cartesian coordinate system; t is time; u and v are the depth averaged velocity components in x and y directions, respectively; h is water depth; η is the elevation; g is the gravitational acceleration. τ_x and τ_y can be obtained by the Manning formula and bed shear stress.

$$\tau_x = \tau_b \frac{u}{V} = \frac{\rho g n^2 u \sqrt{u^2+v^2}}{h^{\frac{1}{3}}} \quad (5)$$

$$\tau_y = \tau_b \frac{u}{V} = \frac{\rho g n^2 v \sqrt{u^2+v^2}}{h^{\frac{1}{3}}} \quad (6)$$

Where, τ_b is the bed shear stress; i is the gradient of water channel,slope; V is velocity in flow direction $\sqrt{(u^2+v^2)}$; n is the Manning's roughness coefficient.

5.2 Numerical Schemes

To solve the Non-conservative N-S equations, the separation technique is adopted, which split the momentum equations into Non-advection term and advection term. For discretization, the staggered grid is used, which can guarantee the compatibility of the non-conserve form.

For the non-advection term, the implicit method is used to guarantee the divergence free of velocity for incompressible flow and the mass conservation due to the runoff error and residual error.

For the advection term, The constrained interpolation profile is adopted, which can prevent the numerical diffusion and predict the velocity in grid cell well.

5.3 Boundary Condition

For the solid wall, the no-slip boundary condition was set.

For inflow boundary, we give a constant slope $i = 0.001$ to calculate the velocity of inflow.

5.4 Time iteration

For each direction, we can calculate the Courant Number as below

$$C = \frac{u\Delta t}{\Delta x} \le 1 \quad (7)$$

The Courant Number is set 0.2 in this study. At every time step, we could find the maxim velocity in x, and y direction. Then we could obtain the smallest dt

$$dt_x = \frac{C\Delta x}{u_{max}}, dt_y = \frac{C\Delta y}{v_{max}} \quad (8)$$

$$dt = \min(dt_x, dt_y) \quad (9)$$

This time step is to used everywhere, regardless of local refinement level.

6. Application

6.1Calculation conditions

Table 1. The calculation condition of the computational model.

Computational domain		2800 m x 2800 m
Manning's roughness coefs.		0.03
Water discharge		771 m^3/s
AMR	**minimum refinement level (level 1)**	280 m x 280 m
	Maximum refinement level (level 7)	4.4 m x 4.4 m
Fine grid model		4.4 m x 4.4 m
Course grid model		17.5m x 17.5m
Calculation end time		2000 seconds

6.2 Results

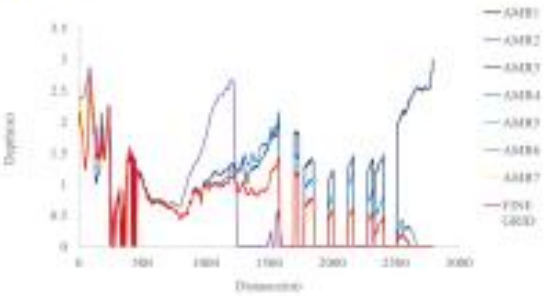


Fig 3. The water depths along the black line (showed in Fig. 1b) at 2000 seconds on various refinement criteria and the fine grid case.

Among these test cases, the fine grid cost 10h45minitues time, ARM1,5,6,7 have the same results with fine grid method on water depth, and cost 3h37min, 3h25min, 3h36min, 3h37min respectively. Among them, the AMR5 is the best.

7. Conclusions

The results showed that AMR can save 47%-68.2% computational time without significant loss of computational accuracy on the uniform discharge case.Among these cases, the best refinement criteria for urban flood is water surface slope.

References

(1) O'Donnell, E. C., & Thorne, C. R. (2020). Drivers of future urban flood risk. Philosophical Transactions of the Royal Society A, 378(2168), 20190216.

(2) DESA, U. (2018). World urbanization prospects: the 2018 revision, key facts. New York: NY. Available online at: https://population. un. org/wup/Publications/(Accessed December 20, 2018).

(3) MacNeice, P., Olson, K.M., Mobarry, C., Fainchtein, R. and Packer, C. (2000). PARAMESH: A parallel Adaptive Mesh Refinement Community Toolkit. Computer Physics Communications, 126(3), 330-354.

(4) Morikawa, G.(2019). Numerical analysis of flood with hyper grid model, dissertation for Master degree of Hokkaido University, Sapporo, Japan.

Chapter 1- Introduction

1.1Background

1.1.1 Climate change and urban flood risks

It may be a general understanding that climate change has caused a lot of natural disasters such as heavy rainfall, flood, and debris flow in the last few years, resulting in a great of loss of people's lives, property and critical infrastructure systems.

O'Donnell et al.,(2020) reported that local and national flood risks have increased since 2008 and are expected to increase further in future owing to climate change , urbanization, reductions in urban green spaces and deteriorating urban water management infrastructure (Lowe,J.A et al.,2018; Speight VL et al.,2015). With the development of urbanization, urban flood risk has become more and more concerned. DESA,U et al.,(2018) reports that, by 2050, 68% of the world's population is expected to reside in cities. So managing urban flood risk is one of the key global challenges of the twenty-first century with future flood risk.

1.1.2 Computational technologies

Several computational technologies have been nowadays an important tool for mitigating the damage due to the natural disasters by predicting risk and hazard in advance. Specifically, the numerical model simulating the physical processes of floods based on detailed hydrological and hydraulic data have been widely used in urban flood process simulations and mechanism analysis (e.g., Caian et al., 2021). With the development and maturity of big data technology,extensive advanced deep learning technologies such as artificial intelligence have been successfully applied for the prediction of hazards and risk in environmental sciences (e.g., Sajedi-Hosseini et al., 2018; Choubin et al., 2019). In our study, we focused on proposing a framework of flood modeling with adaptive mesh refinement method, which can provide the detailed information in terms of flood characteristics, contributing decision making of some countermeasure for flooding and evacuation plan for residential people.

However, a flood modeling in a large scale domain with complicated bed geometry and infrastructure arrangement will be a difficult task due to the expensive computation cost.

1.1.3 Geometries data accuracy

In the past, it is not possible to obtain the highly accurate observation results due to the underdeveloped observation technology. However, in the past of years, accurate environmental data has become easy to obtain due to the rapid measuring instruments. For example, with regard to

topography data, it is accessible to use the radar surveying by aircraft and survey using a 3D scanner. To provide reliable flood modeling predictions in urban areas, high-resolution simulation is essential in order to resolve the complex urban topographic features, for example, buildings, streets and embankments.

1.2 Problem statement

With the technologies development, the more accurate data can be obtained. On the other hand, there is concern that the computational memory and calculation time will be prohibited expensive due to finer mesh. Computational cost has become a major problem in numerical modeling for urban flood.

1.3 Objective

To overcome this problem, many techniques has been developed in the past a decade. These approaches include: reduced-complexity models (Liu and Pender, 2010), parallelization (Hankin et al., 2008; Neal et al., 2010), unstructured mesh (Wang et al., 2010), adaptive grid-based methods (Wang and Liang, 2011), grid coarsening (Yu and Lane, 2006a), and hyper grid method (Morikawa and Kimura, 2018). Among these methods, the grid coarsening and hyper grid method are straightforward to reduce the computation time. However, the low resolution leads to the loss of information and less accuracy of modeling results. In this study, my objective is to propose a two-dimensional numerical modeling for urban flood with adaptive mesh refinement method, and evaluate its accuracy and efficiency compared with the fine grid method.

1.4 Static mesh approach VS AMR

1.4.1 Static grid approach

High resolution require for handling difficult regions(discontinuities, steep gradients, shocks, etc), leading to prohibited expensive computational cost.

1.4.2 Adaptive mesh refinement

1) Start with a coarse grid
2) Identify regions that need finer grid
3) Superimpose finer sub-grids only on those regions
4) Increased computational time savings over a static mesh approach
5) Increased storage savings over a static mesh approach

1.5 Previous research

1.5.1 Multi-grid methods

Multi-grid methods (Brandt, 1977; Briggs, 1987; Briggs et al., 2000) are a class of simulation techniques which illustrate important principles related to structured adaptive mesh refinement. These methods solve complex, sophisticated simulations by rapidly reducing the error inherent in many single grid iterative methods. Multi-grid methods employ a hierarchy of grids of varying resolution, each grid covering the entire computational domain (Figure 1.3.1). The underlying premise is that although iteration on a fine mesh quickly eliminates the high frequency components of error, low frequency components take much longer, resulting in an inefficient or inaccurate solution. However, by iterating on meshes of various scales, the smoother error components can be reduced quickly as well.
The multi-grid with adaptive strategies has much in common with traditional multi-grid methods, which are a special case of certain classes of structured ARM strategy.

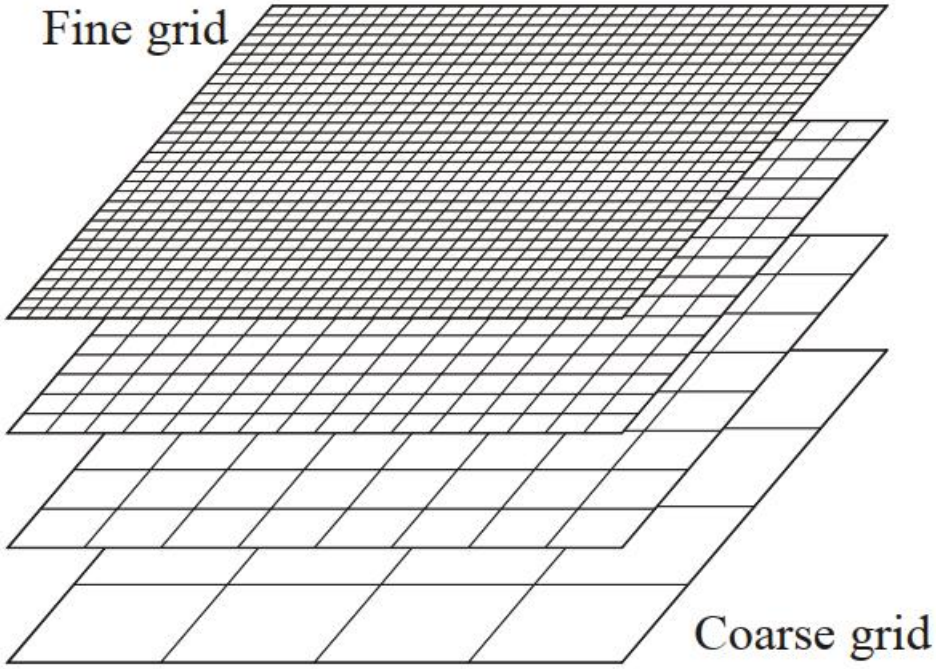


Figure 1.1 Multi-grid hierarchy (Hoang, 2005)

1.5.2 Moving grid method

Petzold et al., (1987) reported a scheme for choosing a moving mesh based on minimizing the time rate of change of the solution in the moving coordinates for one-dimensional systems of PDEs. They showed how to apply this idea to systems where the time derivatives cannot be solved for explicitly, writing the moving mesh equations in an implicit form.

Semper et al., (1995) Presented a new front tracking method based on grid deformation, which was combined with a streamline upwind Petrov-Galerkin finite element method to produce an adaptive method for application to convection dominated transport equations. The practicality of moving grid method is examined and the method on one-dimensional example was also demonstrated.
Koltakov et al., (2012) developed a a second-order accurate Navier – Stokes solver based on r-adaptivity of the underlying numerical discretization. The motion of the mesh is based on the fluid velocity field; however, certain adjustments to the Lagrangian velocities are introduced to maintain quality of the mesh. The adjustments are based on the variational approach of energy minimization to redistribute grid points closer to the areas of rapid solution variation. To quantify the numerical diffusion inherent to each method, the changes in the background potential energy, computation of which is based on the density field.were monitored. They demonstrated on a standing interfacial gravity wave simulation how using our method of grid evolution decreases the rate of increase of the background potential energy compared with using the same advection scheme on the stationary grid. To further highlight the benefit of the proposed moving grid method, they apply it to the nonhydrostatic lock-exchange flow where the evolution of the interface is more complex than in the standing wave test case. Naive grid evolution based on the fluid velocities in the lock-exchange flow leads to grid tangling as Kelvin−Helmholtz billows develop at the interface. This is remedied by grid refinement using the variational approach.

1.5.3 PARAMESH

Since the early 1980's, Berger and Oliger (1984) have been developing an adaptive mesh refinement method for structured meshes. MacNiece et al.(2000,2004) presented a toolkit called PARAMESH.
It is a package of Fortran 90 subroutines is designed to provide an application developer with an easy route to extend an existing serial code which uses a logically cartesian structured mesh into a parallel code with adaptive mesh refinement(AMR). Alternatively, in its simplest use, and with minimal effort, it can operate as a domain decomposition tool for users who want to parallelize their serial codes, but who do not wish to use adaptivity.

The package builds a hierarchy of sub-grids to cover the computational domain, with spatial resolution varying to satisfy the demands of the application. These sub-grid blocks form the nodes of a tree data-structure (quad-tree in 2D or oct-tree in 3D). Each grid block has a logically cartesian mesh, and the index ranges are the same for every block. Thus, in 2D, if we begin with a 10x20 grid on one block covering the entire domain, the first refinement step would produce 4 child blocks, each with its own 10x20 mesh, but now with mesh spacing one-half that of its parent. Any or all of these children can themselves be refined, in the same manner. This process continues, until the domain is covered with a quilt-like pattern of blocks with the desired spatial resolution everywhere. During the refinement process, the refinement level is not allowed to jump by more than one level at any location in the spatial domain.

Figure 1.1 shows a simple example of a 2D mesh generated by PARAMESH and the corresponding quadtree. Noted that, in this example, each block contains exactly 6x4 cell centered data points and the heavy lines in the grid diagram indicates the boundaries of sub-grid blocks, and the lighter lines

indicate individual grid cells.

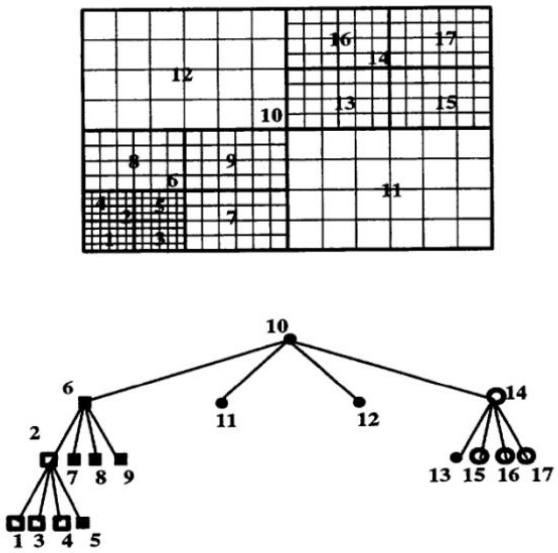


Figure 1.1 A simple 2D example of a grid block tree governing a rectangular domain. (from MacNeice et al.2000)

1.5.4 Shallow Water Equations On Adaptive Mesh Refinement

Wang et al., (2011) tested the a robust numerical tool for modeling different types of floods, e.g. rapid-varying dam breaks or slowing-evolving inundations based on a novel adaptive grid system, the model solves the 2D shallow water equations using a first-order accurate Godunov-type scheme to approximate the flood hydrodynamics.

Pons et al., (2017) used finite volume method with the AMR method on unstructured mesh to solve the multi-dimensional Saint-Venant system, which allows quick meshing and easy parallelization.

Huang et al., (2015) developed a coupled hydrodynamic and non-capacity sediment transport model on adaptive nonuniform rectangular mesh.

Hu et al., (2018) developed a 2D control-volume and finite element flood model using adaptive unstructured mesh technology.

Hu et al., (2019) have first introduced an adaptive isotropic unstructured mesh technique to urban flooding simulations and apply it to a simple flooding event observed as a result of flow exceeding the capacity of the culvert during the period of prolonged or heavy rainfall.In his work, the adaptive mesh flooding model based on 2D shallow water equations (named as Floodity) has been further developed by introducing (1) an anisotropic dynamic mesh optimization technique (anisotropic-DMO); (2) multiple flooding sources (extreme rainfall and sea-level events); and (3) a unique combination of anisotropic-DMO and high-resolution Digital Terrain Model (DTM) data. It has been applied to a densely urbanized area within Greve, Denmark.

1.6 Thesis Overview

This thesis is divided into give chapters.

Chapter 2 provides a brief overview of adaptive mesh refinement method.

Chapter 3 shows the shallow water equations and advection-diffusion equations, finite difference discredization and test cases with AMR

Chapter 4 represents the application of urban flood numerical model with AMR and discusses the results

Chapter 5expresses some conclusions of this study and suggestions for future works.

Chapter 2-Overview of Adaptive Mesh Refinement

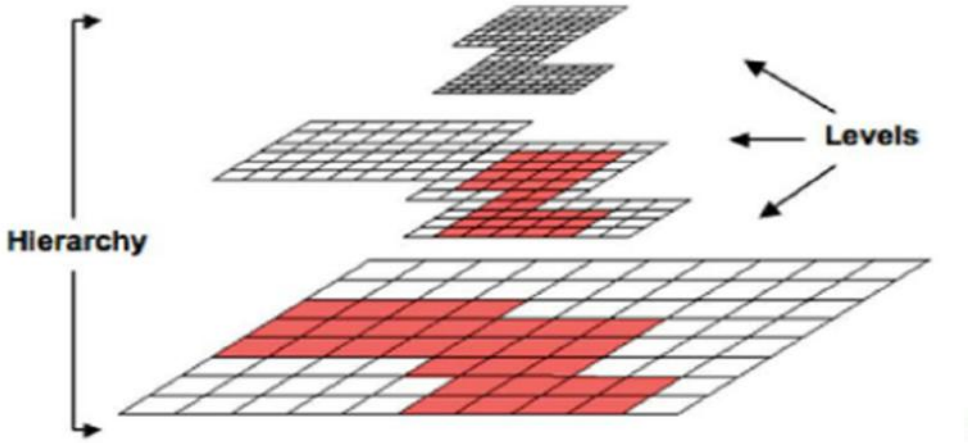


Figure 2.1. Block-structured AMR grid containing a hierarchy of nested refinement levels. (from MacNeice et al.2000)

In many problems, the sufficient resolution to adequately represent the smallest scale features of this solution must be provided. As computational work in these schemes are directly proportional to the number of grid points. The use of a uniform grid with the finest desired resolution is highly inefficient and can lead to a prohibitive discretization both in computation and storage requirement.

Adaptive mesh refinement provides the means for maintaining the computational tactability without sacrificing accuracy. This idea is to concentrate additional computational effort to the areas of interests in the solution. This method starts with a coarse grid that cover the entire computational domain, As the solution advances, regions in the domain requiring additional resolution are identified and finer grids are overlayed on the tagged regions of the coarse grid. Illustration of the block-structured AMR grid is shown in Figure 2.1.

2.1 Adaptive Grid Hierarchy

The Berger-Oliger(1994) presented the grid of component grids at any level l of the grid hierarchy to be an integral multiple of the grid spacing of the component girds at the next level $l+1$, i.e $h_l = kh_{l+1}$, where k is integer. Further, component grids at any level l of the gird hierarchy must be locally uniform with space and time resolution, h_l and t_l , such that $h_l = \lambda t_l$; typically λ >1. All the grid points of any component grid at level $l+1$ must lie in the convex hull of a component grid at level l .The two views of the Berger-Oliger adaptive grid hierarchy are illustrated in Figure 2.2.

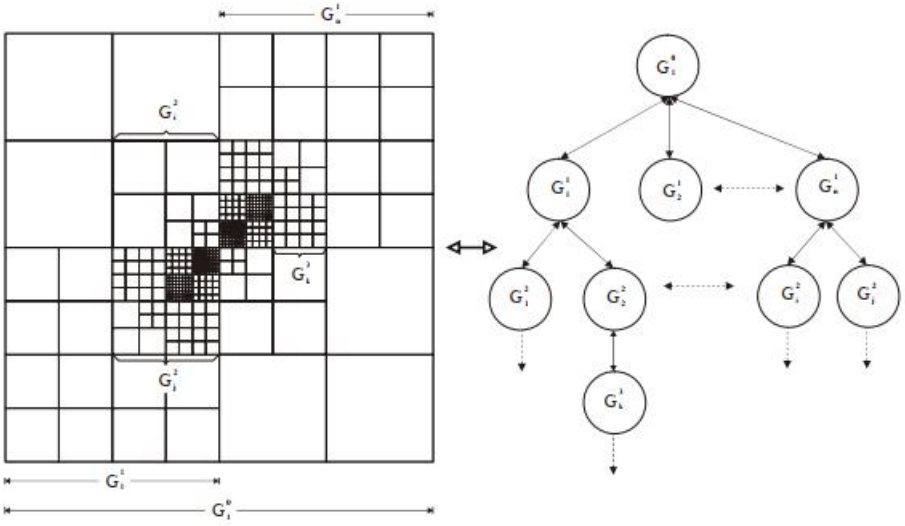


Figure 2.2: Adaptive Grid Hierarchy-2D (from Hoang, H. M, 2005)

2.2 Time Integration

In a parallel AMR code, computing an integration timestep which do not exist in a serial non AMR code, would cause a number of extra complications. The processors must compute local timesteps for each of leaf blocks, then share the information about the shortest timestep required at any point in the mesh.

If the same time step is to used everywhere, regardless of local refinement level, then the time step must be this global minimum time step. If the time step is adopted to vary depending on the local refinement level, it must be a monotonic function of refinement level. The shortest time step must be used on the finest leaf blocks, and the longest on the coarsest leaf blocks, and all timesteps must be integer multiples of the finest time step.

2.3 Flux conservation (from MacNeice et al.2000)

The fluxes entering or leaving a grid cell through a common cell face shared with 4 cells of a more refined neighbor, equal the sum of the fluxes across the appropriate faces of the 4 smaller cells.
This is illustrated in the next diagram, for the case in which the fluxes update cell-centered quantities. The colored faces indicate the cell face areas common to both grid blocks. The fluxes at this common boundary used to update the first interior grid cell on the refined block must be captured by the user and stored in arrays called FLUX_X on x-faces, in FLUX_Y on y-faces and FLUX_Z on z-faces. Then a routine called AMR_FLUX_CONSERVE is called. This makes copies, TFLUX_X, TFLUX_Y and TFLUX_Z, of these arrays, and then updates FLUX_X, FLUX_Y and FLUX_Z by replacing the fluxes on the shared faces of blocks adjoining a refined block, with the sum or average of the appropriate fluxes collected from the equivalent faces on the refined neighbor, as shown in the Figure 2.4.

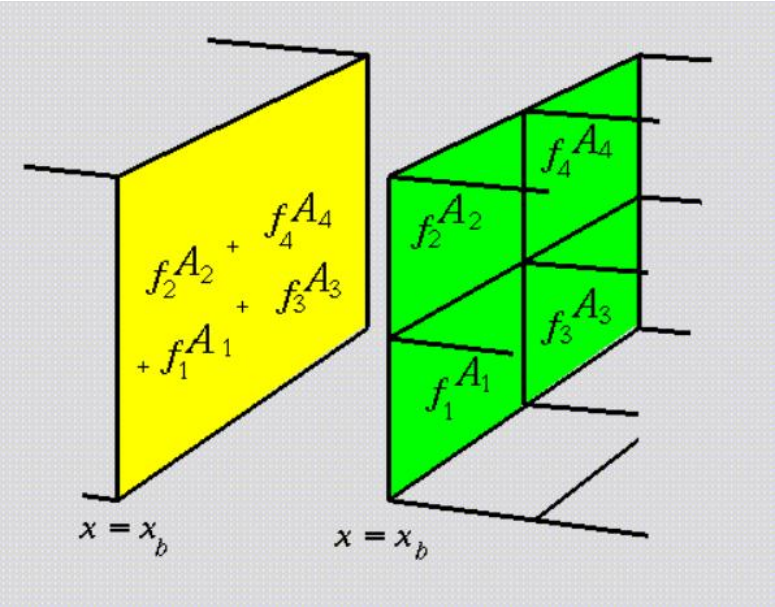


Figure 2.4 Summation of fluxes at shared cell faces on adjoining grid blocks at different refinement levels. Flux densities are denoted by f and cell face areas by A.(MacNeice et al.,2000)

2.4 Error Measure and Regridding

The error measure depends on the refinement criteria including two threshold values to decide refine or de-refine. The regions needing refinement are based on the error measure. The result may be the creation of the new level of refinement or compoent grids at existing levels, and/or the deletion of existing component grids to maintain proper nesting along the grid hierarchy.

2.5 Parameters in AMR

2.5.1 Variables of Block

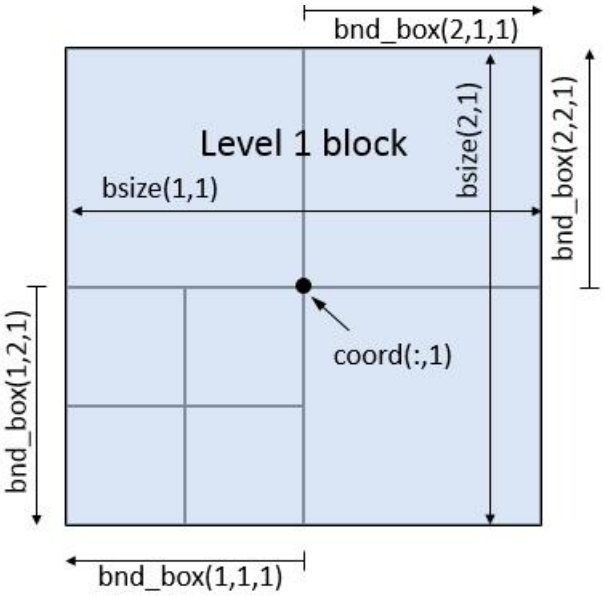


Figure 2.5 Illustration of some variables of block

In the Paramesh, some variables are given below.

Table 2.1 Some Variables of the block

Variable Name	Meaning
lrefine_min	minimum block level
lrefine_max	maximum block level
coord(:,l)	coordinate of the center of the block, l. coord(1,l) is x coord(2,l) is y.
bnd_box(:,:,l)	minimum and maximum coordinate of block, l. bnd_box(1,:,l) is minimum and bnd_box(2,:,l) is maximum
bsize(:,l)	the size of block, l, so this is given by bnd_box(2,:,l) – bnd_box(1,:,l). bsize(1,l) is the size of block, l in x direction, and bsize(2,l) is the size of block, 2 in y direction.

2.5.2 Variables of Neighbors

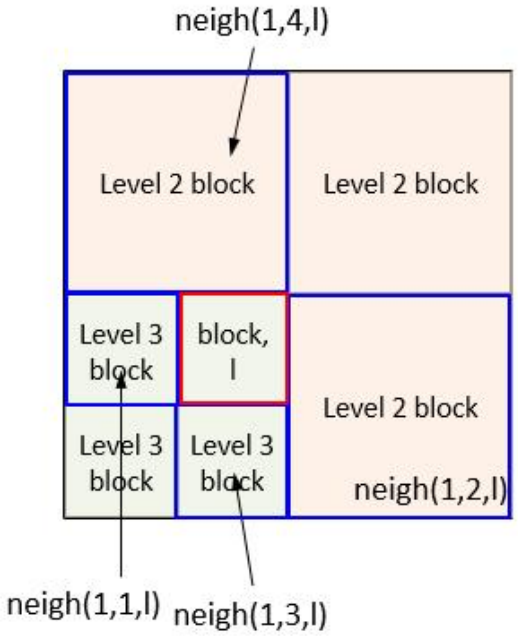


Figure 2.6 Illustration of neighbors

I suppose that the block marked surrounded by red line is the block,I, as shown in the Figure 2.6. In Paramesh, some variables of neighbors are given blew.

Table 2.2 Some Variables of the neighbor

Variable Name	Description
neigh(1,:,l)	reference number of neighbors of block, l.
neigh(2,:,l)	processor of neighbor's blocks.

If you use single computer, neigh(2,:,l) must be zero.

2.5.3 Variables of External Boundary

About the boundaries, the value of neigh(1,:,1) are set to -21 or less than -21.
The Figure 2.7 denotes the external boundary in AMR.
The array NEIGH which stores the location of neighboring grid blocks is also used to identify block faces which the user has designated as physical boundaries requiring specific boundary conditions. Normally NEIGH(2,JFACE,LB) stores the processor number on which to find the neighbor to block LB across its face JFACE, and NEIGH(1,JFACE,LB) stores the block number of that neighbor on that processor. However if these addresses are set to a value of -20 or less, this face is considered to be a physical boundary. On faces such as these the user will have to supply a routine to compute values for the guard cells using the appropriate boundary conditions. Different boundary conditions can be selected on different boundaries by using a range of values in NEIGH which are less than or equal to -20. Because values in NEIGH are inherited during the refinement process, it is enough to

identify physical boundaries when the initial grid is established.
Periodic boundary conditions can be implemented in an even simpler way. In this case when we establish the initial grid the array NEIGH must be initialized so that blocks at a periodic physical boundary store the location of the block on the opposite end of the computational domain. Then the normal calls to AMR_GUARDCELL or AMR_1BLK_GUARDCELL ensure that the guard cells at these boundaries are filled with the correct data.

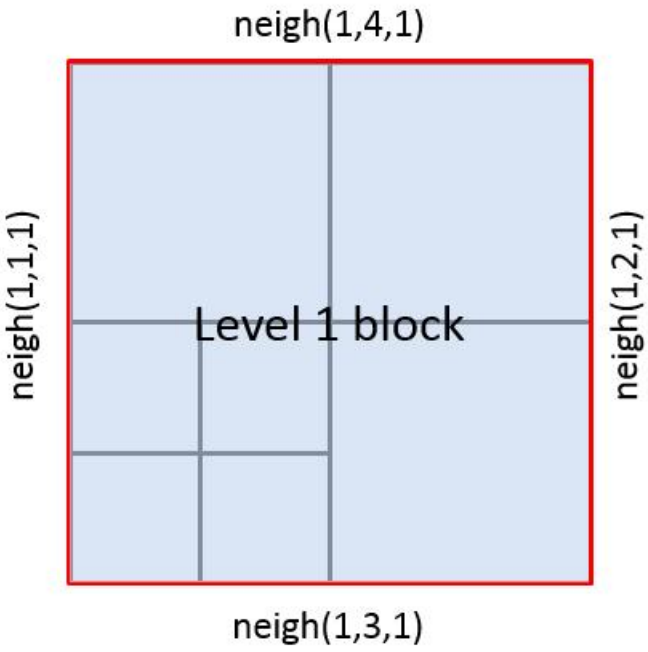


Figure 2.7 Illustration of boundaries in AMR

2.6 Grid Structure

All the grid blocks have an identical logical structure. They are assumed to be logically cartesian (or structured). By this we mean that the grid cells can be indexed as though they were cartesian. If a cell's first dimension index is i, then it lies between cells i-1 and i+1. The actual physical grid geometry can be cartesian, cylindrical, spherical, polar(in 2D), or any other metric which enables the physical grid to be mapped to a cartesian grid.
Each grid block has (NXB,NYB,NZB) interior cells, and there are NGUARD guard cells at each grid block boundary. As a result the cell centered data is indexed from 1 : NXB+2*N_GUARD in the x direction. The values of NXB,NYB,NZB and NGUARD are defined as parameters by the user, when they set values for the pre-processor variables NX_B, NY_B, NZ_B in the header file PARAMESH_PREPROCESSOR.FH. The values of NXB, NYB and NZB must be even.
The physical dimensionality of the model is set by defining the value of the pre-processor variable N_DIM in PARAMESH_PREPROCESSOR.FH. If N_DIM is 1 then a 1D model is set up and NYB and NZB are automatically set to 1. If N_DIM = 2then a 2D model is set up if L_2P5DIM = 0 , and a 2.5D model if L_2P5DIM = 1, and NZB is automatically set to 1. Finally if N_DIM = 3 a 3D model is set up.

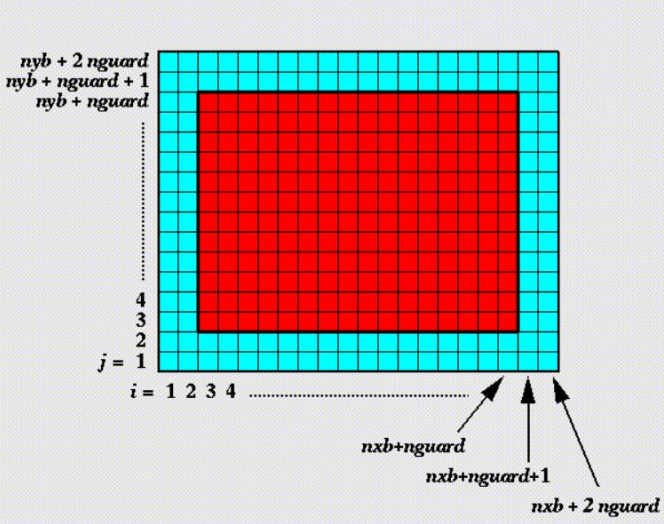


Figure 2.8 indexing scheme for a 2D grid block with 2 guard cells (nguard=2) at each block boundary. Interior cells are colored red, guard cells are colored blue.(From Olson et al., 2005)

The location and size of a grid block are specified in two different ways. The block location is stored in the array COORD. For grid block LB, say, on a particular processor, COORD(1,LB) is the location of the mid-point of the block along the first coordinate axis. Similarly COORD(2,LB) and COORD(3,LB) define the position along the other coordinate axes. A similar convention applies to the array BSIZE which stores the coordinate interval spanned by the grid block along each axis. Note that this is not always equivalent to a physical length. The second way we store this information is in the form of bounding box coordinates. The array BND_BOX stores these. The lower and upper coordinates of the block boundaries along the IC-th coordinate axis of block LB are stored as BND_BOX(1,IC,LB) and BND_BOX(2,IC,LB) respectively. The reason we choose to

store this information in two different and seemingly redundant ways is that we would like child blocks to have exact copies of those bounding box coordinates which they may share with their parent and siblings. If these quantities were computed as needed they would suffer roundoff error and slight differences between the coordinates assigned to the same physical location on different blocks. Because the appropriate data in the arrays COORD, BSIZE and BND_BOX are inherited rather than recomputed when children are created, we do not suffer these inconsistencies.

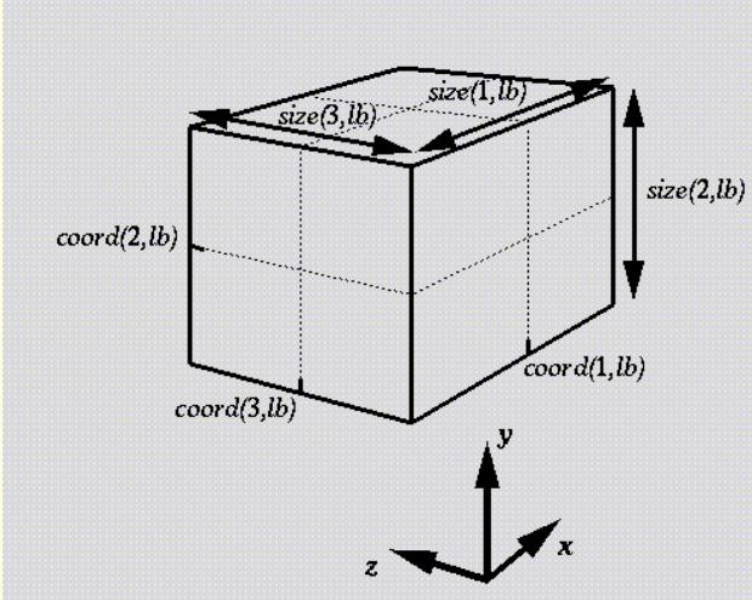


Figure 2.6. The physical relationship of the basic solution data words to their grid cell shown for grid cell indices (i, j, k) in a 3D model. (from MacNeice et al.2000)

All the grid blocks are related to one another as the nodes of a tree. When a leaf block is designated for refinement, it spawns 2 child blocks in 1D, 4 child blocks in 2D or 8 child blocks in 3D. These child blocks cover the same physical line, area or volume as their parent but with twice the spatial resolution. The J-th child of parent block LB is located at block CHILD(1,J,LB) on processor CHILD(2,J,LB). The children of a parent are numbered according to the Fortran array ordering convention, ie child 1 is at the lower x, y and z corner of the parent, child 2 at the higher x coordinate but lower y and z, child 3 at lower x, higher y and lower z, child 4 at higher x and y and lower z, and so on. The parent of block LB is located at PARENT(1:2,LB), where, as before the first word is the local block number and the second word is the processor number.

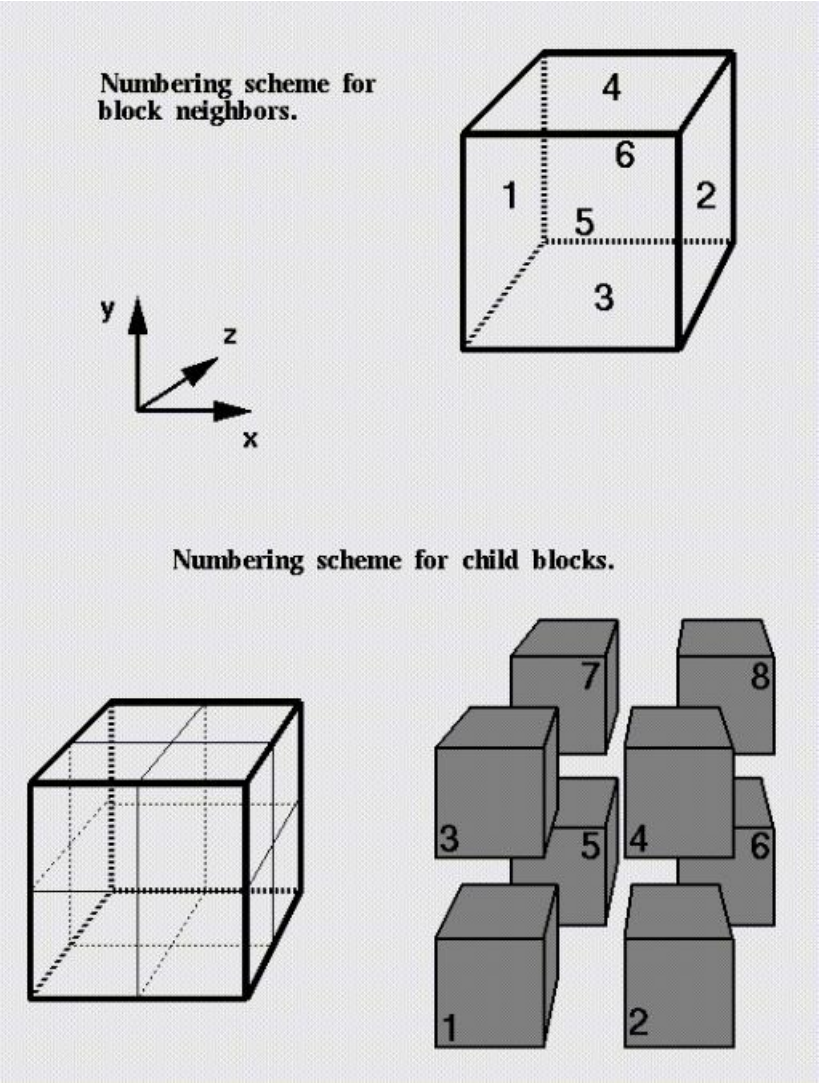


Figure 2.7 Numbering scheme for block neighbors and Number scheme for child blocks. (MacNeice et al.2000)

The locations of a a blocks neighbors are stored in the array NEIGH. The neighbor on the lower x face of block LB is at NEIGH(1:2,1,LB), the neighbor on the upper x face at NEIGH(1:2,2,LB), the lower y face at NEIGH(1:2,3,LB), the upper y face at NEIGH(1:2,4,LB), the lower z face at NEIGH(1:2,5,LB) and the upper z face at NEIGH(1:2,6,LB). If any of these values are set to -1 or lower, there is no neighbor to this block at this refinement level. However there may be a neighbor to this blocks parent. If the value is -20 or lower then this face represents an external boundary, and the user is required to apply some boundary condition on this face. The exact value below -20 can be used to distinguish between the different boundary conditions which the user may wish to implement.

The grid refinement process is constrained to guarantee that no leaf blocks ever vary by more than one refinement level from any of their neighboring leaf blocks. In this respect, a blocks neighbors also includes leaf blocks which touch it only along a single edge or at a single corner.

2.7 Data Structure

The data which constitutes the solution can include data located at

- the center point of grid cells
- the centers of the grid cell faces
- the centers of the grid cell edges
- the corners of the grid cell.

The array UNK stores the cell centered data. The face centered data is stored in FACEVARX for faces perpendicular to the x-axis, FACEVARY for faces perpendicular to the y-axis, and FACEVARZ for faces perpendicular to the z-axis. The edge centered data is stored in UNK_E_X for the edge aligned along the x-axis, UNK_E_Y for the edge aligned along the y-axis and UNK_E_Z for the edge aligned along the z-axis. Finally the corner data is stored in UNK_N. This is shown below for cell (I,J,K) in block LB, for the default case in which IFACE_OFF=0.

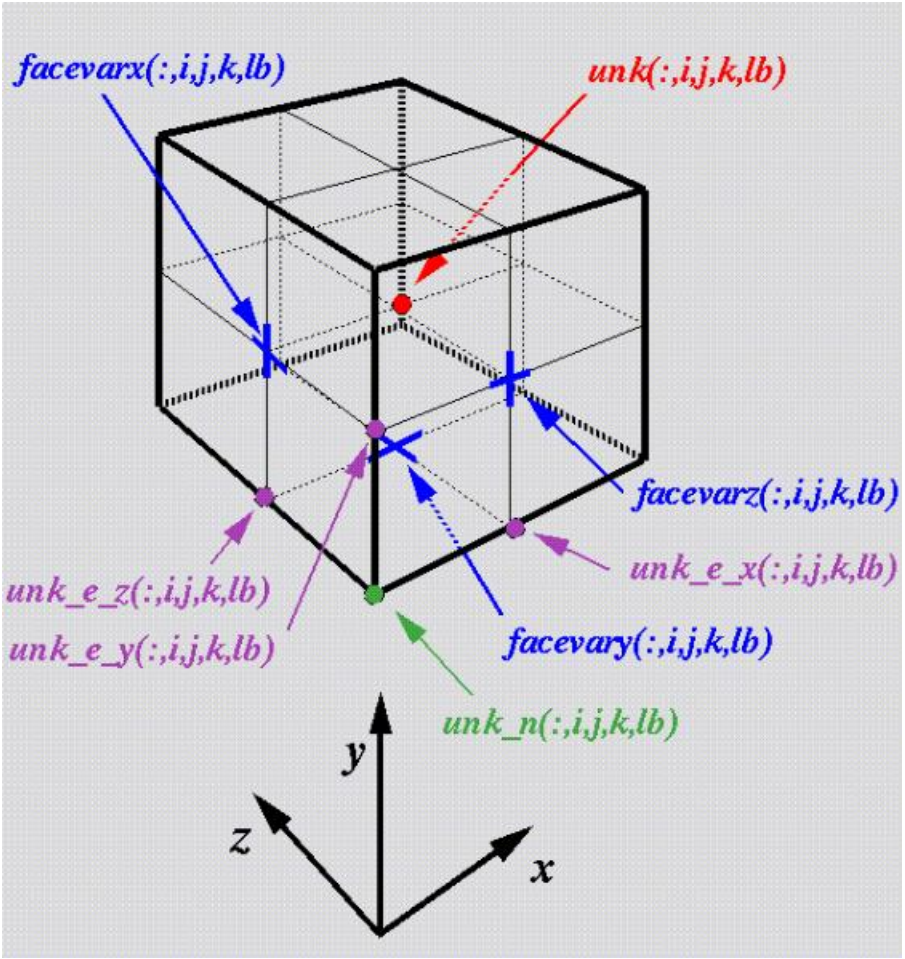


Figure 2.8 Physical location of unk, facevar, unk_e and unk_n variables within grid cell (i,j,k) on grid block lb.(MacNeice et al.2000)

IFACE_OFF is a user-set parameter which defines how the indeces associated with cell centered and non-cell-centered data are related. For example, the default value (0) means that

FACEVARX(:,I,:,:,:) and FACEVARX(:,I+1,:,:,:) bound UNK(:,I,:,:,:). If IFACE_OFF=-1 then FACEVARX(:,I-1,:,:,:) and FACEVARX(:,I,:,:,:) bound UNK(:,I,:,:,:).
The leading dimension of each of these arrays is used to store the different data words of that type within the grid cell. The number of words of each type in each grid cell is defined in PARAMESH_PREPROCESSOR.FH by the variables N_VAR for the cell centered data, N_FACEVAR for the face-centered data, N_VAR_EDGE for edge centered and N_VAR_CORN for corner data. Here are some examples of how various models might be set

Chapter 3-Governing Equations, Finite Difference Discretization, and Test Cases

In this work, the objective is to develop two-dimensional depth-averaged numerical modeling for urban floods with AMR and finite difference method, then to evaluate its accuracy and efficiency compared with the fine grid method.

This work's purpose is to lay a foundation for further improvements in current flood modeling, and better managing urban flood risks based on the AMR.

3.1 Overview of Fluid Flow Simulation

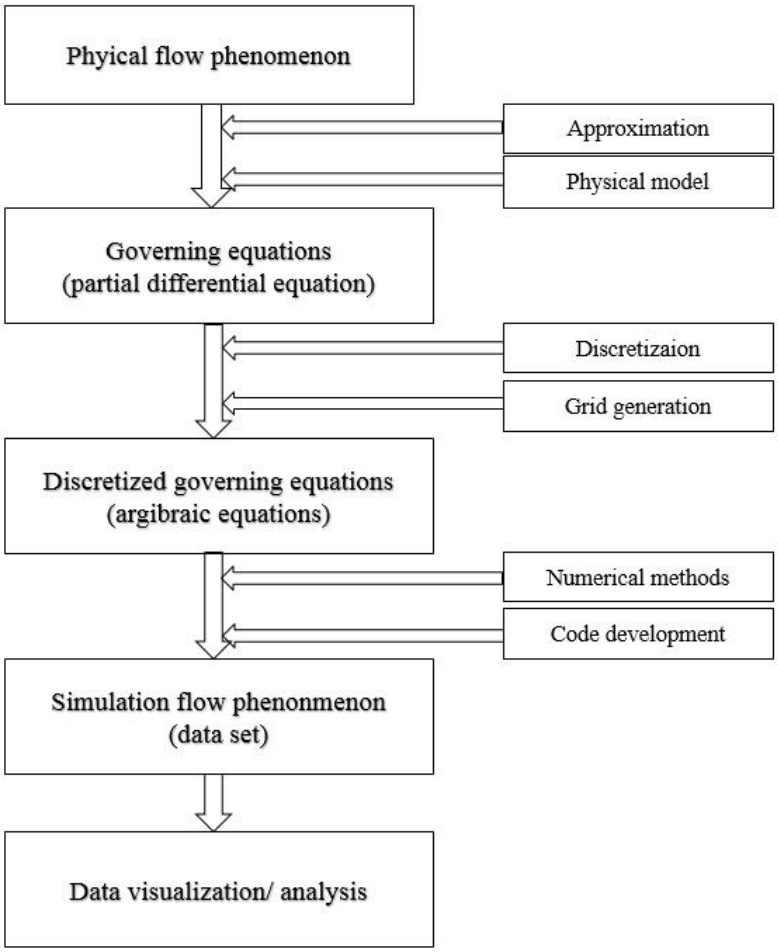


Figure 3.1 General Process of simulating of fluid flows. (Kajishima et al.,2017)

Numerical model of fluid flow follows the steps laid out in Fig3.1.

1. Decide to solve the full Navier-Stokes equations with some approximation like the inviscid

approximation or other approximation to reproduce the physical of interest. Based on these approximations and choices, the governing partial equations will be solved in simulations

2. Discretize the governing equations with the FDM, FVM and FEM, and choose the spatial discretization and time discretization. The numerical schemes are also decided to solve these equations. Then the program should be developed.

3. Through the Numerical model, a large number of numerical values as the solution can be outputted. Hence, graphs and visualizations with computer graphics and animations are used to aid the analysis of simulation results.

3.2 Governing Equations

3.2.1 Advection-Diffusion Equation

Advection-diffusion equation in one dimension

$$\frac{\partial \mathrm{u}}{\partial t} + a\frac{\partial u}{\partial x} = D\frac{\partial^2 u}{\partial x^2} \tag{3.1}$$

Which is also called Burgers' equation. This equation is often used to validate numerical methods because the exact solution is available. If $a = 0$, the diffusion is obtained. If $D = 0$, The advection equation is obtained.

Advection-diffusion equation in two dimension

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} = D\frac{\partial^2 \phi}{\partial x^2} + E\frac{\partial^2 \phi}{\partial y^2}$$

3.2.2 Shallow water equations

In the Cartesian coordinate system, governing equations are given as follows:
Continuity equation:

$$\frac{\partial \mathrm{h}}{\partial t} + \frac{\partial hu}{\partial x} + \frac{\partial hv}{\partial y} = 0 \tag{3.2}$$

Momentum equation in x direction for non-viscous fluid:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -g\frac{\partial h}{\partial x} - g\frac{\partial \eta}{\partial x} - \frac{\tau_x}{\rho h}$$

Momentum equation in y direction for non-viscous fluid:

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -g\frac{\partial h}{\partial y} - g\frac{\partial \eta}{\partial y} - \frac{\tau_y}{\rho h}$$

Where x and y are the spatial coordinate components in the Cartesian coordinate system; t is time; u and v are the depth averaged velocity components in x and y directions, respectively; h is water depth; η is the elevation; g is the gravitational acceleration.

τ_x and τ_y can be obtained from the following the two equations

$$V = \frac{1}{\mathrm{n}} h^{\frac{2}{3}} i^{\frac{1}{2}} \tag{3.4}$$

$$\tau_b = \rho g h i \tag{3.5}$$

Where, τ_b is the bed shear stress; i is the gradient of water channel,slope; V is velocity in flow direction $\sqrt{(u^2+v^2)}$; n is the Manning's roughness coefficient.

$$\tau_b = \frac{\rho g n^2 V^2}{h^{\frac{1}{3}}} \tag{3.6}$$

$$\tau_x = \tau_b \frac{u}{V} = \frac{\rho g n^2 u\sqrt{u^2+v^2}}{h^{\frac{1}{3}}} \tag{3.7}$$

$$\tau_y = \tau_b \frac{u}{V} = \frac{\rho g n^2 v\sqrt{u^2+v^2}}{h^{\frac{1}{3}}} \tag{3.8}$$

The above momentum equations are called the non-conservative form, which will become incompatible in some finite difference scheme. The compatibility of finite difference will be analyzed in chapter 3.3.2.2

In Eq. (3.2) and Eq. (3.3), the advection term is nonlinear. The pressure gradient term represents forcing on a fluid element caused by the spatial change in pressure. The pressure term will be solved by coupling the continuity equation Eq. (3.1) and momentum equation Eq.(3.2) Eq. (3.3).

3.3 Finite-difference Discretization

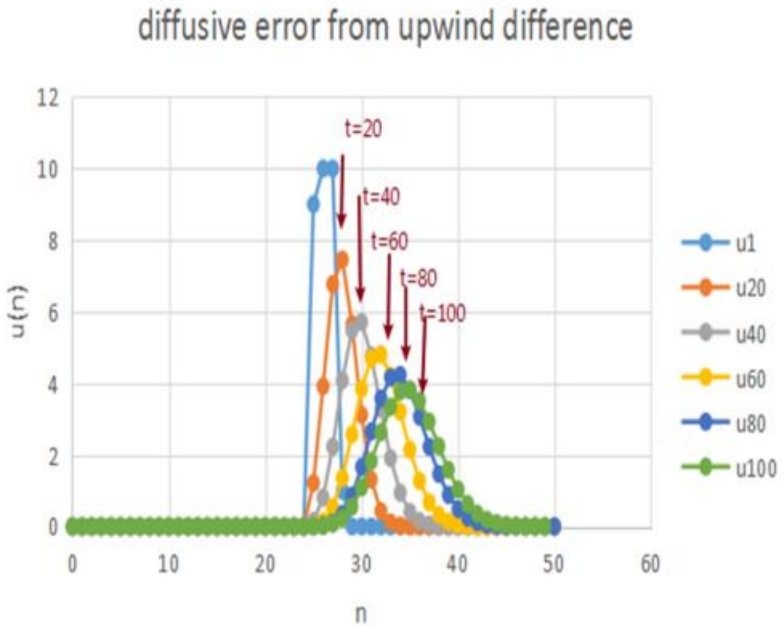


Figure 3.1 Solution of advection using the backward difference under the condition of under the condition of dt =0.02s,dx= 0.2m, propagation c =1m/s,CFL=0.1.

3.3.1 Taylor Series Expansion

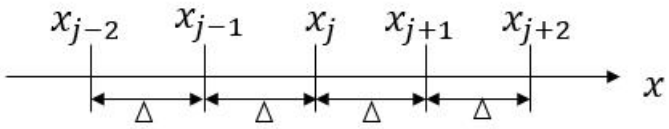


Figure 3.2 Placement of grid points with uniform spacing (one-dimensional)

Let's consider the one dimensional case. Taylor series expansion of a function about a point $x_j = j\Delta$ for uniform mesh, as shown in Fig.3.2. Denoting the spatial derivatives of a continuous function $\Phi(x)$ as

$$\Phi^m(x) = \frac{d^m \Phi(x)}{dx^m} \tag{3.9}$$

The Taylor series expansion about point x evaluated at point $x + \Delta x$ is

$$\Phi(x + \Delta x) = \Phi(x) + \sum_{m=1}^{\infty} \frac{\Delta x^m}{m!} \Phi^m(x) \tag{3.10}$$

Order of accuracy

For example,

$$\sin(x) = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \ldots = x - \frac{1}{3!}x^3 + o(x^5) \quad \text{as} \quad x \to 0 \tag{3.11}$$

with the two term approximation, we can say $\sin(x) \approx x - \frac{1}{3!}x^3$ with an fifth-order truncation error term $o(x^5)$

Usually $o(\Delta x^m)$ with a Δ being a small quantity ≤ 1, if use the logarithmic scales for both x and y-axes, the slope of error curve (convergence curve) becomes m as illustrated in Figure 3.1. The m is called the order of accuracy.

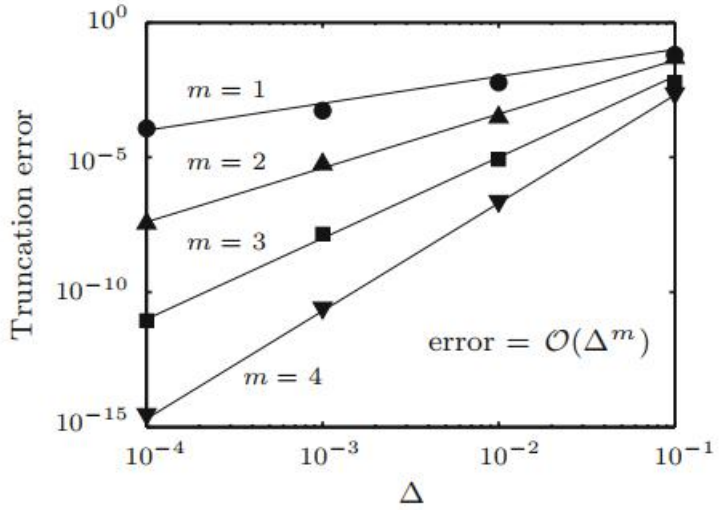


Figure 3.3 Typical convergence of truncation errors. The slopes of the convergence curves on the log-log plot related the orders of accuracy m (Kajishima et al., 2017).

3.3.2 Error properties

3.3.2.1 Central-Difference for uniform grid

$$\phi_{i+1} = \phi_i + (\phi_{i+1} - \phi_i)\left.\frac{d\phi}{dx}\right|_{x=i} + \frac{(\phi_{i+1} - \phi_i)^2}{2!}\left.\frac{d^2\phi}{dx^2}\right|_{x=i} + \frac{(\phi_{i+1} - \phi_i)^3}{3!}\left.\frac{d^3\phi}{dx^3}\right|_{x=i} + \frac{(\phi_{i+1} - \phi_i)^4}{4!}\left.\frac{d^4\phi}{dx^4}\right|_{x=i} + \ldots \quad (3.11)$$

$$\phi_{i-1} = \phi_i + (\phi_{i-1} - \phi_i)\left.\frac{d\phi}{dx}\right|_{x=i} + \frac{(\phi_{i-1} - \phi_i)^2}{2!}\left.\frac{d^2\phi}{dx^2}\right|_{x=i} + \frac{(\phi_{i-1} - \phi_i)^3}{3!}\left.\frac{d^3\phi}{dx^3}\right|_{x=i} + \frac{(\phi_{i-1} - \phi_i)^4}{4!}\left.\frac{d^4\phi}{dx^4}\right|_{x=i} + \ldots \quad (3.12)$$

$$\phi_{i+1} + \phi_{i-1} = 2\phi_i + (\Delta x)^2\left(\frac{d^2\phi}{dx^2}\right) + \frac{(\Delta x)^4}{12}\left(\frac{d^4\phi}{dx^4}\right) + \frac{(\Delta x)^6}{60}\left(\frac{d^6\phi}{dx^6}\right) + \ldots \quad (3.13)$$

$$\frac{d^2\phi}{dx^2} = \frac{\phi_{i+1} + \phi_{i-1} - 2\phi_i}{(\Delta x)^2} - \frac{(\Delta x)^2}{12}\left(\frac{d^4\phi}{dx^4}\right) - \frac{(\Delta x)^4}{60}\left(\frac{d^6\phi}{dx^6}\right) + \ldots \quad (3.14)$$

If the lowest order partial derivative term on the RHS is ‘even’ i.e. 2,4,6 etc. the solution is **dissipative** in nature. Shown in the Figure 3.1, the scheme is dissipative.

$$\frac{d^2\phi}{dx^2} = \frac{\phi_{i+1} + \phi_{i-1} - 2\phi_i}{(\Delta x)^2} + o(\Delta^2) \quad (3.15)$$

In the Eq.(3.15), withe the three point stencil, we can obtain up to the second-order

derivative with finite differencing. The second terms on the right-hand side of Eq.(3.15) represents the **truncation errors**. As we consider the small Δ with bounded high order derivatives, the truncation errors becomes proportional to Δ^2. So in the Eq.(3.15), the second order of accuracy is obtained.

$$\frac{d\phi}{dx} = \frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x} - \frac{(\Delta x)^2}{6}\frac{d^3\phi}{dx^3} \quad (3.16)$$

Whereas if the lowest order partial derivative term on the RHS is ‘odd’ i.e. 1,3,5 etc. the solution is DISPERSIVE in nature. This is shown in the Figure 3.3, we say the scheme is **dispersive.**

$$\frac{d\phi}{dx} = \frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x} + o(\Delta^2) \quad (3.17)$$

In the equation of Eq.(3.16) and Eq.(3.17), the second order of accuracy are obtained.

Figure 3.2 Solution of advection

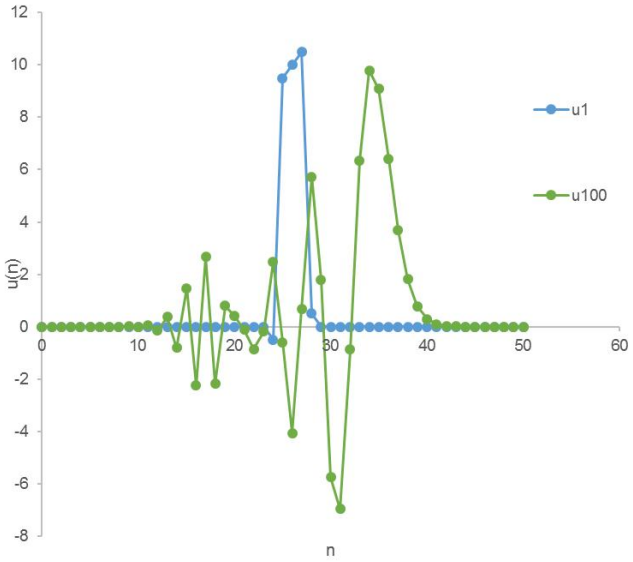


Figure 3.4 Solution of advection using the central difference under the condition of dt =0.02s,dx= 0.2m, propagation c =1m/s,CFL=0.1.

3.3.2.2 Central-Difference at Midpoint

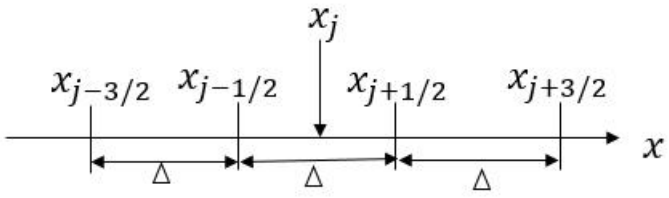


Figure 3.5 Central difference at midpoint on uniform grid(one-dimensional)

As shown in Figure 3.5, . Let us consider a finite-difference formula for a stencil using points of $j \pm \frac{1}{2}, j \pm \frac{3}{2}, \ldots$ centered about x_j. Since the functional values at midpoints are unknown, we use the second-order accurate interpolation and the Taylor series expansion in the derivation of central-difference at midpoints.

Using the values from two adjacent points of $j \pm \frac{1}{2}$, the second-order accurate interpolation and finite-difference can be formulated as

$$\phi_j = \frac{\phi_{j-\frac{1}{2}} + \phi_{j+\frac{1}{2}}}{2} - \frac{\Delta^2}{8}\phi_j'' + o\left(\Delta^4\right) \qquad (3.18)$$

$$\phi_j{}' = \frac{-\phi_{j-\frac{1}{2}} + \phi_{j+\frac{1}{2}}}{2} - \frac{\Delta^2}{24}\phi_j{}^{(3)} + o\left(\Delta^4\right) \tag{3.19}$$

For the second derivative, we should take the first-derivative finite difference of the first-derivative difference at points and $j \pm \frac{1}{2}, j \pm \frac{3}{2}, \ldots$. By taking the finite difference twice in this manner, we obtain for the second-order accurate formulation

$$\phi_j{}'' = \frac{-\phi'_{j-\frac{1}{2}} + \phi'_{j+\frac{1}{2}}}{\Delta} = \frac{\phi_{j-1} - 2\phi_j + \phi_{j+1}}{\Delta^2} \tag{3.20}$$

Which matches the second-order second-derivative difference formula

3.3.2.3 Upwind difference for uniform grid

$$\frac{d\phi}{dx}=\frac{\phi_{i+1}-\phi_i}{\Delta x}-\frac{\Delta x}{2}\frac{d^2\phi}{dx^2}-\frac{(\Delta x)^2}{6}\frac{d^3\phi}{dx^3}+\ldots \quad (3.21)$$

$$\frac{d\phi}{dx}=\frac{\phi_{i+1}-\phi_i}{\Delta x}+o(\Delta) \quad (3.22)$$

$$\frac{d\phi}{dx}=\frac{\phi_i-\phi_{i\text{-}1}}{\Delta x}+\frac{\Delta x}{2}\frac{d^2\phi}{dx^2}-\frac{(\Delta x)^2}{6}\frac{d^3\phi}{dx^3}+\ldots \quad (3.23)$$

$$\frac{d\phi}{dx}=\frac{\phi_i-\phi_{i\text{-}1}}{\Delta x}+o(\Delta) \quad (3.24)$$

In the Eq.(3.22) and Eq.(3.24), by using the upwind scheme and two point stencil, the first order of accuracy are obtained.

3.3.2.4 Discretization for Non-uniform grid

Figure 3.3 illustration of nonuniform grid

For the non-uniform grid, using the central difference method, we can obtain,

$$\phi_{i+1}=\phi_i+\Delta x_R\left.\frac{d\phi}{dx}\right|_{x=i}+\frac{(\Delta x_R)^2}{2}\left.\frac{d^2\phi}{dx^2}\right|_{x=i}+\frac{(\Delta x_R)^3}{3!}\left.\frac{d^3\phi}{dx^3}\right|_{x=i}+\ldots \quad (3.25)$$

$$\phi_{i\text{-}1}=\phi_i-\Delta x_L\left.\frac{d\phi}{dx}\right|_{x=i}+\frac{(\Delta x_L)^2}{2}\left.\frac{d^2\phi}{dx^2}\right|_{x=i}-\frac{(\Delta x_L)^3}{3!}\left.\frac{d^3\phi}{dx^3}\right|_{x=i}+\ldots \quad (3.26)$$

To obtain the $\frac{d^2\phi}{dx^2}$, the Eq.(3.25) multiple the value of Δx_L, the Eq.(3.26) multiple the value of Δx_R, then we can get the truncation error,

$$\delta=-\frac{1}{3}(\Delta x_R-\Delta x_L)\frac{d^3\phi}{dx^3} \quad (3.27)$$

In the uniform grid, the truncation is

$$\delta=-\frac{(\Delta x)^2}{12}\frac{d^4\phi}{dx^4} \quad (3.28)$$

From the Eq.(3.27) and Eq.(3.28), we could see that the second order of accuracy, using the central difference scheme, becomes the first order accuracy, because the Δx_L and Δx_R are different and the

$\frac{d^3\phi}{dx^3}$ term can not be removed.

3.2.2.5 Conservative, non-conservative form, and compatibility of finite difference

The mass conservation equation is also referred to as the continuity equation and is

$$\frac{\partial \rho}{\partial t} + \nabla \bullet (\rho u) = 0 \tag{3.29}$$

for flows without sinks or sources. We can consider the time rate of change and transport of some physical quantity φ by the velocity field u to be decomposed as

$$\frac{\partial \rho\phi}{\partial t} + \nabla \bullet (\rho u\phi) = \rho\left(\frac{\partial \phi}{\partial t} + \nabla \bullet (u\phi)\right) + \phi\left(\frac{\partial \rho}{\partial t} + \nabla \bullet (\rho u)\right) \tag{3.30}$$

where the second term on the right-hand side becomes zero due to continuity, Eq.(1.17). Accordingly, we have

$$\frac{\partial (\rho\phi)}{\partial t} + \nabla \bullet (\rho u\phi) = \rho \frac{D\phi}{Dt} = \rho\left(\frac{\partial \phi}{\partial t} + u \bullet \nabla(\phi)\right) \tag{3.31}$$

The second term on the left-hand side of Eq.(3.31) represents advection by the divergence of ρuφ. For that reason, this form is called the divergence form (**conservative form**). The second term on the right-hand side is expressed as the inner product of the velocity u and the gradient ∇φ, and is thus referred to as the gradient form or advective form (**non-conservative form**). For correctly discretized equations, the identity should hold, which makes the use of the term non-conservative form for the right-hand side of Eq.(3.31) somewhat misleading. Depending on the discretization schemes, this relation may not hold. It would appear more appropriate to use the term non-conservative to describe incompatible discretization schemes rather than the form of the right-hand side of Eq.(3.31).

For example, differentiation rules for continuous functions f and g must hold also in a discrete manner.

$$\frac{\partial (fg)}{\partial x} = f\frac{\partial g}{\partial x} + g\frac{\partial f}{\partial x} \tag{3.32}$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right) \tag{3.33}$$

$$\frac{\partial^2 f}{\partial x \partial f} = \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial y}\right) = \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right) \tag{3.34}$$

If those finite-difference schemes that satisfy the derivative properties discretely, it is called this schemes is **compatible.**

Using the second-order differencing from Eq.(3.16) and Eq.(3.17) for the differentiation of a product of two functions f and g shown in Eq.(3.32), (3.35)

$$\frac{-(fg)_{j-1}+(fg)_{j+1}}{2\Delta} \neq f_j \frac{-g_{j-1}+g_{j+1}}{2\Delta} + \frac{-f_{j-1}+f_{j+1}}{2\Delta} g_j$$

Which shows the differential rule does not hold discretely for the chosen differencing scheme.

The Eq.(3.33) also does not hold discretely when the first-derivative finite differencing is applied twice, we could obtain

$$\frac{1}{2\Delta}\left(-\frac{-f_{j-2}+f_j}{2\Delta}+\frac{-f_j+f_{j+2}}{2\Delta}\right)=\frac{f_{j-2}-2f_j+f_{j+2}}{4\Delta^2} \neq \frac{f_{j-1}-2f_j+f_{j+1}}{\Delta^2} \qquad (3.36)$$

Which is not equivalent to Eq.(3.15) that directly derived the finite-difference scheme for the second derivative. Thus, the first derivative finite difference scheme about x_j is not compatible if the stencil is based on $j \pm 1, j \pm 2, \ldots$

For upwind scheme using the first order differencing form Eq.(3.24)

$$\frac{d(fg)}{dx}=\frac{-(fg)_{j-1}+(fg)_j}{\Delta}=f_{j-\frac{1}{2}}\frac{-g_{j-1}+g_j}{\Delta}+\frac{-f_{j-1}+f_j}{\Delta}g_{j-\frac{1}{2}} \qquad (3.37)$$

Which is equivalent to Eq.(3.24). So it is compatible.

Using interpolation and difference operations from the Eq(3.19), we obtain

$$\left[\frac{\partial(fg)}{\partial x}\right]_j=\frac{1}{2}\left\{\left[\frac{\partial(fg)}{\partial x}\right]_{j-1/2}+\left[\frac{\partial(fg)}{\partial x}\right]_{j+1/2}\right\}=\frac{1}{2}\left(\frac{-f_{j-1}g_{j-1}+f_ig_i}{\Delta}+\frac{-f_jg_j+f_{j+1}g_{j+1}}{\Delta}\right) \qquad (3.38)$$

Which agrees with

$$\left[f\frac{\partial g}{\partial x}+\frac{\partial f}{\partial x}g\right]_j=\frac{1}{2}\left\{\left[f\frac{\partial g}{\partial x}+\frac{\partial f}{\partial x}g\right]_{j-1/2}+\left[f\frac{\partial g}{\partial x}+\frac{\partial f}{\partial x}g\right]_{j+1/2}\right\} \qquad (3.39)$$

$$=\frac{1}{2}\left[\left(\frac{f_{j-1}+f_j}{2}\frac{-g_{j-1}+g_i}{\Delta}+\frac{-f_{j-1}+f_j}{\Delta}\frac{g_{j-1}+g_i}{\Delta}\right)+\left(\frac{f_{j+1}+f_j}{2}\frac{-g_j+g_{j+1}}{\Delta}+\frac{-f_j+f_{j+1}}{\Delta}\frac{g_j+g_{j+1}}{\Delta}\right)\right]$$

The above discretization exhibits compatibility for differentiation in discrete sense. It is also equivalent to the left hand side of Eq.(3.35),

$$\left[\frac{\partial(fg)}{\partial x}\right]_j = \frac{1}{\Delta}\left(-[fg]_{j-½} + [fg]_{j+½}\right) = \frac{1}{\Delta}\left(\frac{-f_{j-1}g_{j-1} + f_i g_i}{2} + \frac{-f_j g_j + f_{j+1}g_{j+1}}{2}\right) \quad (3.40)$$

Summary

By interpolating the difference approximation at $x_{j\pm½}$ to determine the derivative at x_j , the compatibility will be satisfied.

3.3.3 Courant-Friedrichs-Lewy condition(CFL) (from wiki)

For one-dimensional case, the CFL has the following form

$$C \equiv \frac{u\Delta t}{\Delta x} \le 1 \quad (3.24)$$

Where the dimensionless number C is called the Courant number,
u is the magnitude of the velocity(whose dimension is length/time)
Δt is the time step (whose dimension is time)
Δx is the length interval (whose dimension is length)

The value of $C_{\max}$ changes with the method used to solve the discretised equation, especially depending on whether the method is explicit or implicit. If an explicit (time-marching) solver is used, then typically $C_{\max}$ =1. Implicit (matrix) solver are usually less sensitive to numerical instabilities and so larger values of $C_{\max}$ may be tolerated.

In the two-dimensional case, the CFL condition becomes

$$C \equiv \frac{u_x\Delta t}{\Delta x} + \frac{u_y\Delta t}{\Delta y} \le C_{\max} \quad (3.25)$$

3.3.4 Boundary Conditions

Summary of boundary conditions for the unknown function u, constants c_0 and c_1 specified by boundary conditions, and known scalar functions f and g specified by the boundary conditions.

Table 3.1 Boundary types

Name	Form on 1st part of boundary	From on 2nd part of boundary
Dirichlet	$u = f$	
Neumann	$\frac{\partial u}{\partial x} = f$	
Robin	$c_0 u + c_1 \frac{\partial u}{\partial x} = f$	
Mixed	$u = f$	$c_0 u + c_1 \frac{\partial u}{\partial x} = f$
Cauchy	Both $u = f$ and $c_0 \frac{\partial u}{\partial x} = g$	

In numerical modeling for the fluid dynamics, the Dirichlet boundary conditions usually is used for the wall or the inlet condition. The Neumann boundary conditions are usually applied into the open boundary for flows.

For example, Neumann boundary for open boundary,

$$\frac{\partial y}{\partial n} = 0, \Delta y = 0 \tag{3.26}$$

3.4 Numerical schemes for incompressible flow on Cartesian grids

3.4.1 Upwind scheme

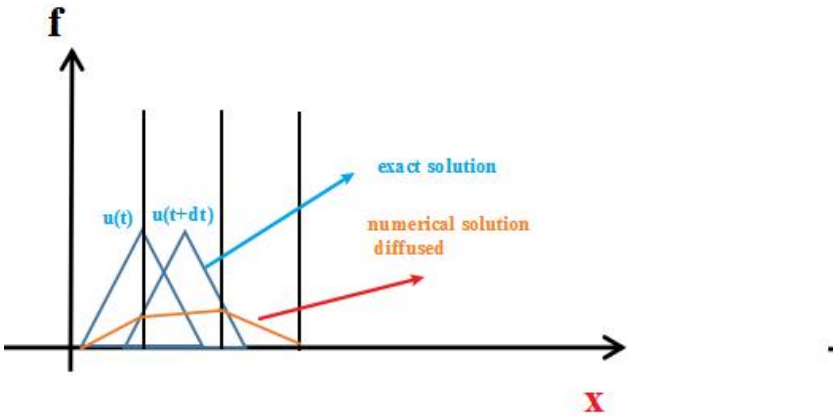


Figure 3.4. The illustration of the solution's diffusion solved by the upwind scheme

For computing the advection term, we can make conclusions as following.

Table 3.2 backward difference, forward difference, central difference for advection term

	U>0	U<0
Backward	Stable	Unstable
$\frac{\partial h}{\partial x}=\frac{h(i)-h(i-1)}{\Delta x}$	Inaccurate	Inaccurate
Forward	Unstable	Stable
$\frac{\partial h}{\partial x}=\frac{h(i+1)-h(i)}{\Delta x}$	Inaccurate	Inaccurate
Central	Unstable	Unstable
$\frac{\partial h}{\partial x}=\frac{h(i+1)-h(i-1)}{2\Delta x}$	Accurate	Accurate

So for the stable solution, the upwind schemes, which is described by Gentry, Martin and Daly(1996) and Runchal and wolfshtein (1969), attempt to discretize hyperbolic partial differential equations by using differencing biased in the direction. The first order upwind scheme is given by

$$u\frac{\partial h}{\partial x} = u\frac{h(i)-h(i-1)}{\Delta x}; u > 0 \qquad (3.27)$$

$$u\frac{\partial h}{\partial x} = u\frac{h(i+1)-h(i)}{\Delta x}; u < 0 \qquad (3.28)$$

If the carrier velocity u is positive, the backward space difference scheme is used. If u is negative, a forward difference scheme must be used to assure stability. This upwind scheme is only calculated in the convective term, the formulation of diffusion term remains unchanged.
Upwind scheme in one equation with first order accuracy.

$$u\frac{\partial h}{\partial x} = \frac{1}{2}(u+|u|)\frac{h(i)-h(i-1)}{\Delta x} + \frac{1}{2}(u-|u|)\frac{h(i+1)-h(i)}{\Delta x} \qquad (3.29)$$

3.4.2 Constrained Interpolation Profile Method (CIP)

To solve the advection term, the high-order Godunov scheme known as the Constrained Interpolation Profile Scheme (CIP) proposed by Takewakiet al.(1985) is used.
Yabe et.al (2001) explained the strategy of CIP method by using an advection equation.

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} = 0 \qquad (3.30)$$

As we know, using the first order upwind scheme the numerical diffusion arises when we construct the profile by the linear interpolation. If we use a quadratic polynomial for the interpolation, the model suffers from overshooting, which is called the Lax-Wendroff scheme or the Leith scheme.
The accuracy declines, because we neglect the behavior of the solution inside of the grid cell and merely follow the smoothness of the solution. So the CIP concept is to develop a method incorporating the real solution into the profile within a grid cell.
The profile is as shown below. If we differentiate the above equation with spatial variable x,we get

$$\frac{\partial g}{\partial t} + u\frac{\partial g}{\partial x} = -\frac{\partial u}{\partial x}g \qquad (3.31)$$

where the g represents the spatial derivative of $\partial f / \partial x$. if two values of f and g are given at two grids points, the profile between these points can be interpolated by the cubic polynomial $F(x) = ax^3 + bx^2 + cx + d$. thus , the profile at the $n+1$ step can be obtained by shifting the profile by $u\Delta t$ so that

$$f^{n+1} = F(x - u\Delta t) \qquad (3.32)$$

$$g^{n+1} = dF(x - u\Delta t)/dx \qquad (3.33)$$

$$a_i = \frac{g_i + g_{iup}}{D^2} + \frac{2(f_i - f_{iup})}{D^3} \qquad (3.34)$$

$$b_i = \frac{3(f_{iup} - f_i)}{D^2} - \frac{2g_i + g_{iup}}{D} \qquad (3.35)$$

(3.36)

$$f_i^{n+1} = a_i\xi^3 + b_i\xi^2 + g_i^n\xi + f_i^n$$

$$g_i^{n+1} = 3a_i\xi^2 + 2b_i\xi + g_i^n \qquad (3.37)$$

where we define ξ = -uΔt . $D = -\Delta x, iup = i-1$ for $u \geq 0$ and $D = \Delta x, iup = i+1$ for $u < 0$.

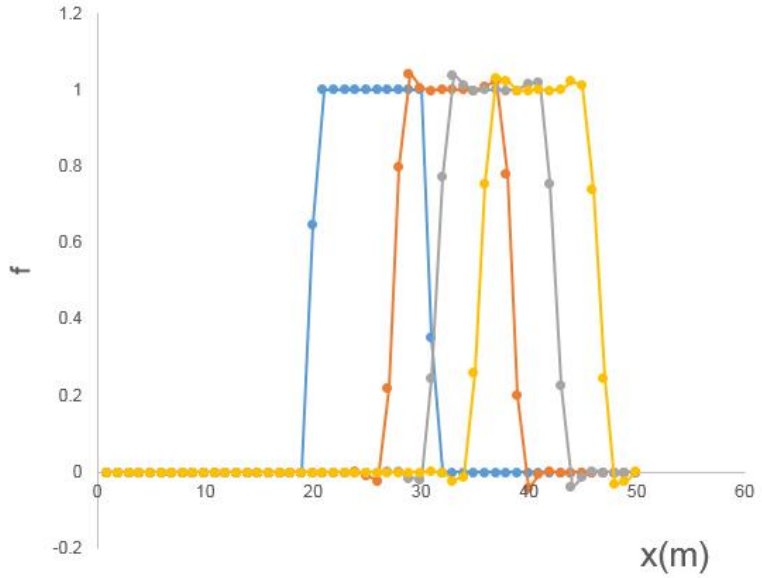


Figure 3.5 The solution of advection solved by CIP method under the conditions of dx = 1.0m ,dt = 0.2s,u = 2m/s, CFL= 0.4.

Which show the cip can predict the value in grid cell well

3.5 Test case1: advection and diffusion equation with AMR

Advection-diffusion

$$\frac{\partial \mathrm{f}}{\partial t}+u\frac{\partial f}{\partial x}+v\frac{\partial f}{\partial y}=D_1\frac{\partial^2 f}{\partial^2 x}+D_2\frac{\partial^2 f}{\partial^2 \mathrm{y}}$$

Explicit Euler's method using forward time central space sheme (FTCS)

$$\frac{f_{i,j}^{n+1}-f_{i,j}^{n}}{\Delta t}=-u\frac{f_{i+1,j}^{n}-f_{i-1,j}^{n}}{2\Delta x}-v\frac{f_{i,j+1}^{n}-f_{i,j-1}^{n}}{2\Delta y}$$
$$+D_1\frac{f_{i+1,j}^{n}-2f_{i,j}^{n}+f_{i-1,j}^{n}}{\Delta x^2}+D_2\frac{f_{i,j+1}^{n}-2f_{i,j}^{n}+f_{i,j-1}^{n}}{\Delta y^2}$$

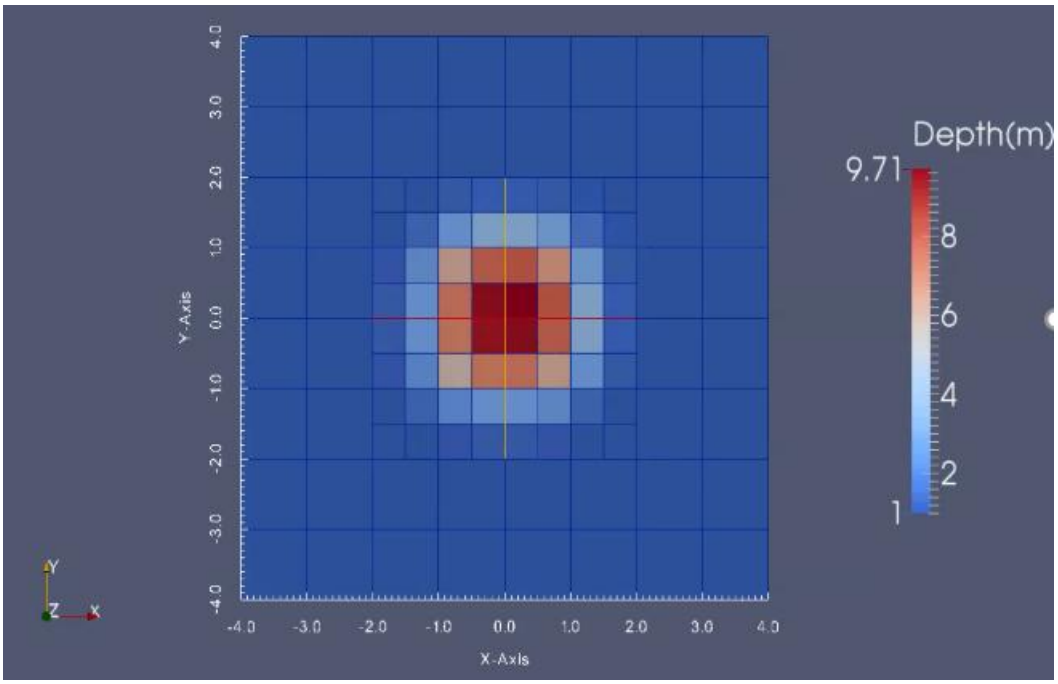
Calculation condition

$$\begin{cases} f=10, if\left(-0.5<x,y<0.5,\right) \\ f=0, otherwise \end{cases}$$

$$u=v=0.5$$
$$D_1=D_2=1$$

Periodic boundary condition

Result:



3.6 Test case2: Karman vortex street with AMR(upwind scheme)

Governing equations

Momentum in x-direction

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-g\frac{\partial h}{\partial x}-g\frac{\partial \eta}{\partial x}-\frac{gn^2u\sqrt{u^2+v^2}}{h^{4/3}}$$

Momentum in y-direction

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-g\frac{\partial h}{\partial y}-g\frac{\partial \eta}{\partial y}-\frac{gn^2v\sqrt{u^2+v^2}}{h^{4/3}}$$

Continuity

$$\frac{\partial h}{\partial t}+\frac{\partial (uh)}{\partial x}+\frac{\partial (vh)}{\partial y}=0$$

Calculation conditions

$$\mathrm{n}=0.3$$
$$\mathrm{i}=0.0001$$
$$Q=1$$

$$h_0=(\frac{nQ}{i^{1/2}})$$

$$\mathrm{V}=\frac{1}{\mathrm{n}}*h^{\frac{2}{3}}*i^{\frac{1}{2}}$$

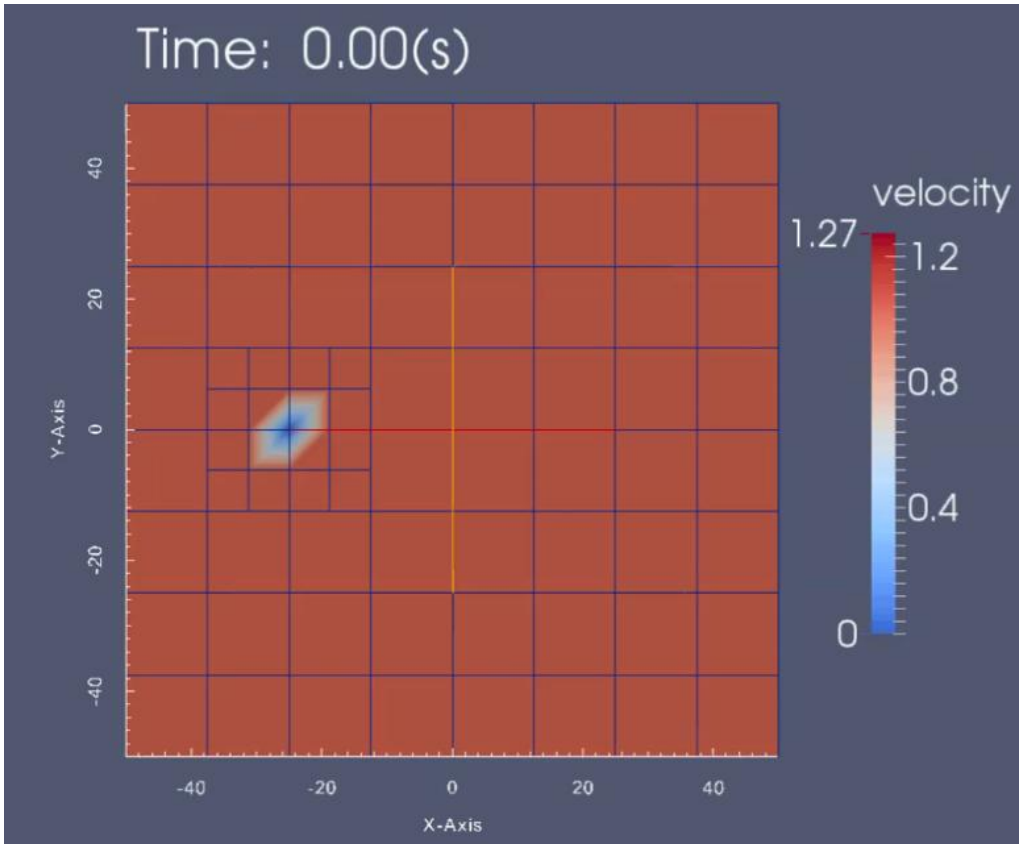
$$\tau=\rho \mathrm{g}\frac{n^2\mathrm{V}^2}{\mathrm{h}^{1/3}}$$

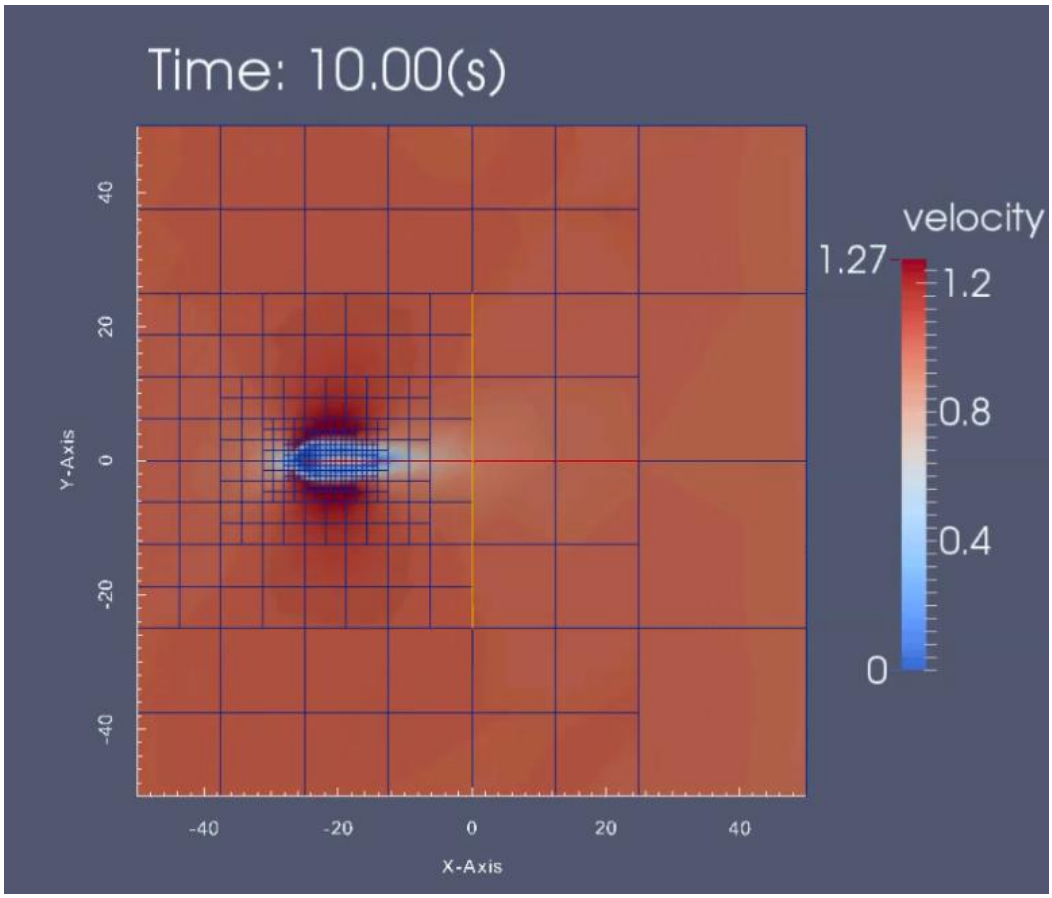
Rectangular center location:(-25,0)

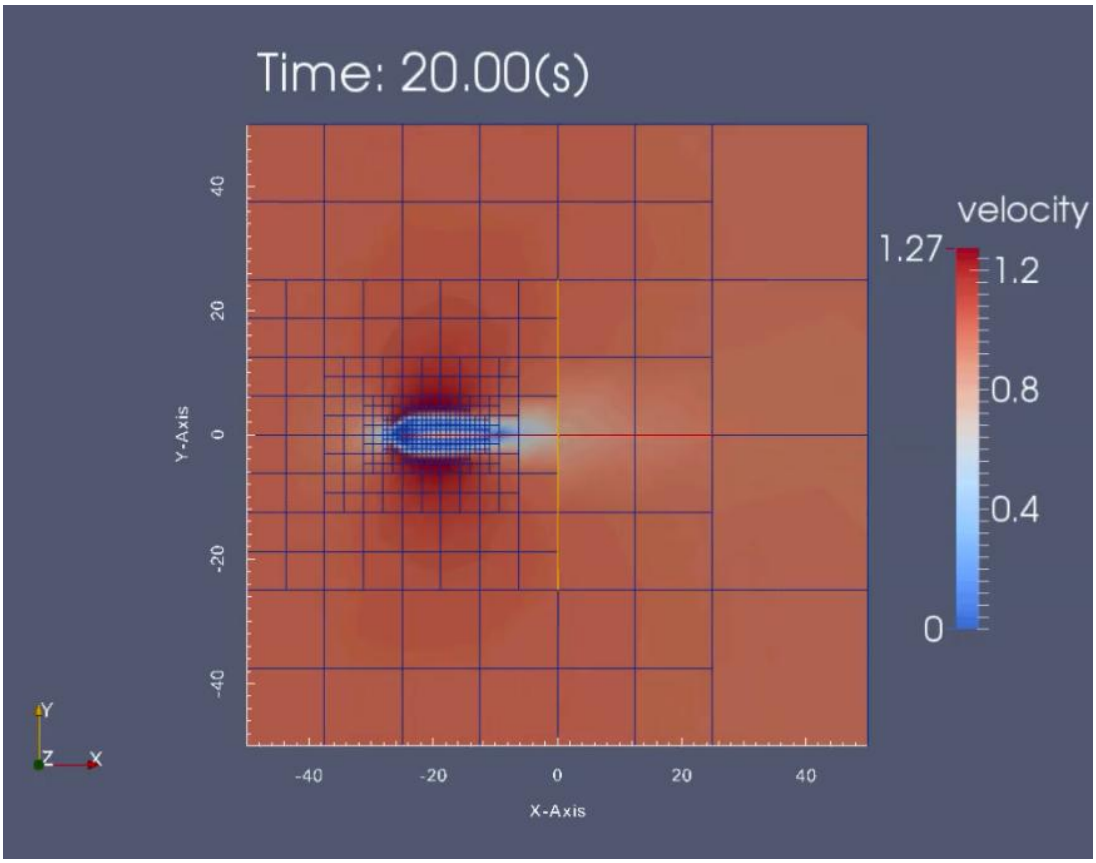
Rectangular size:4x4 m^2

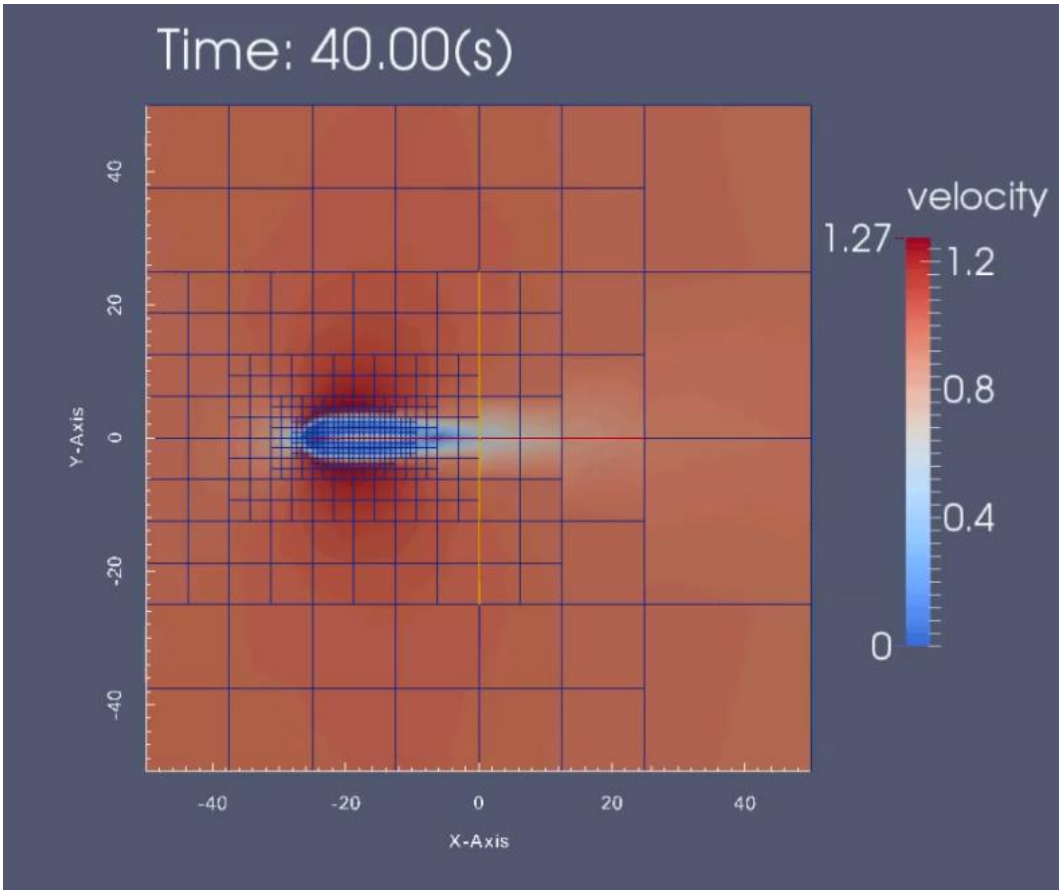
Domain: 100mx100m

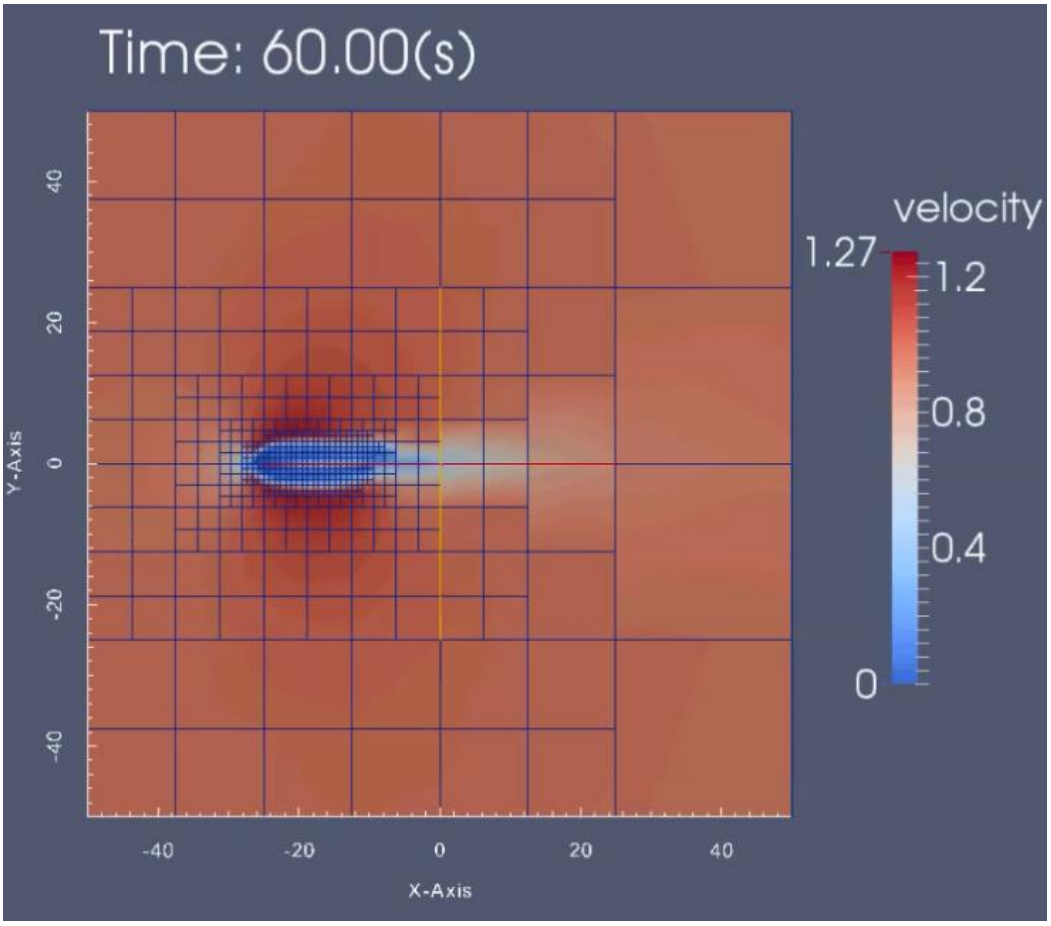
Result

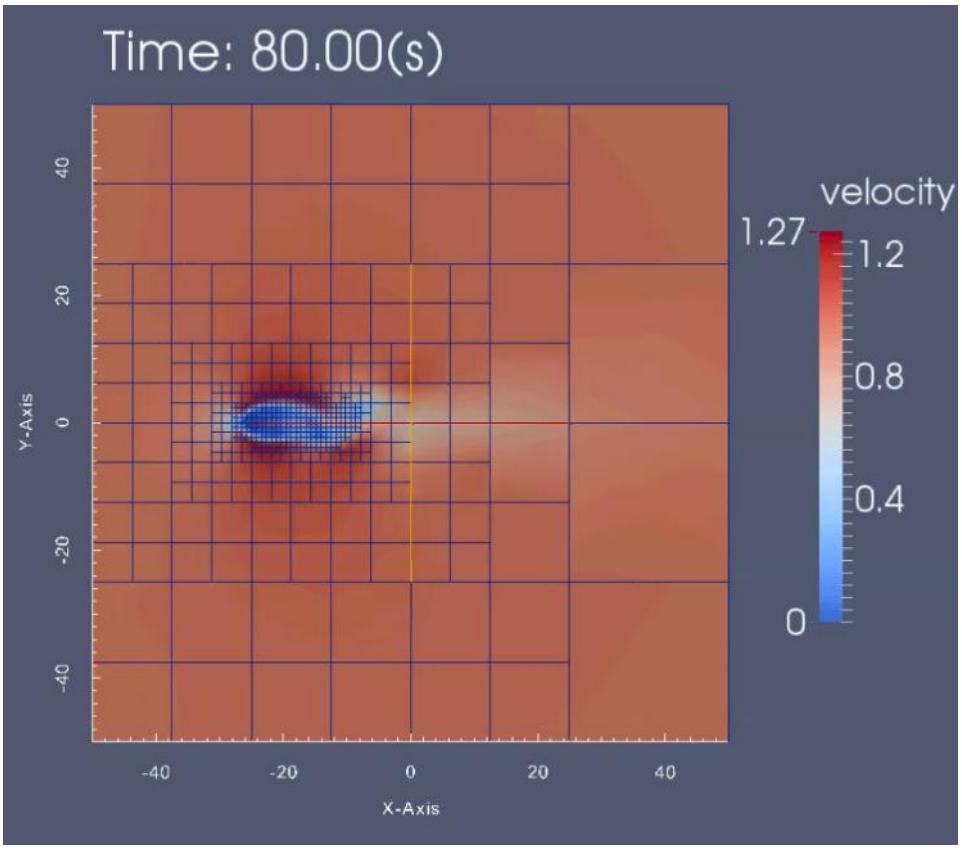


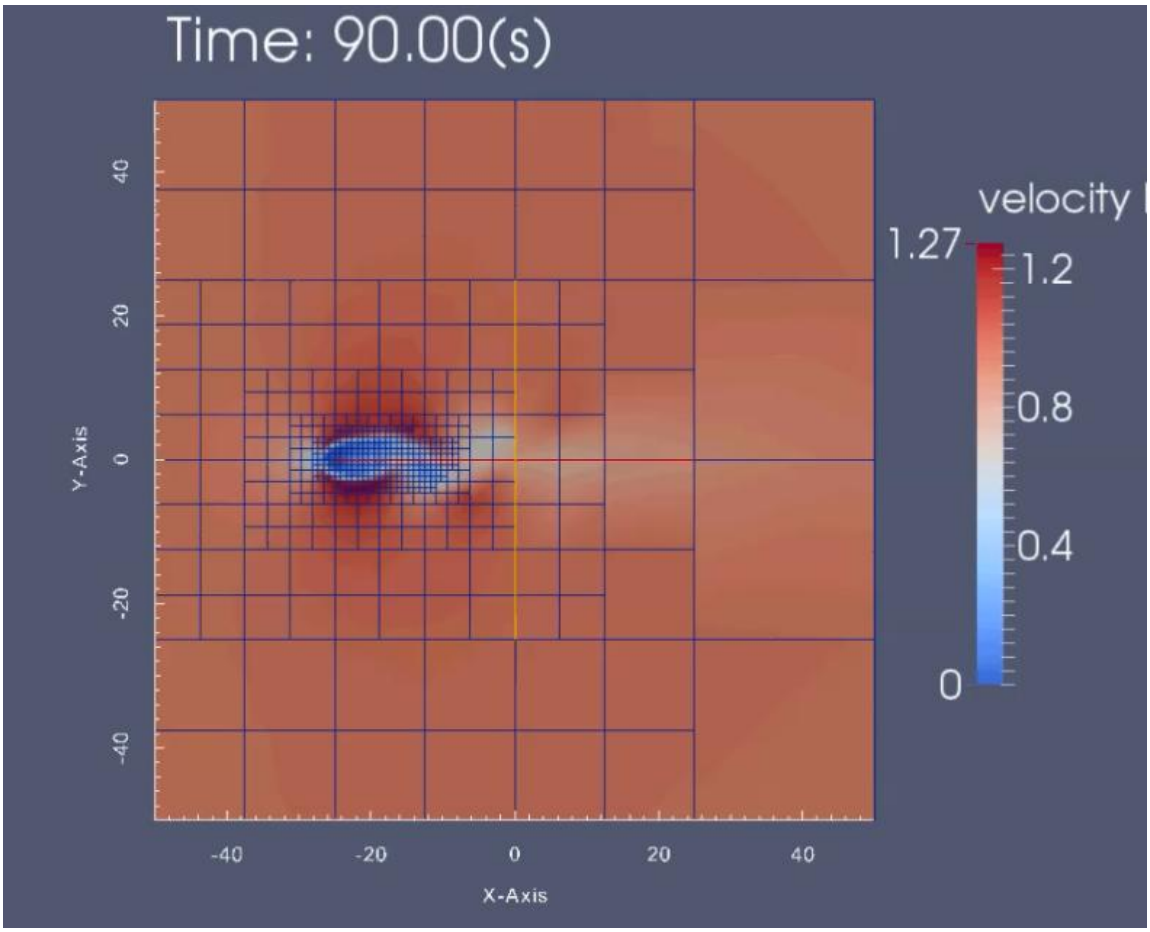

Time: 10.00(s)
velocity
1.27
1.2
0.8
0.4
0
Y-Axis
X-Axis
-40
-20
0
20
40

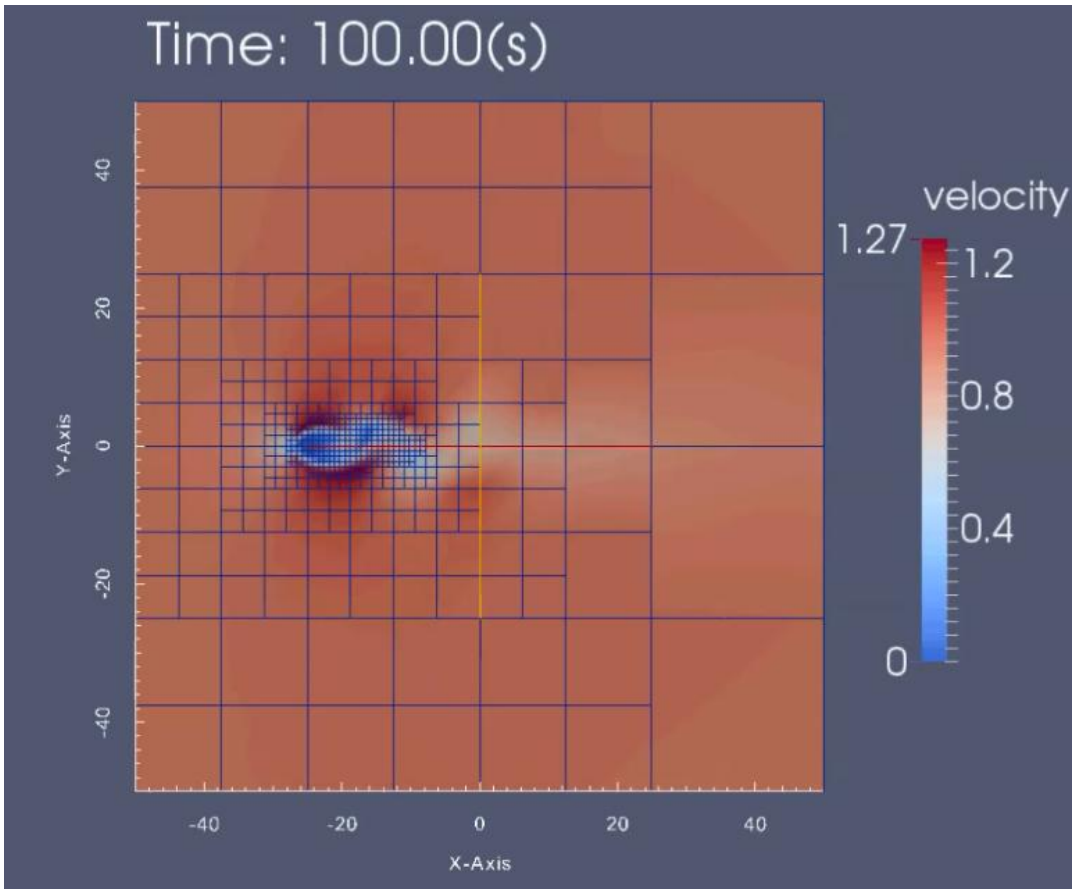

Time: 20.00(s)
velocity
1.27
1.2
0.8
0.4
0
Y-Axis
X-Axis
-40
-20
0
20
40
Y
Z
X


Time: 40.00(s)
velocity
1.27
1.2
0.8
0.4
0
Y-Axis
40
20
0
-20
-40
-40
-20
0
20
40
X-Axis


Time: 60.00(s)
velocity
1.27
1.2
0.8
0.4
0
Y-Axis
X-Axis
40
20
0
-20
-40
-40
-20
0
20
40


Time: 80.00(s)
velocity
1.27
1.2
0.8
0.4
0
Y-Axis
X-Axis
40
20
0
-20
-40
-40
-20
0
20
40


Time: 90.00(s)
velocity
1.27
1.2
0.8
0.4
0
Y-Axis
40
20
0
-20
-40
-40
-20
0
20
40
X-Axis


Time: 100.00(s)
velocity
1.27
1.2
0.8
0.4
0
Y-Axis
40
20
0
-20
-40
-40
-20
0
20
40
X-Axis

Chapter 4. Application :Urban Flood Modeling with Adaptive Mesh Refinement

In this Chapter, we propose an urban flood modeling framework by using a high-accuracy shallow water flow model and the AMR method. The shallow water flow model is solved by using separation technique and the CIP method for advection term to minimize the numerical diffusion. We then implement the AMR method into the proposed shallow flow modeling. The model is applied to a case of urban flooding in Sapporo city, Japan. For this simulation, we use a terrain elevation data that includes the building height and the position, which is complicated enough to check robustness of the proposed model. We also investigate the various refinement criteria to observe the performance of flood modeling with AMR.

4.1 Target City and Geometries

Figure 4.1 Urban area of Sapporo divided by building coverage (Morikawa (2019)

Showed in Figure4.1, the actual building coverage is calculated form the city planning map in 1999, and reproduced by arranging bricks based on the actual building coverage. The actual building coverage is set to 70%, 40%, and 35%. the read area shows the building coverage ratio of 70%, the blue area represents the the coverage ratio of 40%, and the green area shows the coverage ratio of 35%. (Morikawa (2019)

Table-2.3 Sizes of modeled terrain structures

Coverage ratio	Size of modeled terrain structure
70%	120m x 120m
40%	50m x 50m
35%	40m x 40m

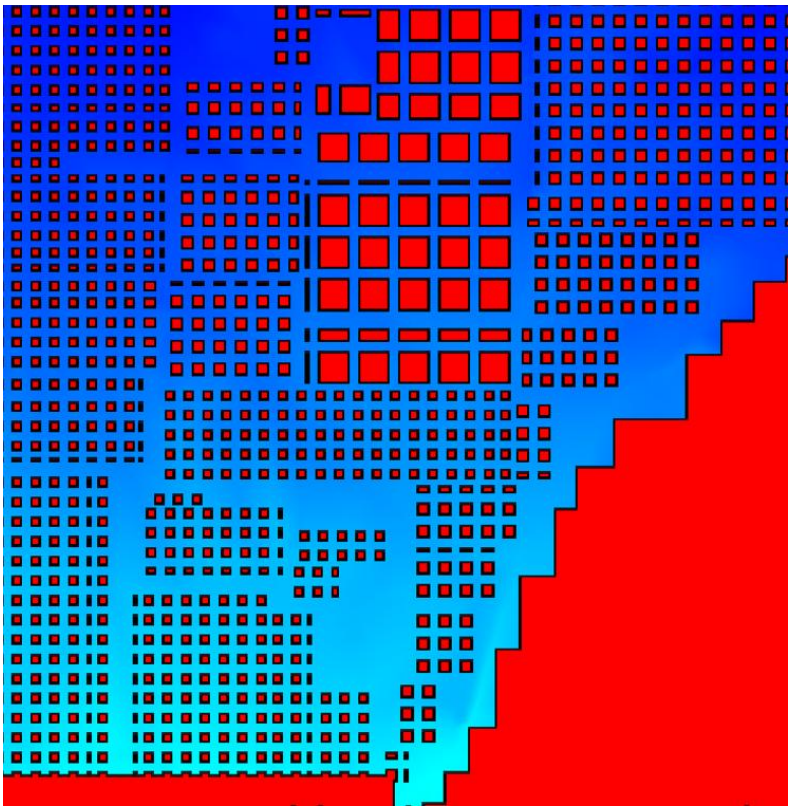


Figure 4.2 Modeled terrain

a) b)

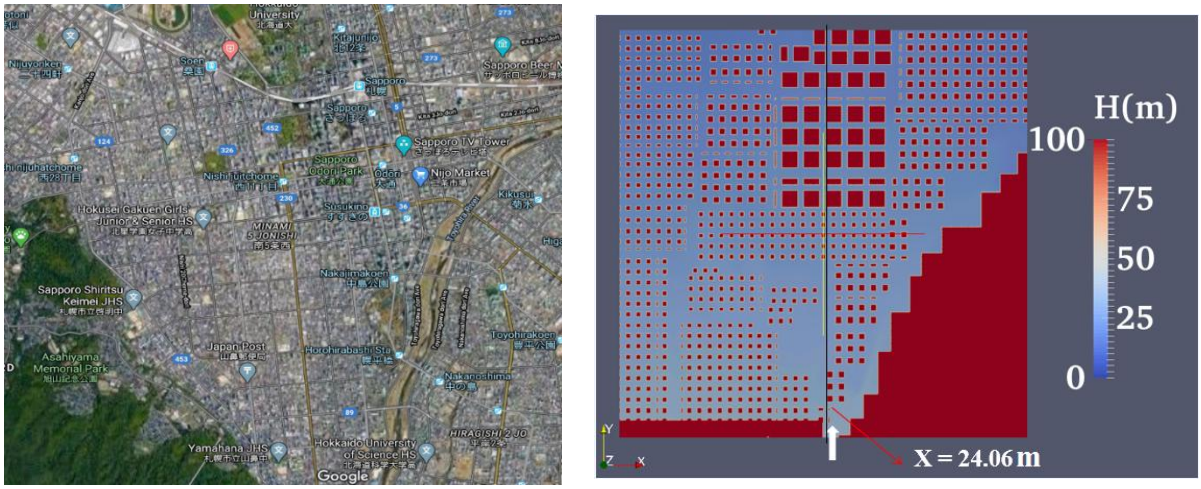


Figure 4.3a) The center of Sapporo city, Japan. 4.3b) Modeled elevation data of the objective area .

The proposed flood model with AMR is applied to a case of flooding in Sapporo, Japan. The objective area of this study is shown in Fig.4.3a. Fig.1a shows that the objective area is highly urbanized by a lot of buildings. Showed in Figure 4.3b, the elevation data set of which the special resolution is 10m in each direction is developed based on the laser profile elevation measurement, and the building is considered as a high elevation area on the elevation data as shown in Fig.4.3b. The white arrow in Figure 4.3b) means the inflow location, and the simulation results among fine grid case, AMR cases and coarse cases will be compared along the black line.

4.2 Computational model

The governing equations are introduced in chapter 3.1. To solve this governing equations, the separation technique and CIP method introduced in chapter 3.3 are adopted for the urban flood modeling. The momentum equations are split into the non-advection phase and advection phase, which is showed below.

Non-advection phase:

$$\frac{\partial u}{\partial t} = -g\frac{\partial H}{\partial x} - \frac{\tau_x}{\rho h} \tag{4.1}$$

$$\frac{\partial v}{\partial t} = -g\frac{\partial H}{\partial y} - \frac{\tau_y}{\rho h} \tag{4.2}$$

Advection phase:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = 0 \tag{4.3}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = 0 \tag{4.4}$$

We use the semi-implicit method for the non-advection phase and continuity equations. If the error between new water depth and old water depth are small, the iteration of water depth and velocity in non-advection phase of momentum equation and continuity equation will be stopped. The computation procedure is denoted in Figure 3.10.

4.3 Staggered Grid

As shown in Figure 4.4, the water depth are stored in the cell center, the velocity are stored in the cell face respectively. The flux of unit width, is also stored in the cell face.

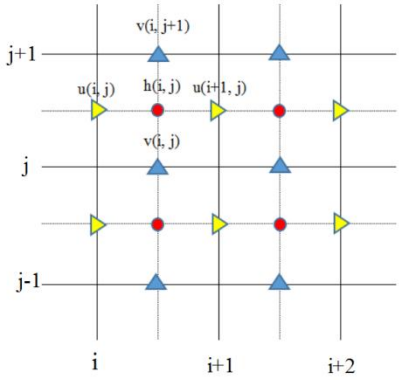


Figure 4.4 The location of the *h* (water depth), *u*, *v* (flow velocities in *x* and *y* directions) on the staggered grid.

Why use the staggered grid

For example, Let's consider discretizing the pressure Poisson equation on a regular grid. The continuity equation about point (x_j, y_j) in Figure. 3.8 becomes

$$\frac{-u_{i-1,j}^{n+1} + u_{i+1,j}^{n+1}}{2\Delta x} + \frac{-v_{i,j-1}^{n+1} + v_{i,j+1}^{n+1}}{2\Delta y} = 0 \quad \text{(a)}$$

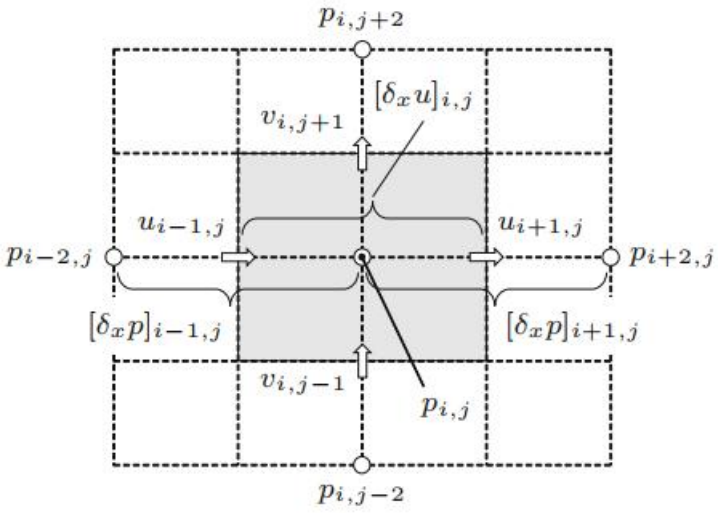


Figure 4.5. Velocity and pressure coupling on a regular grid (Kajishima et al., 2017)

Eq.(a) considers a region, bounded by points $(\pm \Delta x, \pm \Delta y)$ that's a double in size in both directions, as the control volume. Because Eq.(a) does not represent a discretization scheme based on a control volume that shares a cell face with adjacent cells, it is not suitable for flux balancing. In the final simplifies mark and cell method we add the gradient of ϕ to values of $u_{i\pm1,j}$ and $u_{i,j\pm1}$ in Eq. (a):

$$\begin{cases} u_{i-1,j}^{n+1} = u_{i-1,j}^{p} - \Delta t \dfrac{-\phi_{i-2,j} + \phi_{i,j}}{2\Delta x} \\ u_{i+1,j}^{n+1} = u_{i+1,j}^{p} - \Delta t \dfrac{-\phi_{i,j} + \phi_{i+2,j}}{2\Delta x} \end{cases} \quad \text{(b)}$$

$$\begin{cases} v_{i,j-1}^{n+1} = v_{i,j-1}^{p} - \Delta t \dfrac{-\phi_{i,j-2} + \phi_{i,j}}{2\Delta y} \\ v_{i,j+1}^{n+1} = v_{i,j}^{p} - \Delta t \dfrac{-\phi_{i,j} + \phi_{i,j+2}}{2\Delta y} \end{cases} \quad \text{(c)}$$

Now, we substitute the above velocity expressions, Eqs. (b) and (c), into Eq. (a) to solve for φ that provides the gradient correction ($\nabla\varphi$) to make velocity at next time step satisfy incompressibility:

$$\frac{\phi_{i-2,j}-2\phi_{i,j}+\phi_{i+2,j}}{(2\Delta x)^2}+\frac{\phi_{i,j-2}-2\phi_{i,j}+\phi_{i,j+2}}{(2\Delta y)^2}$$
$$=\frac{1}{\Delta t}\left(\frac{-u^p_{i-1,j}+u^p_{i+1,j}}{\Delta x}+\frac{-v^p_{i,j-1}+v^p_{i,j+1}}{\Delta y}\right) \quad \text{(d)}$$

Note that the second-derivative differencing scheme on the left-hand side of Eq.(d) consists of the values ($\phi_{i\pm2,j}$, $\phi_{i,j\pm2}$) skipping every other point about $\phi_{i,j}$. If we consider a pressure boundary condition p0 as shown in Fig. 4.5.1, the odd-number pressure values become independent of the boundary condition. If the pressure gradient $\partial p/\partial x=[-p_{-1}+p1]/2\Delta$ is specified, then the even-number pressure points become decoupled from the boundary conditions. As a result for either case of the boundary condition, the odd and even-number pressure values are not coupled and only enforce smoothness at every other point. This allows for spatial oscillations in the pressure solution to develop between adjacent values. Because of the pattern that this oscillation generates, this numerical instability is referred to as the checkerboard instability

The use of a staggered grid on the other hand, resolves this problem by shifting the locations for discrete velocities by half a mesh width from the pressure locations.

In summary, the use of the staggered grid leads to discretizing the relationship

$$\frac{\partial^2\phi}{\partial x^2}=\frac{\partial}{\partial x}\frac{\partial\phi}{\partial x} \quad \text{(e)}$$

with a compatible finite-difference scheme

$$\frac{\partial^2\phi}{\partial x^2}=\delta_x(\delta_x\phi) \quad \text{(f)}$$

and simultaneously preventing the spatial oscillation in pressure to appear.

4.4 Iteration for the non-advection term

In the two-dimensional Cartesian coordinates, the pressure form is

$$g\frac{\partial H}{\partial x}, g\frac{\partial H}{\partial y}$$

Which, g means gravitational acceleration, H means the sum of water depth and elevation, x,y mean the Cartesian coordinate in x and y direction respectively

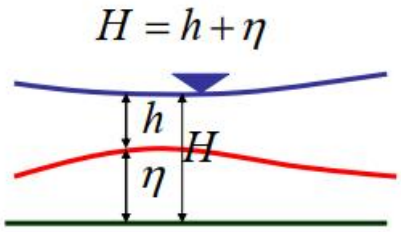


Figure 3.9 Water surface elevation

Let the $h(i,j)$ represents the water depth, and the $\eta(i,j)$ represents the elevation

We could obtain

$$hn(i,j) = h(i,j) + \eta(i,j) \tag{4.5}$$

Use the first order accuracy to discretize the pressure term, we could get

$$\left[g\frac{\partial H}{\partial x}\right]_{i,j} = g\frac{hn(i,j) - hn(i-1,j)}{\Delta x} \tag{4.6}$$

$$\left[g\frac{\partial H}{\partial y}\right]_{i,j} = g\frac{hn(i,j) - hn(i,j-1)}{\Delta y} \tag{4.7}$$

In the two-dimensional Cartesian coordinates, the shear stress form are

n means the Manning coefficient

As shown in Figure 3.7, the shear stress term discretization term are obtained

$$\left[g\frac{n^2\sqrt{\left(u^2+v^2\right)}}{h^{4/3}}\right]_{i,j} = g\frac{n^2\sqrt{u_{i,j}{}^2 + \left[\frac{1}{4}\left(v_{i-1,j} + v_{i,j} + v_{i-1,j+1} + v_{i,j+1}\right)\right]^2}}{\left[\frac{1}{2}\left(\mathrm{h_{i-1,j}} + \mathrm{h_{i,j}}\right)\right]^{4/3}} \tag{4.8}$$

$$\left[g\frac{n^2\sqrt{\left(u^2+v^2\right)}}{h^{4/3}} \right]_{i,j} = g\frac{n^2\sqrt{v_{i,j}{}^2+\left[\frac{1}{4}\left(u_{i,j-1}+u_{i+1,j-1}+u_{i,j}+u_{i+1,j}\right)\right]^2}}{\left[\frac{1}{2}\left(\mathrm{h_{i,j}}+\mathrm{h_{i,j\text{-}1}}\right)\right]^{4/3}} \tag{4.9}$$

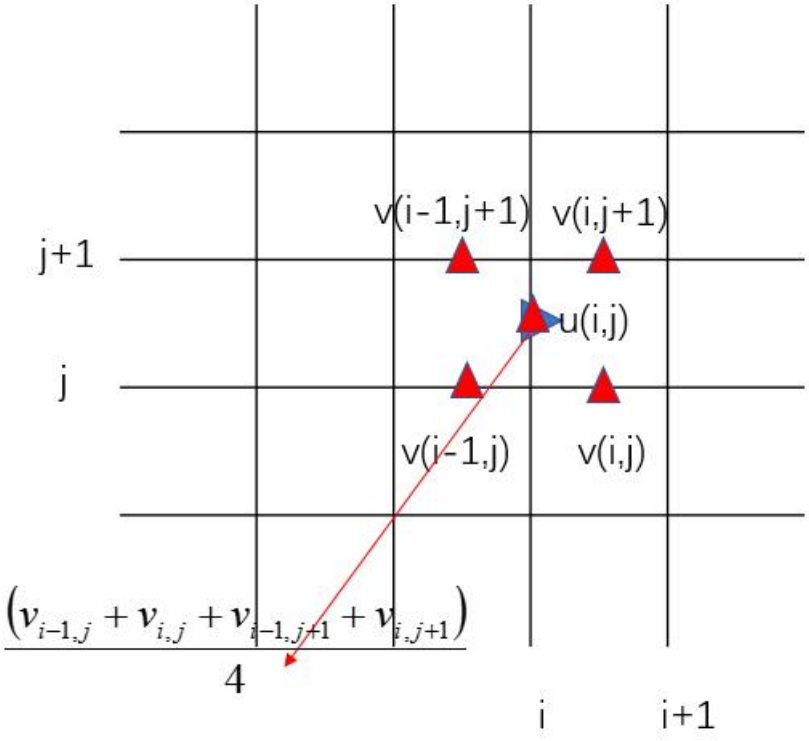


Figure 4.6. illustration of $vup_{i,j}$

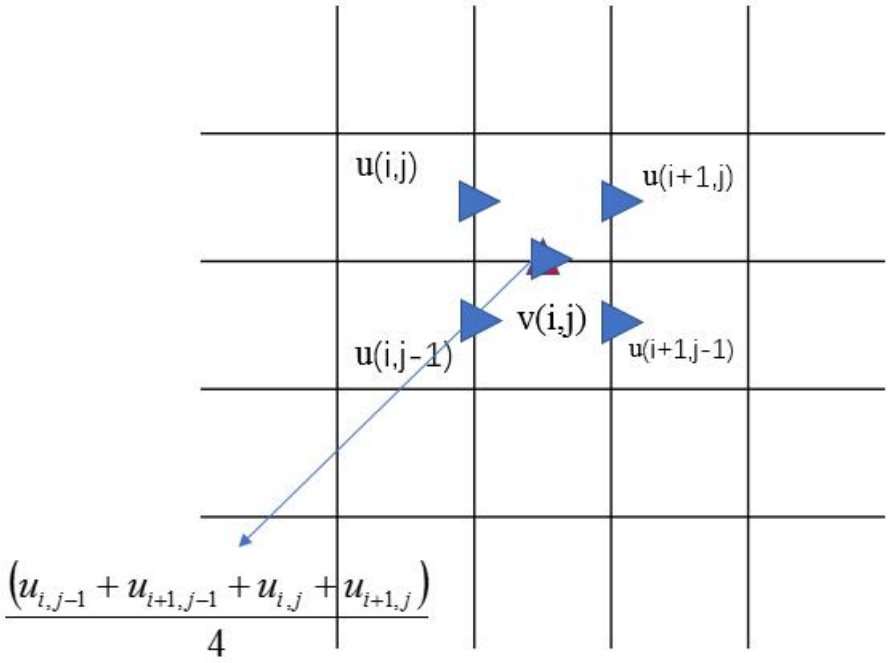


Figure 4.7. illustration of $uvp_{i,j}$

To solve the nonadvection term, the Gauss-seidel iteration method is used
In the Nonadvection phase,

$$\tilde{u} = u^n + \Delta t\left(-g\left(\frac{\partial \tilde{h}}{\partial x}+\frac{\partial \eta}{\partial x}\right)-\frac{gn^2 u\sqrt{u^2+v^2}}{\tilde{h}^{4/3}}\right) \tag{4.10}$$

In this equation, we update u^n to $\tilde{u}$.

$$\tilde{v} = v^n + \Delta t\left(-g\left(\frac{\partial \tilde{h}}{\partial y}+\frac{\partial \eta}{\partial y}\right)-\frac{gn^2 v\sqrt{u^2+v^2}}{\tilde{h}^{4/3}}\right) \tag{4.11}$$

In this equation, we update v^n to $\tilde{v}$.

Then, the fluxed qu, qv are also updated considering the upwind scheme,

$$qu(i,j)=\frac{(\tilde{u}(i,j)+abs(\tilde{u}(i,j)))h(i-1,j)+(\tilde{u}(i,j)-abs(\tilde{u}(i,j)))h(i,j)}{2} \tag{4.12}$$

$$qv(i,j)=\frac{(\tilde{v}(i,j)+abs(\tilde{v}(i,j)))h(i,j-1)+(\tilde{v}(i,j)-abs(\tilde{v}(i,j)))h(i,j)}{2} \tag{4.13}$$

the $\tilde{h}$ in above equations have to satisfy the continuity equation

$$\frac{\partial h}{\partial t}+\frac{\partial(uh)}{\partial x}+\frac{\partial(vh)}{\partial y}=0 \tag{4.14}$$

Discretize it using first order upwind scheme, we could obtain,

$$\tilde{h}_{i,j}=h_{i,j}{}^{n}-\Delta t\left(\frac{-[qu]_{i,j}+[qu]_{i+1,j}}{\Delta x}+\frac{-[qv]_{i,j}+[qv]_{i,j+1}}{\Delta y}\right) \tag{4.15}$$

In this equation, we update the water depth h^n to $\tilde{h}$

If $\tilde{h}_{i,j}-h_{i,j}^{n}<\varepsilon$, the iteration will be stopped.

Why the implicit method is used here?

For incompressible flow, the mass and momentum conservation equations are given belw.

$$\nabla\cdot u=0 \tag{4.16}$$

$$\frac{\partial u}{\partial t}=-\frac{1}{\rho}\nabla P-\frac{\tau_x}{\rho h} \tag{4.17}$$

The continuity equation does not include the time-derivative term and constrains the flow to be divergence-free. The pressure needs to be determined such that the computed flow is consistent with the two conservation equations.

Considering a simple time stepping example using the explicit Euler's method for the Eq.(3.6). We assume density ρ to be constant for simplicity. We can obtain

$$\tilde{u}=u^n+\Delta t\left[-\frac{1}{\rho}\nabla P^n-\frac{\tau^n{}_x}{\rho h^n}\right] \tag{4.18}$$

Even if the known velocity u^n satisfies the continuity equation $\nabla\cdot u^n=0$, the predicted $\tilde{u}$ based on the Eq(3.7) would contain discretization and round-off errors, leading to error in

We may noticed the right hand side nonadvection term including the implicit term $\tilde{h}$

Residual error is nature when the residual error in enforcing incompressibility. If the error accumulate over time, the computation would result in a blow up. In order to prevent such a faliure,

the pressure field should be determined such that $\nabla \cdot \tilde{u} = 0$ is satisfied.

Now considering replacing P^n with P^{n+1} in Eq(3.7),

$$\tilde{u} = u^n + \Delta t\left[-\frac{1}{\rho}\nabla P^{n+1} - \frac{\tau^n{}_x}{\rho h^{n+1}}\right] \tag{4.19}$$

Where P can be regard as Lagrange multiplier that is needed to enforce the incompressibility constraint $\nabla \cdot \tilde{u} = 0$. To determine the true pressure filed P, we must solve the continuity equation

$$\tilde{h}_{i,j} = h_{i,j}{}^n - \Delta t\left(\frac{-[qu]_{i,j} + [qu]_{i+1,j}}{\Delta x} + \frac{-[qv]_{i,j} + [qv]_{i,j+1}}{\Delta y}\right) \tag{4.20}$$

Then using the $\tilde{h}_{i,j}$ to enforce the $\nabla \cdot \tilde{u} = 0$, using the below Equation,

$$\tilde{u} = u^n + \Delta t\left(-g\left(\frac{\partial \tilde{h}}{\partial x} + \frac{\partial \eta}{\partial x}\right) - \frac{gn^2 u\sqrt{u^2+v^2}}{\tilde{h}^{4/3}}\right) \tag{4.21}$$

It's unavoidable to have round-off errors and residuals form the iterative solver, when solving the continuity equation. As long as we solve the continuity equation with sufficient accuracy, the error in mass conservation does not grow as the solution is advance over time.

It is unavoidable to have round-off errors and residuals from the iterative solver the velocity field $\nabla \cdot \tilde{u} = 0$ will inherent these errors. As long as we solve the Non-advection term with sufficient accuracy, the error in mass conservation does not grow as the solution is advanced over time.

It is unavoidable to have round-off errors and residuals from the iterative solver the velocity field $\nabla \cdot \tilde{u} = 0$ will inherent these errors. As long as we solve the Non-advection term with sufficient accuracy, the error in mass conservation does not grow as the solution is advanced over time.

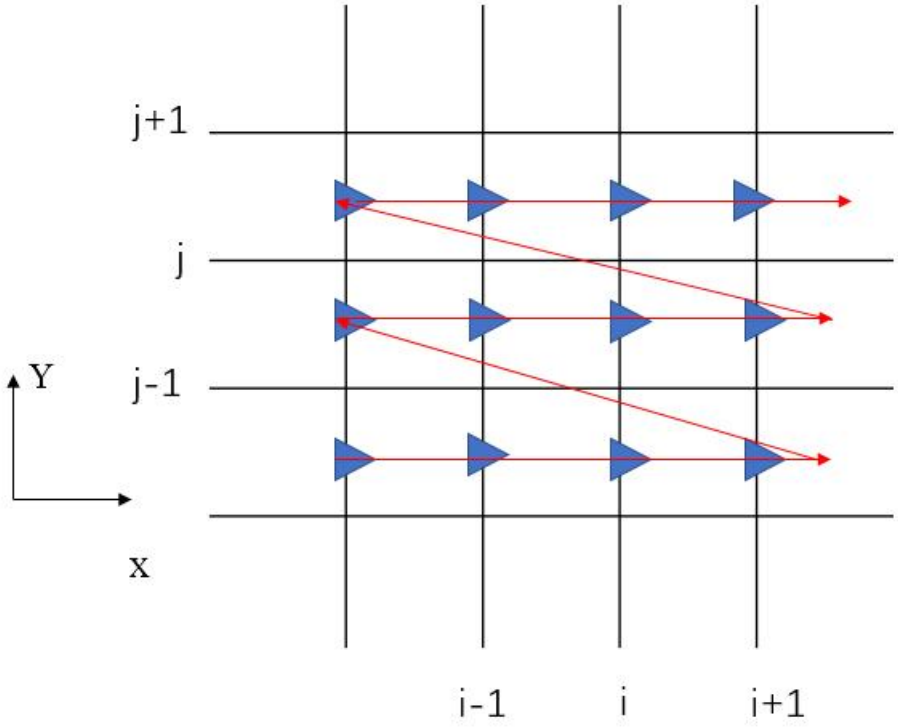


Figure 4.8. Updating the u in space

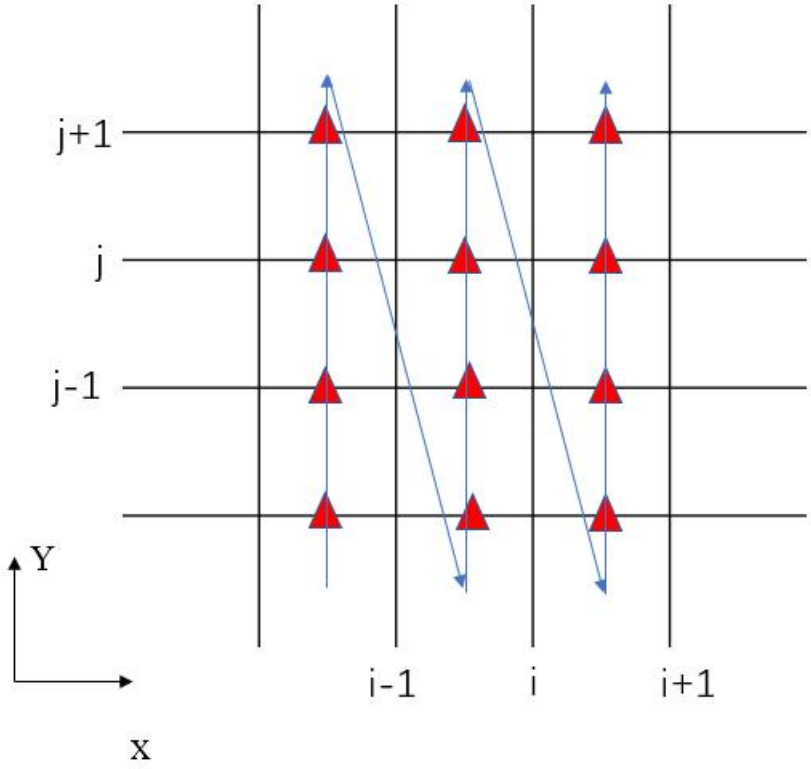


Figure 4.9. Updating the v in space

4.5 Iteration for the advection term

Using the first order upwind scheme for advection term, Firstly, we can define $vup_{i,j} = \frac{\left(v_{i-1,j} + v_{i,j} + v_{i-1,j+1} + v_{i,j+1}\right)}{4}$, then we could obtain the velocity in x direction,

$$u^{n+1} = \widetilde{u} - \frac{\Delta t}{2}\left\{\begin{array}{l}\left[\left(u_{i,j} + abs\left(u_{i,j}\right)\right)\frac{-u_{i-1,j} + u_{i,j}}{\Delta x} + \left(u_{i,j} - abs\left(u_{i,j}\right)\right)\frac{-u_{i,j} + u_{i+1,j}}{\Delta x}\right] \\ + \left(vup_{i,j} + abs\left(vup_{i,j}\right)\right)\frac{-u_{i,j-1} + u_{i,j}}{\Delta y} + \left(vup_{i,j} - abs\left(vup_{i,j}\right)\right)\frac{-u_{i,j} + u_{i,j+1}}{\Delta y}\end{array}\right\} \quad (4.22)$$

For the velocity in y direction, define $uvp_{i,j} = \frac{\left(u_{i,j-1} + u_{i,j} + u_{i+1,j-1} + u_{i+1,j}\right)}{4}$, we could obtain,

$$v^{n+1} = \widetilde{v} - \frac{\Delta t}{2}\left\{\begin{array}{l}\left[\left(v_{i,j} + abs\left(v_{i,j}\right)\right)\frac{-v_{i,j-1} + v_{i,j}}{\Delta y} + \left(v_{i,j} - abs\left(v_{i,j}\right)\right)\frac{-v_{i,j} + v_{i,j+1}}{\Delta y}\right] \\ + \left(uvp_{i,j} + abs\left(uvp_{i,j}\right)\right)\frac{-v_{i-1,j} + v_{i,j}}{\Delta x} + \left(uvp_{i,j} - abs\left(uvp_{i,j}\right)\right)\frac{-v_{i,j} + v_{i+1,j}}{\Delta y}\end{array}\right\} \quad (4.23)$$

First Order Schemes

Merits

As for the numerical method, the classic first-order schemes such as upwind, hybrid, and power-law are unconditionally bounded (solutions do not suffer from over/undershoot),

Demerits

but the first order schemes tend to misrepresent the transport process through addition of numerical diffusion arising from flow-to-grid skewness.

Using CIP method for the advection term

Partial Derivatives in Non-advection term

$$\left[\frac{\partial \widetilde{u}}{\partial x}\right]_{i,j} = \frac{\partial u}{\partial x} + \frac{\left(-\widetilde{u}_{i-1,j} + \widetilde{u}_{i+1,j} + u_{i-1,j} - u_{i+1,j}\right)}{2\Delta x} \quad (4.24)$$

$$\left[\frac{\partial \widetilde{u}}{\partial y}\right]_{i,j} = \frac{\partial u}{\partial y} + \frac{\left(-\widetilde{u}_{i,j-1} + \widetilde{u}_{i,j+1} + u_{i,j-1} - u_{i,j+1}\right)}{2\Delta y} \quad (4.25)$$

$$\left[\frac{\partial \tilde{v}}{\partial x}\right]_{i,j} = \frac{\partial v}{\partial x} + \frac{\left(-\tilde{v}_{i-1,j} + \tilde{v}_{i+1,j} + v_{i-1,j} - v_{i+1,j}\right)}{2\Delta x} \tag{4.26}$$

$$\left[\frac{\partial \tilde{v}}{\partial y}\right]_{i,j} = \frac{\partial v}{\partial y} + \frac{\left(-\tilde{v}_{i,j-1} + \tilde{v}_{i,j+1} + v_{i,j-1} - v_{i,j+1}\right)}{2\Delta y} \tag{4.27}$$

When $u > 0$, $u < 0$ and $v > 0$, $v < 0$

$$i_s = sign(u) \tag{4.28}$$

$$j_s = sign(v) \tag{4.29}$$

$$i_m = i - i_s \tag{4.30}$$

$$j_m = j - j_s \tag{4.31}$$

Advection phase of u,

$$u_{i,j}^{n+1} = \left[\left(a_1 X + c_1 Y + e_1\right)X + g_1 Y + \frac{\partial \tilde{u}_{i,j}}{\partial x}\right]X + \left[\left(b_1 Y + d_1 X + f_1\right)Y + \frac{\partial \tilde{u}_{i,j}}{\partial y}\right]Y + \tilde{u}_{i,j} \tag{4.32}$$

In which,

$$X = -\tilde{\mathrm{u}}_{\mathrm{i,j}}\Delta t, Y = -\tilde{v}_{up_{i,j}}\Delta t \tag{4.33}$$

$$\tilde{v}_{up_{i,j}} = \frac{\left(\tilde{v}_{i-1,j} + \tilde{v}_{i,j} + \tilde{v}_{i-1,j+1} + \tilde{v}_{i,j+1}\right)}{4} \tag{4.34}$$

$$a_1 = \frac{i_s\left[\left(\left(\frac{\partial \tilde{u}}{\partial x}\right)_{i_m,j} + \left(\frac{\partial \tilde{u}}{\partial x}\right)_{i,j}\right)dx - 2\left(\tilde{u}_{i,j} - \tilde{u}_{i_m,j}\right)\right]}{i_s \Delta x^3} \tag{4.35}$$

$$b_1 = \frac{j_s\left[\left(\frac{\partial \tilde{u}}{\partial y}\right)_{i,j_m} + \left(\frac{\partial \tilde{u}}{\partial y}\right)_{i,j}\right]\Delta y - 2\left[\tilde{u}_{i,j} - \tilde{u}_{i,j_m}\right]}{j_s \Delta y^3} \tag{4.36}$$

$$c_1 = \frac{\tilde{u}_{i,j} - \tilde{u}_{i,j_m} - \tilde{u}_{i_m,j} + \tilde{u}_{i_m,j_m} - i_s\left[\left(\frac{\partial \tilde{u}}{\partial x}\right)_{i,j_m} - \left(\frac{\partial \tilde{u}}{\partial x}\right)_{i,j}\right]\Delta x}{j_s \Delta x^2 \Delta y} \tag{4.37}$$

$$d_1 = \frac{\tilde{u}_{i,j} - \tilde{u}_{i,j_m} - \tilde{u}_{i_m,j} + \tilde{u}_{i_m,j_m} - i_s\left[\left(\frac{\partial \tilde{u}}{\partial x}\right)_{i_m,j} - \left(\frac{\partial \tilde{u}}{\partial x}\right)_{i,j}\right]\Delta y}{j_s \Delta x \Delta y^2} \tag{4.38}$$

$$e_1 = \frac{3\left[\tilde{u}_{i,j_m} - \tilde{u}_{i,j}\right] - i_s\left[\frac{\partial \tilde{u}}{\partial x}_{i_m,j} + 2\frac{\partial \tilde{u}}{\partial x}_{i,j}\right]\Delta x}{\Delta x^2} \tag{4.39}$$

$$f_1 = \frac{3\left[\tilde{u}_{i,j_m} - \tilde{u}_{i,j}\right] - j_s\left[\frac{\partial \tilde{u}}{\partial x}_{i,j_m} + 2\frac{\partial \tilde{u}}{\partial x}_{i,j}\right]\Delta y}{\Delta y^2} \tag{4.40}$$

$$g_1 = \frac{-\frac{\partial \tilde{u}}{\partial y}_{i_m,j} + \frac{\partial \tilde{u}}{\partial y}_{i,j} - c_1 \Delta x^2}{i_s \Delta x} \tag{4.41}$$

Advection phase of v,

$$v_{i,j}^{n+1} = \left[\left(a_1 X + c_1 Y + e_1\right)X + g_1 Y + \frac{\partial \tilde{v}_{i,j}}{\partial x}\right]X + \left[\left(b_1 Y + d_1 X + f_1\right)Y + \frac{\partial \tilde{v}_{i,j}}{\partial y}\right]Y + \tilde{v}_{i,j} \tag{4.42}$$

In which,

$$X = -\tilde{\mathrm{u}}_{\mathrm{vp_{i,j}}}\Delta t, Y = -\tilde{v}_{i,j}\Delta t \tag{4.43}$$

$$\tilde{u}_{vp_{i,j}} = \frac{\left(\tilde{u}_{i,j-1} + \tilde{u}_{i+1,j-1} + \tilde{u}_{i,j} + \tilde{u}_{i+1,j}\right)}{4} \tag{4.44}$$

$$a_1 = \frac{i_s\left[\left(\left(\frac{\partial \tilde{v}}{\partial x}\right)_{i_m,j} + \left(\frac{\partial \tilde{v}}{\partial x}\right)_{i,j}\right)dx - 2\left(\tilde{v}_{i,j} - \tilde{v}_{i_m,j}\right)\right]}{i_s \Delta x^3} \tag{4.45}$$

$$b_1 = \frac{j_s\left[\left(\frac{\partial \tilde{v}}{\partial y}\right)_{i,j_m} + \left(\frac{\partial \tilde{v}}{\partial y}\right)_{i,j}\right]\Delta y - 2\left[\tilde{v}_{i,j} - \tilde{v}_{i,j_m}\right]}{j_s \Delta y^3} \tag{4.46}$$

$$c_1 = \partial\frac{v_{i,j} - v_{i,j_m} - v_{i_m,j} + v_{i_m,j_m} - i_s\left[\left(\frac{\partial v}{\partial x}\right)_{i,j_m} - \left(\frac{\partial v}{\partial x}\right)_{i,j}\right]\Delta x}{j_s \Delta x^2 \Delta y} \tag{4.47}$$

$$d_1 = \frac{v_{i,j} - v_{i,j_m} - v_{i_m,j} + v_{i_m,j_m} - i_s\left[\left(\frac{\partial v}{\partial x}\right)_{i_m,j} - \left(\frac{\partial v}{\partial x}\right)_{i,j}\right]\Delta y}{j_s \Delta x \Delta y^2} \tag{4.48}$$

$$e_1 = \frac{3\left[v_{i,j_m} - v_{i,j}\right] - i_s\left[\frac{\partial v}{\partial x}_{i_m,j} + 2\frac{\partial v}{\partial x}_{i,j}\right]\Delta x}{\Delta x^2} \tag{4.49}$$

$$f_1 = \frac{3\left[v_{i,j_m} - v_{i,j}\right] - j_s\left[\frac{\partial v}{\partial x}_{i\ ,j_m} + 2\frac{\partial v}{\partial x}_{i,j}\right]\Delta y}{\Delta y^2} \tag{4.50}$$

$$g_1 = \frac{-\frac{\partial v}{\partial y}_{i_m,j} + \frac{\partial v}{\partial y}_{i,j} - c_1 \Delta x^2}{i_s \Delta x} \tag{4.51}$$

4.6 Geometries interpolation

In this elevation data, the different spatial density of building is also modeled based on the aerial photograph (i.e., Fig.3.6a). When refinement or de-refinement is taken place in the computation, this elevation data is interpolated into every node of the computational grid by weighted averaging as,

$$\eta = \frac{\Sigma z / r}{\Sigma 1 / r} \tag{4.52}$$

where z represents the elevation data, η is the elevation at the computational node, and r is the distance between the elevation data and grid node. The window size for searching the elevation data for this averaging, denoted in the Figure 4.10, is set to be local grid size. If there is no data within this window size, the elevation of nearest data is used for the elevation of the computational node.

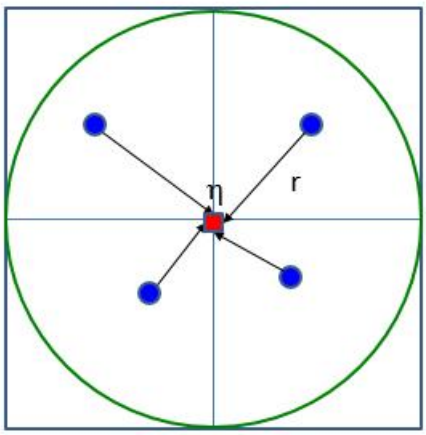


Figure 4.10 Illustration of geometries interpolation

4.7 Simulation method applied to bottom friction

This urban flood modeling asts the bottom friction by using the Manning's roughness coefficient. In Eq(3.2), Eq(3.3), Eq(3.7) and Eq(3.8), the shear stress τ_x and τ_y are expressed by the bed friction coefficient C_f . The relationship between bed friction coefficient C_f and the Manning's roughness coefficient n is as follows:

$$C_f = \frac{gn^2}{h^{1/3}} \tag{4.53}$$

n can be given to each grid cell and n are given to the uniform value in the urban flood modeling.

4.8 Variables on PARAMESH

Table 4.2 variblaes on Paramesh

Variables Name	Description
unk(1,i,j,k,lb)	water depth
unk(2,i,j,k,lb)	bed elevation
unk(5,i,j,k,lb)	old value of water depth
facevarx(1,i,j,k,lb)	flow velocity in x direction
facevarx(2,i,j,k,lb)	water discharge in x direction
facevarx(3,i,j,k,lb)	du/dx
facevarx(4,i,j,k,lb)	du/dy
facevarx(5,i,j,k,lb)	old value of flow velocity in x direction
facevary(1,i,j,k,lb)	flow velocity in y direction
facevary(2,i,j,k,lb)	water discharge in y direction
facevary(3,i,j,k,lb)	dv/dx
facevary(4,i,j,k,lb)	dv/dy
facevary(5,i,j,k,lb)	old value of flow velocity in y direction

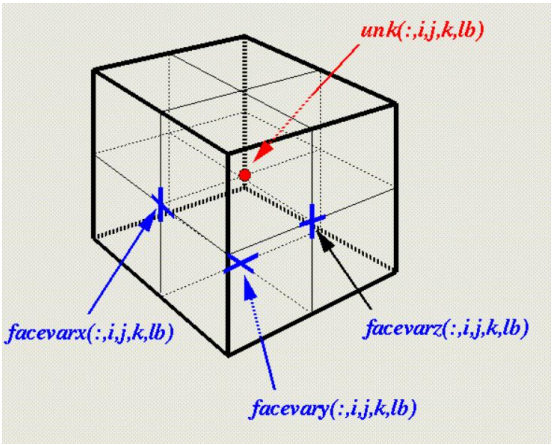


Figure 4.11 A single cell within a PARAMESH block. Data can be located in any combination at cell centers, cell edges, cell faces, or cell corners. (from MacNeice et al.2000)

Table 4.3 Description of the physical data module setting in PARAMESH_PREPROCESSOR.FH

Data name	Value	Description
maxblocks	30000	The maximum blocks you can treat in the computation. When the number of blocks in the computation exceeds this number, the computation stops.
ndim	2	The dimension of the computation. 1 -> 1 dimensional, 2 -> 2 dimensional, 3 -> 3 dimensional computation.
l2p5d	0	A 2.5D MHD code with mass density, x and y momentum and energy density all specified at grid cell center, and with the x-component of magnetic field on the x-face, y-component of magnetic field on the y-face and z-component of magnetic field at cell center :In this case the parameter L_2P5DIM must be set to 1.
nxb	10	number of grid in the blocks in x direction
nyb	10	number of grid in the blocks in y direction
nzb	0	number of grid in the blocks in z direction
nvar	5	number of variables defined in the cell.
nfacevar	5	number of variables defined in the face of cell.
nvaredge	0	number of variables defined in the edge of cell.
nvarcorn	0	number of variables defined in the corn of cell.
nvar_work	2	Number of work variables
nguard	1	Number of guard cell
nguard_work	1	the number of guard cells at the block boundary is set by a distinct parameter

Data name	Value	Description
nfluxvar	1	Number of flux variables
nedgevar	0	Numbder of edge variables
iface_off	0	User set parameter which defines how the indeces associated with cell centered and non-cell-centered data are related. For example, the default value (0) means that FACEVARX(:,I,:,:,:) and FACEVARX(:,I+1,:,:,:) bound UNK(:,I,:,:,:). If IFACE_OFF=-1 then FACEVARX(:,I-1,:,:,:) and FACEVARX(:,I,:,:,:) bound UNK(:,I,:,:,:).
mflags	1	Defines the number of flags which can be associated with each block
nfield_divf	0	PARAMESH offers three possible approaches to applying divergenceless prolongation to solution fields. They are all applicable to fields whose components are located at cell-face centers, and which are stored in the FACEVAR arrays. The number of divergence-free fields which the user requires need to be specified in the PARAMESH_PREPROCESSOR.FH file, by defining a value for NFIELD_DIVF. Balsara's polynomial fit (see Balsara reference) A block-local divergence cleanup (suggested by Phil Colella) Linear or zeroth order(injection) interpolation, followed by adjustment of field components adjacent to refinement jumps.
nboundaries	4	Number of boundaries
diagonals	ture	If your algorithm requires diagonal elements then define the preprocessor variable DIAGONALS to ensure that this extra guard cell data is provided.
amr_error_checking	true	
no_permanent_guardcells	false	If NO_PERMANENT_GUARDCELLS is defined, then you must not modify these arrays between the call to MPI_AMR_COMM_SETUP which precedes guardcell filling and the subsequent calls to AMR_1BLK_GUARDCELL
advance_all_levels	false	If you wish to advance the solution at all refinement levels then define ADVANCE_ALL_LEVELS. Otherwise the solution will be advanced on leaf nodes only. (Note, if VAR_DT is defined above then you must define ADVANCE_ALL_LEVELS.)

Data name	Value	Description
force_consistency	true	If roundoff error can introduce inconsistencies in these data values, which, in principle should remain identical, and these inconsistencies can grow because of the nature of your algorithm, you may wish to force them to be made consistent. If so, then define the pre-processor variable FORCE_CONSISTENCY_AT_SRL_INTERFACES. Currently this only works for face-centered data.
consv_fluxes	false	the fluxes at the block boundary facing a finer block are replaced by the sum of the fluxes in the corresponding cells on the finer block.
consv_flux_densities	true	the flux densities at the block boundary facing a finer block are replaced by the sum of the average of the flux densities in the corresponding cells on the finer block.
edge_value	true	To ensure consistency of circulation integral it may be necessary for you to store information temporarily about edge variables at block boundaries. You can choose to store this information in the form of the edge values. If you choose to use
edge_value_integ	false	To ensure consistency of circulation integral it may be necessary for you to store information temporarily about edge variables at block boundaries. You can choose to store this information in the form of the edge value times the edge length If you are using curvilinear coords you must assume EDGE_VALUE_INTEG, and it will automatically be selected for you here. If you do not intend calling the routine AMR_EDGE_AVERAGE, it does not matter which choice you make
var_dt	false	To use different timesteps on different grid blocks define VAR_DT
pred_corr	false	You can define the preprocessor variable PRED_CORR, which causes a second copy of the basic data-structure, T_UNK, T_FACEVARX, etc, to be set up. This can then be used in place of doubling the number of variables.
empty_cells	false	The use of empty blocks has to be signaled to the preprocessor by defining the variable EMPTY_CELLS
conserve	true	CONSERVE is available to force the standard linear interpolation routine to conserve the interpolated variable

Data name	Value	Description
divergence_free	false	Apply either linear or zeroth order(injection) interpolation. This will maintain divergence free fields inside the grid block. The only problem with this is that it may not match existing field values located on the boundaries of pre-existing neighbors at the refinement level of the newly created child block. Since we require that the newly prolonged child block have the same values on the common block boundary as already exist on the pre-existing neighbor block, we must overwrite the values set there by the interpolation process. This can introduce non-zero divergence in the cells adjacent to this block boundary. To correct this, we need to call a routine named AMR_PROLONG_FC_DIVBCONSIST which adjusts the field values on the internal faces of these cells to re-establish zero divergence. This will be called during AMR_PROLONG, if you have defined the preprocessor variable DIVERGENCE_FREE in PARAMESH_PREPROCESSOR.FH. The variables which compose the divergence fields are specified in the same way, prior to the call to AMR_PROLONG.
curvilinear	false	coordinate system
curvilinear_conserve	false	coordinate system
cartesian	true	coordinate system
cylindrical	false	coordinate system
spherical	flase	coordinate system
polar	false	coordinate system
lsingular_line	false	
timing_mpi	true	
timing_mpix	false	
output_dir	false	

4.9 Boundary Conditions

For the solid wall conditions

Along the solid wall, the no-slip boundary condition of v =0 or the slip boundary conditions of $\partial v / \partial y = 0$

The parameters in the Figure 4.12 are given in the table 4.4

Parameters Name	Description
qin	water discharge
Slope	slope used for uniform flow calculation at upstream boundary condition
snm	Manning's roughness coefficient
Width	channel width for upstream inflow
Riv_x	location of upstream inflow

The Dirichlet boundary condition are implemented into the boundaries in this urban flood modeling denoted in Figure 4.12.

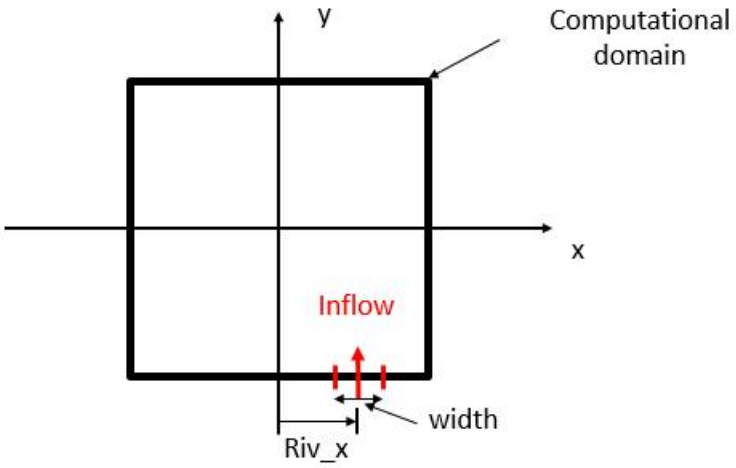


Figure 4.12 Illustration of computational domain

For the inflow boundary, we give a constant slope for the inflow, $i = 0.001$. Then, the water depth on the inflow boundary can be obtained by the given parameters in table 4.4

$$h_0 = \left(\frac{n(qin)}{(width)(slope)^{1/2}} \right)^{3/5} \tag{4.54}$$

$$Q_{disc} = qin / width \tag{4.55}$$

$$v_{\text{inflow}} = qin / h_0 \tag{4.56}$$

The boundary conditions are given for other external boundaries as follows:

$$Q_{disc} = 0 \tag{4.57}$$

$$u = 0 \text{ or } v = 0 \tag{3.48}$$

Wet/dry boundary tracking

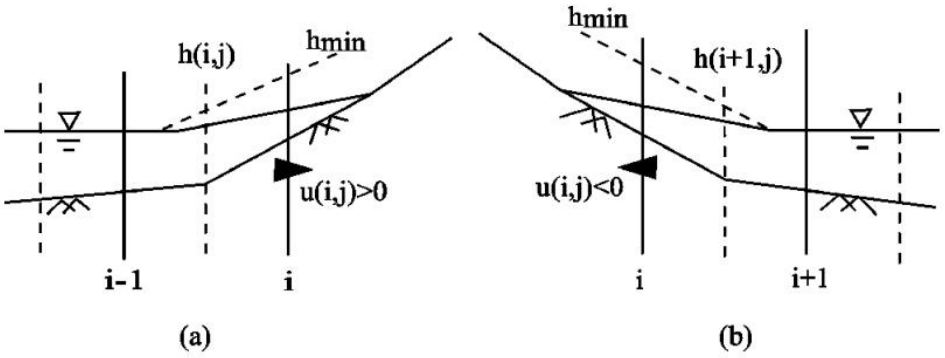


Figure 4.13 Scheme of boundary near the emerged bars and banks in a staggered grid system:(a) $dh/dx < 0, u > 0$ or $dh/dy < 0, v > 0$ in the x or y direction and (b) $dh/dx > 0, u < 0$ or $dh/dy > 0, v < 0$ in the x or y direction (Jang and Shimizu et al.,2005)

Dealing with a partially drying or wetting boundary is important to simulate the inundation of flood plains and wetlands, shallow flows over and near islands or emerged bars in rivers, and wave run-up in a coastal line(Bradford and Sanders 2002; Zhao et al. 1994). Referring to Fig. 4.13, the boundary in the shallow regions around emerged bars and near side banks with a moving boundary is dealt with as follows.

In the y direction, near the emerged bars:

$$\text{If } dh/dy < 0, v > 0 \text{, then } v_{i,j} = 0$$

$$\text{If } dh/dy < 0, v > 0 \text{, then } v_{i,j} = 0$$

In the x direction, near emerged bars and side banks:

$$\text{If } dh/dx < 0, u > 0 \text{ then } u_{i,j} = 0$$

$$\text{If } dh/dx > 0, u < 0 \text{ then } u_{i,j} = 0$$

Since *hmin* has a very small value, but there is flow velocity in the y or x directions in these situations, the momentum equations are not solved and the flow velocity is set to zero, which probably leads to some numerical error, and must be studied further in the future. In these equations, *hmin* is set to 0.02 mm to minimize the error. In the fully wetted regions, the continuity and momentum equations are solved using the CIP method in the advection terms.

4.10 Time stepping

For each direction, we can calculate the Courant Number as below

$$C \equiv \frac{\mathrm{u}\Delta t}{\Delta x} \leq 1$$

The Courant Number is set 0.2 in this study. As time iteration, we could find the maxium velocity in x, and y direction. Then we could obtain the smallest dt

$$dt_x = \frac{C\Delta \mathrm{x}}{u_{\max}}, dt_y = \frac{C\Delta \mathrm{y}}{v_{\max}}$$

This global minimum time step is to used everywhere, regardless of local refinement level.

4.11 Flow Chart of Simulation

To solve the shallow water equations, the momentum equation is split into advection term and nonadvection term. The implicit method is used for the nonadvection term and the CIP method is used for the advetion term. The procedures of the main program are given in Figure 4.13.

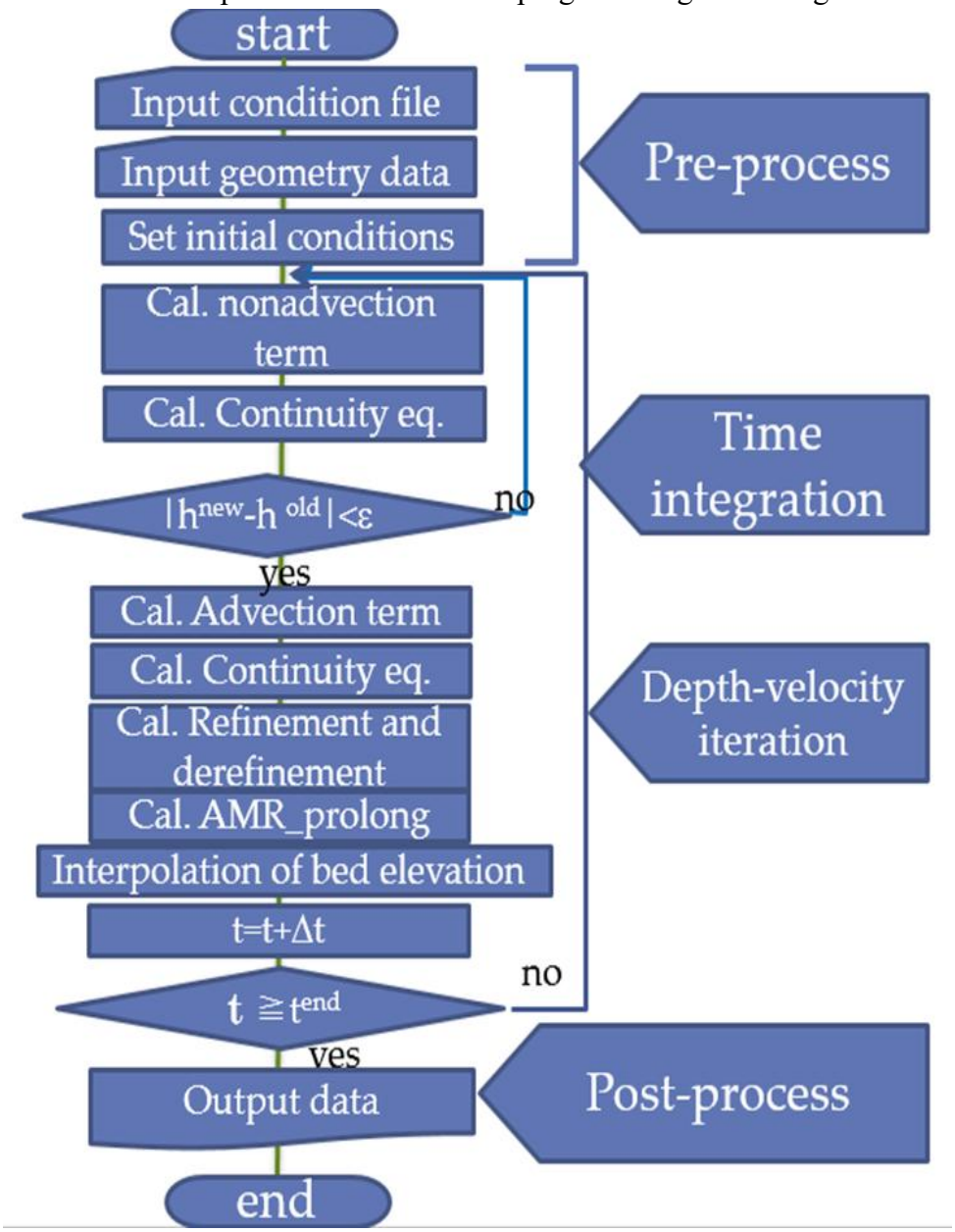


Figure 4.13 the flow chart of the simulation

4.12 Test case1: Uniform discharge

In the test case1 , the uniform discharge is tested. The details of the calculation conditions are given by Table 4.4.

Table 4.5. The calculation conditions of the computational model.

Computational domain		2800 m x 2800 m
Manning's roughness coefs.		0.03
Water discharge		771 m^3/s
AMR	minimum refinement level (level 1)	280 m x 280 m
	Maximum refinement level (level 7)	4.4 m x 4.4 m
Fine grid model		4.4 m x 4.4 m
Coarse grid model		17.5m x 17.5m
Calculation end time		2000 seconds

In the test case2, the non-uniform discharge are given in the Figure 4.20, other calculation conditions are same with test case1.

4.12.1 Results and discussion

Table 4.6. The results of the AMR with different refinement criteria and the fixed fine grid.

Refinement Criteria		Threshold value upper value (refine)	Threshold value lower value (derefine)	Refinement level Min.	Refinement level Max.	Number of meshes	Computational Time
AMR1	$\frac{\Delta v}{v}$	0.35	0.25	4	7	314900	3h37min
AMR2	$\frac{\Delta h}{h}$	0.24	0.09	1	6	74100	39min
AMR3	$\frac{\Delta h}{h}$	0.35	0.25	1	7	384900	5h42min
AMR4	$\frac{\Delta h}{h}$	0.24	0.09	1	7	380500	3h39min
AMR5	$\frac{\Delta h}{\Delta x}, \frac{\Delta h}{\Delta y}$	0.005	0.0005	1	7	306900	3h25min
AMR6	Δh	0.05	0.002	3	7	306900	3h36min
AMR7	Δv	0.05	0.002	1	7	314900	3h37min
-	Fine grid	-	-	-	-	546100	10h45min

downstream area, where the building density is high. This is obviously because that the coarse model cannot represent complicated bed geometry, which is mostly caused by a lot of buildings.

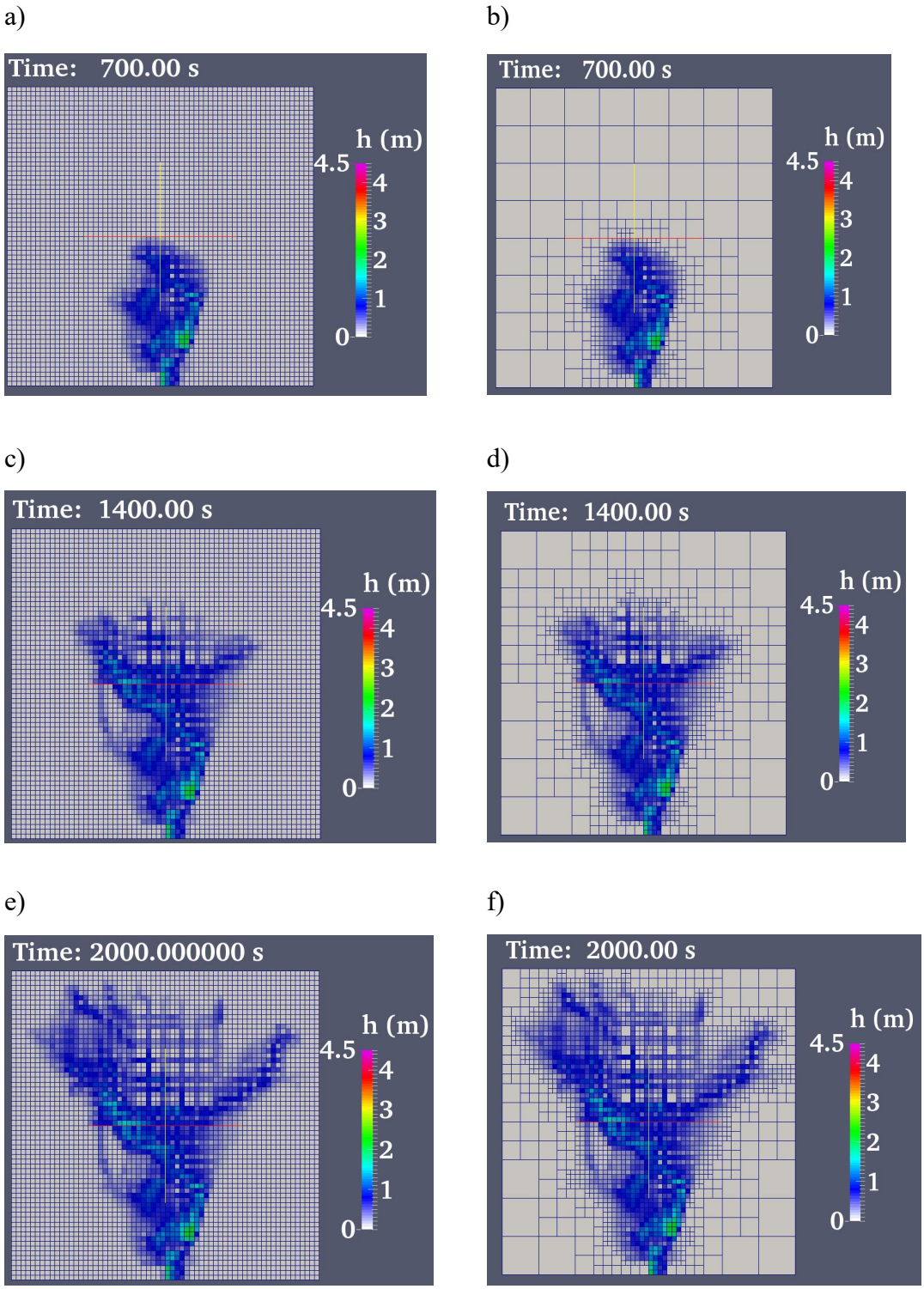


Figure 4.14 The temporal change of water depth. a), c) and e) fine-grid case and the b), d) and f) AMR case at 700, 1400 and 2000 seconds, respectively.

a) b)

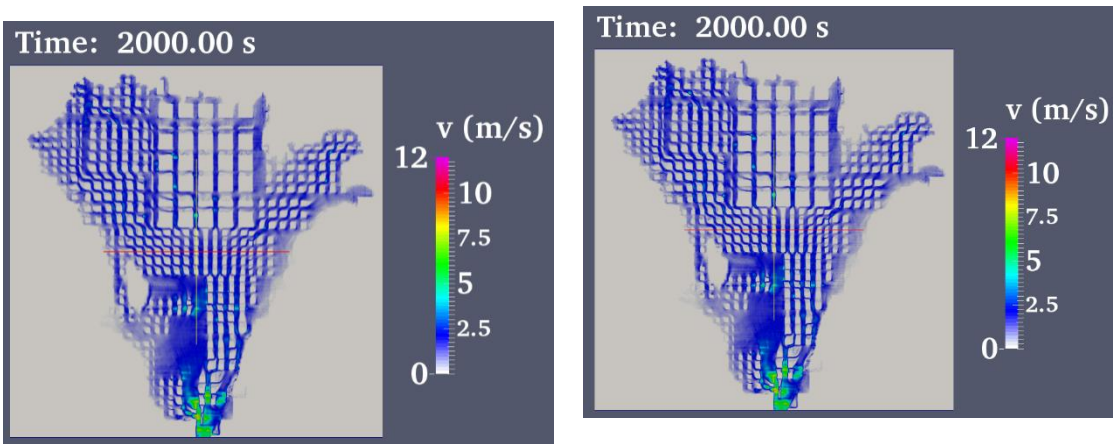


Figure 4.15.The velocity distribution of a) fine grid case, b) AMR case at 2000 seconds.

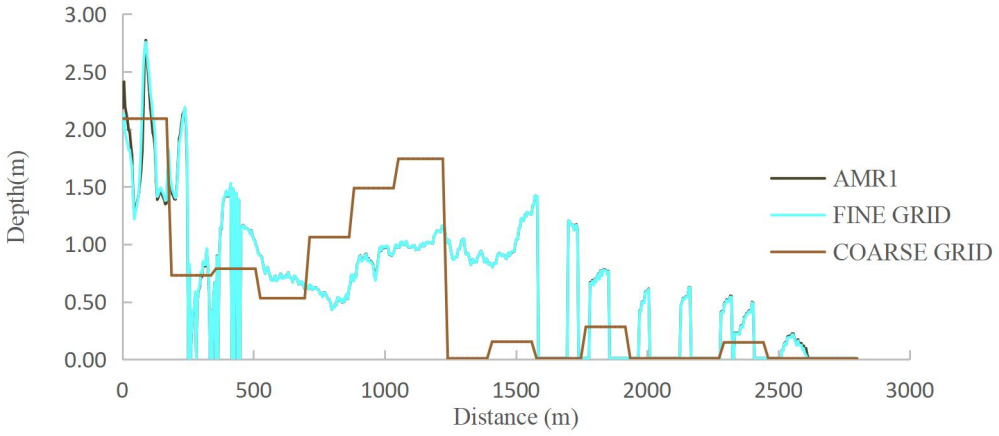


Figure 4.16. The water depth along the black line defined in Fig. 4.3b on the AMR case, fine grid and the coarse gird model, respectively.

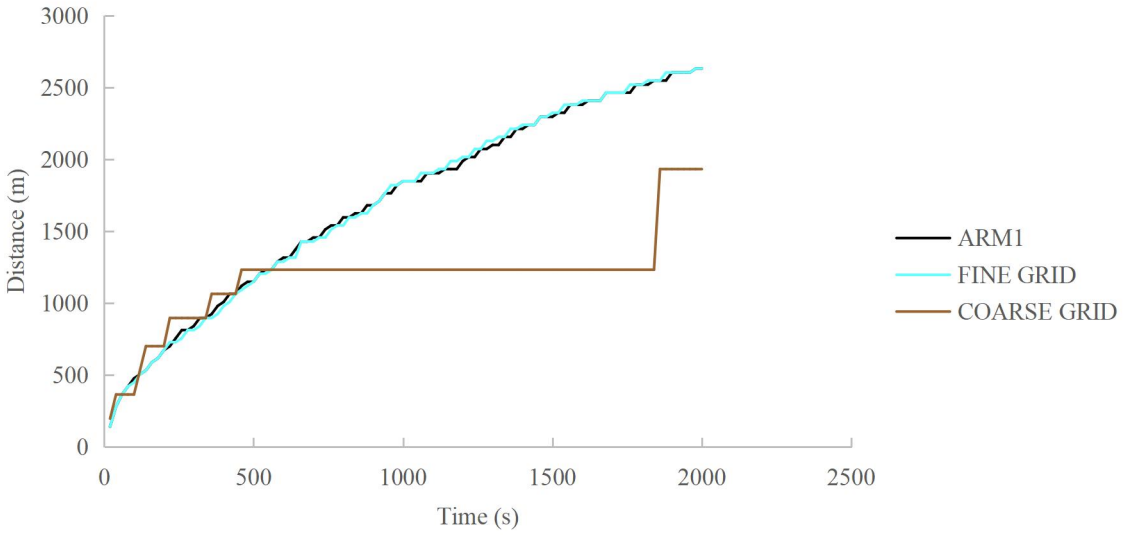


Figure 4.17. Temporal change of the location of the front inundated area from the upstream inflow along the black line defined in Fig. 4.3b.

Firstly, we compare the water depth (Fig.4.14) and flow velocity (Fig.4.15) of the flood modeling between the AMR case and the fixed fine grid case. For the AMR calculation, the relative velocity difference (i.e., $\Delta v/v$) is used for the refinement criteria (see AMR1 on Table 4.6).
Figures 4.14b, d, and f show that the computational grid is spatially refined depending on the flood propagation but the grid remains coarse where there is no flood flow. The comparison of the water depth also shows that the AMR case reasonably captures the temporal change of water depth (i.e., flood propagation) simulated by the fine-grid model (Figs 4.14a, c, and e). In addition, as shown in Figure 4.15, the flow velocity simulated in the fine-grid model and AMR are almost identical.

Figure 4.16 visualizes the one-dimensional view of water depths of the grid point along a downstream direction as indicated in the black line in Fig. 1b at 2000 seconds. In addition, Figure 4.17 shows the temporal change of the location of flooding (i.e., the location of the head of the flood flow along the black line shown in Fig. 4.3b), describing how fast the flood propagates in downstream direction. Both figures indicate that the AMR correctly simulate the result of fine grid case, capturing inundation depth and flood propagation speed. The coarse grid model partly captures these flow feature in the upstream area, where the spatial density of buildings is not very high, but completely fails to reproduce these features in

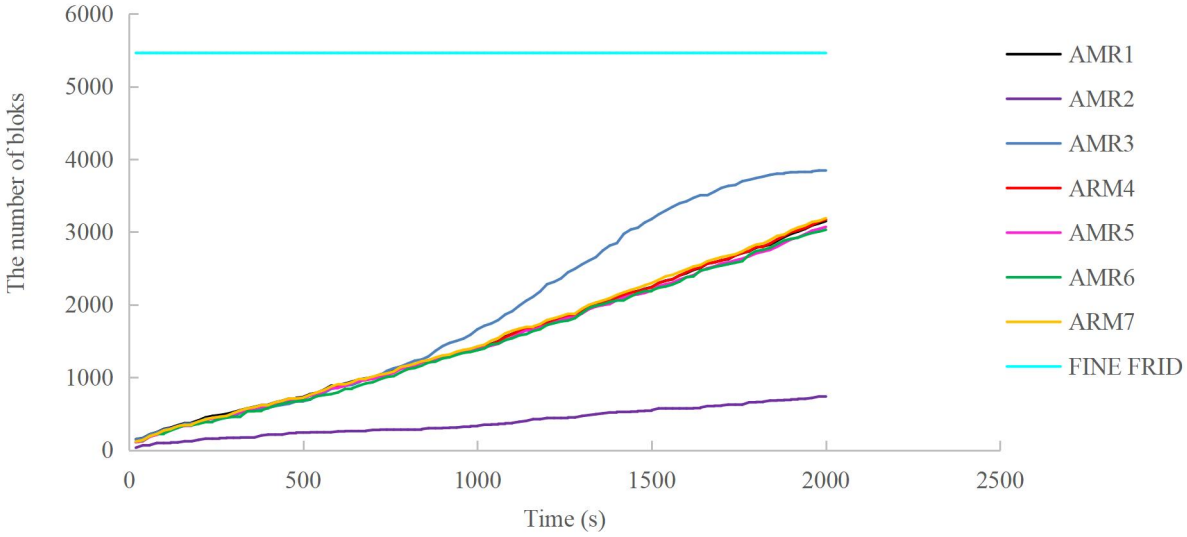


Figure 4.18. the number of blocks on various refinement criteria change along the time

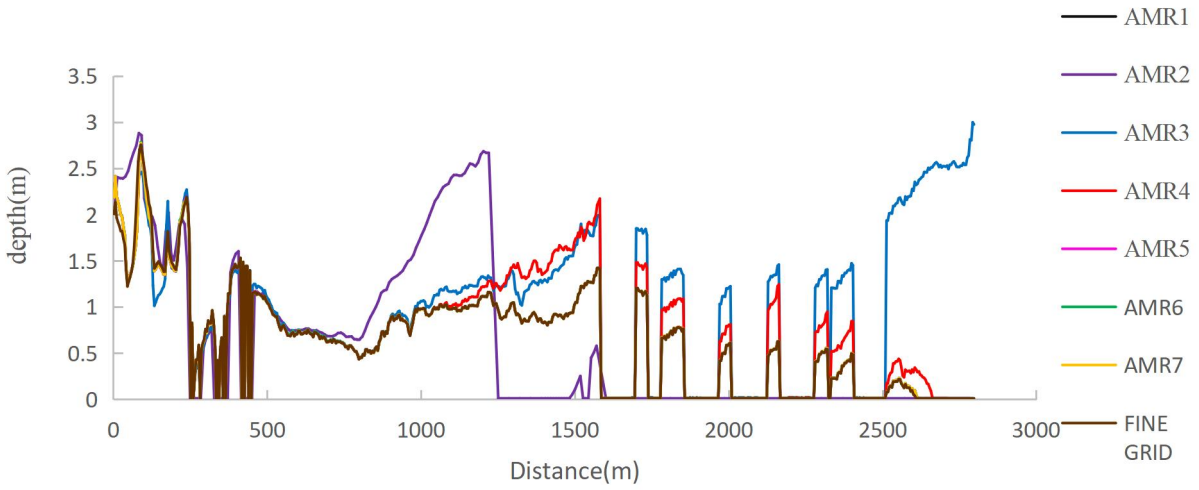


Figure 4.19. The water depths along the black line (showed in Fig. 4.3b) at 2000s on various refinement criteria and the fine grid case.

This result suggests that the proposed flood model with AMR is able to capture result obtained by the high-resolution, but uniform fine grid without significant loss of the accuracy but greatly reducing the computational grid number.

This result suggests that the proposed flood model with AMR is able to capture result obtained by the high-resolution, but uniform fine grid without significant loss of the accuracy but greatly reducing the computational grid number.

From the Fig 8. we found the water depth of AMR1,5,6,7 cases are almost identical with fine grid case.

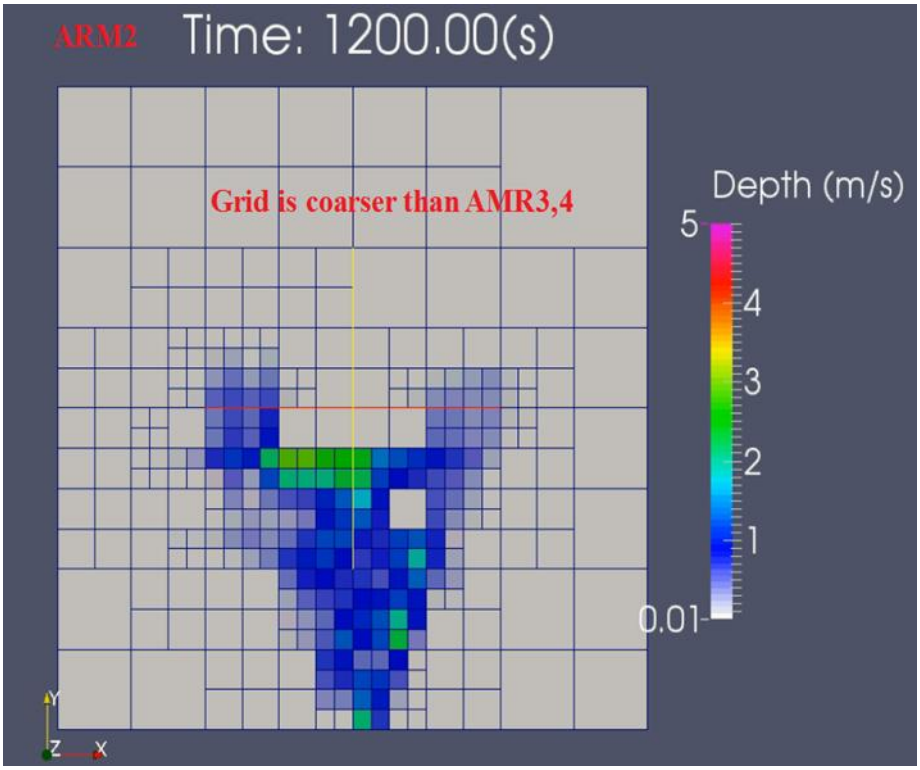


Figure 4.20. AMR2 model simulate the flood at 1200s

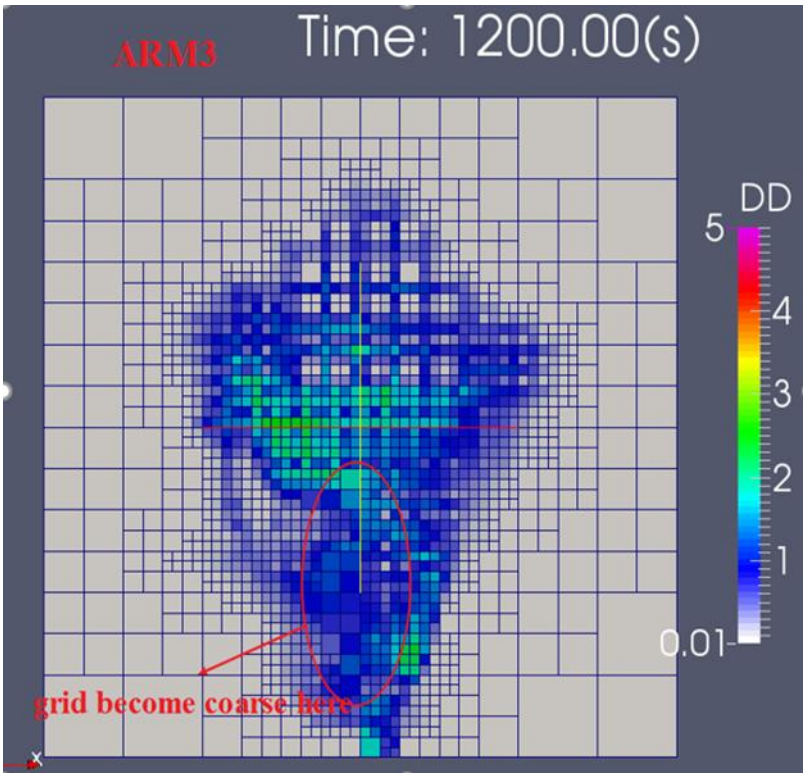


Figure 4.21. AMR3 model simulate the flood at 1200s

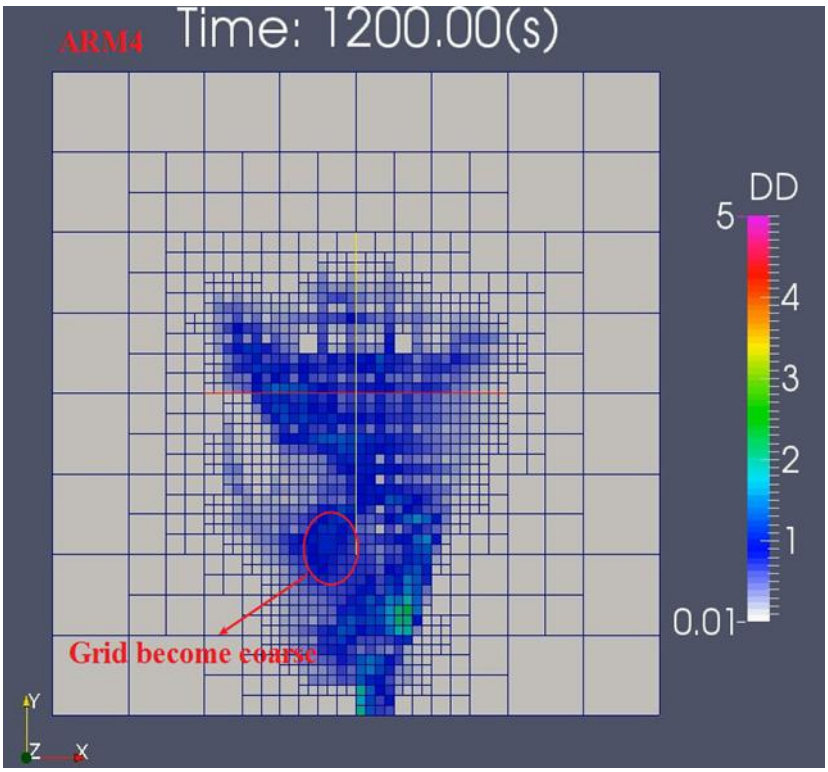


Figure4.22. AMR4 model simulate the flood at 1200s

Although AMR2, 3 and 4 is not so accurate compared with Fine grid from Figure 8, the results could become more accurate by revising the refinement level and threshold value for refinement and derefinement.

4.13 Test case2: Non-uniform discharge

In the nonuniform discharge case, the water surface slope still can capture the flow characteristics well and saved 43% time.

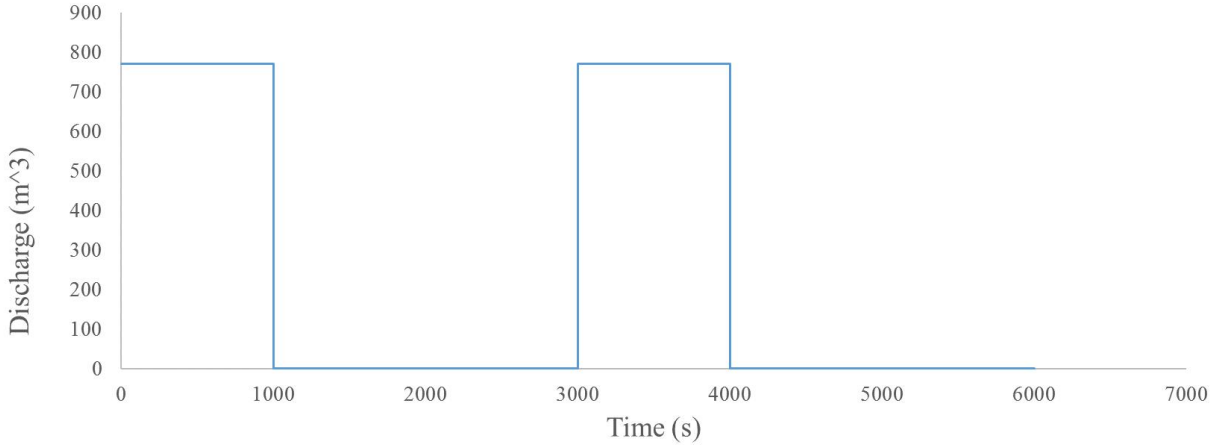


Figure 4.23.The Discharge change along the time

4.13.1 Results and discussion

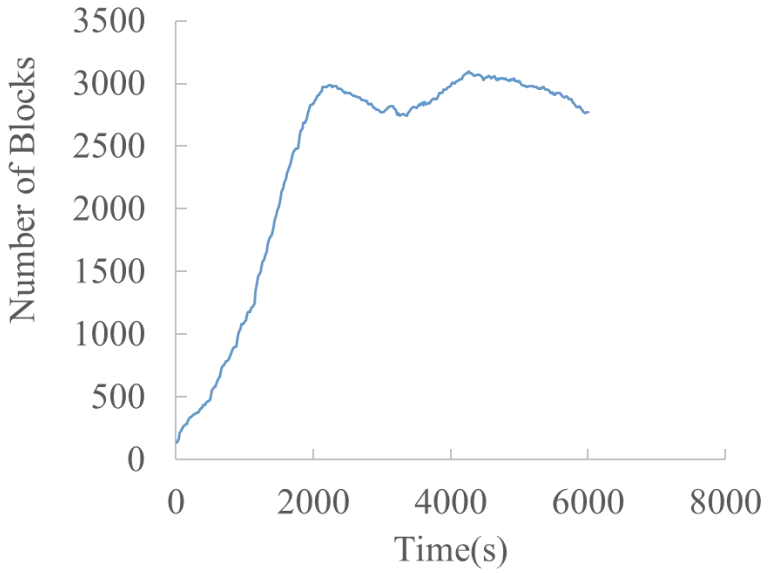


Figure 4.24. The block numbers of AMR case at different time case on non-uniform discharge case.

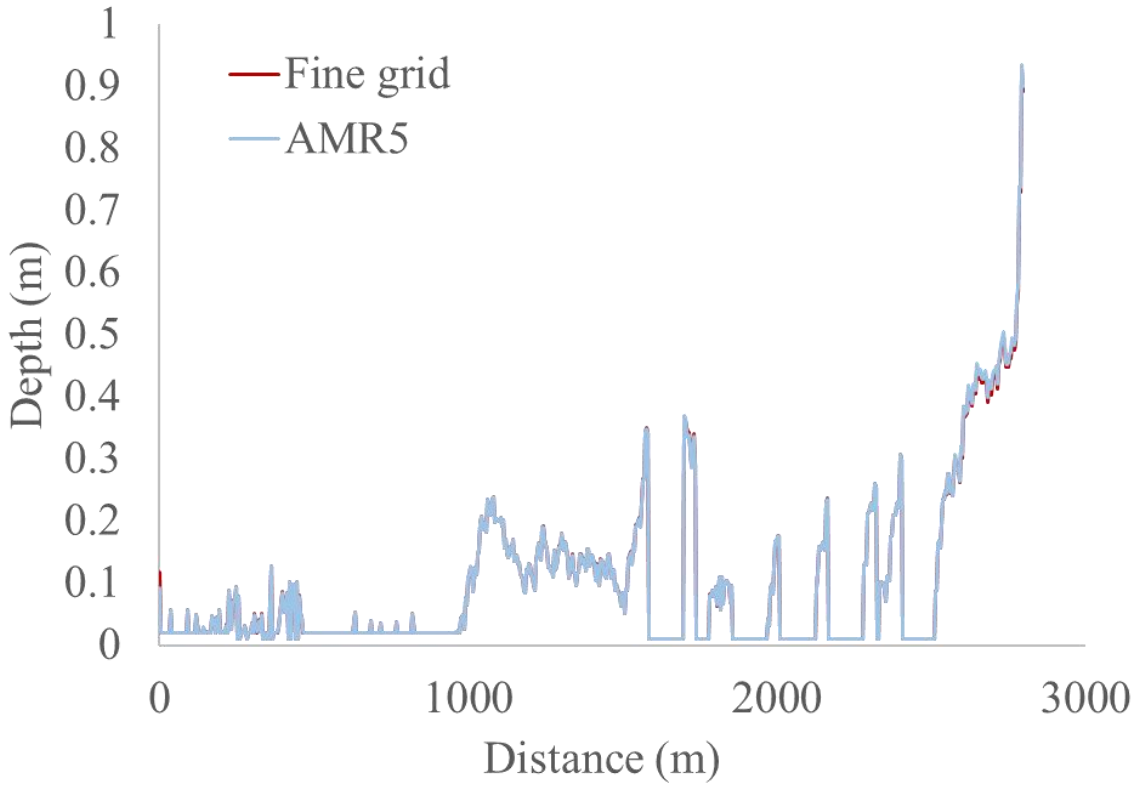
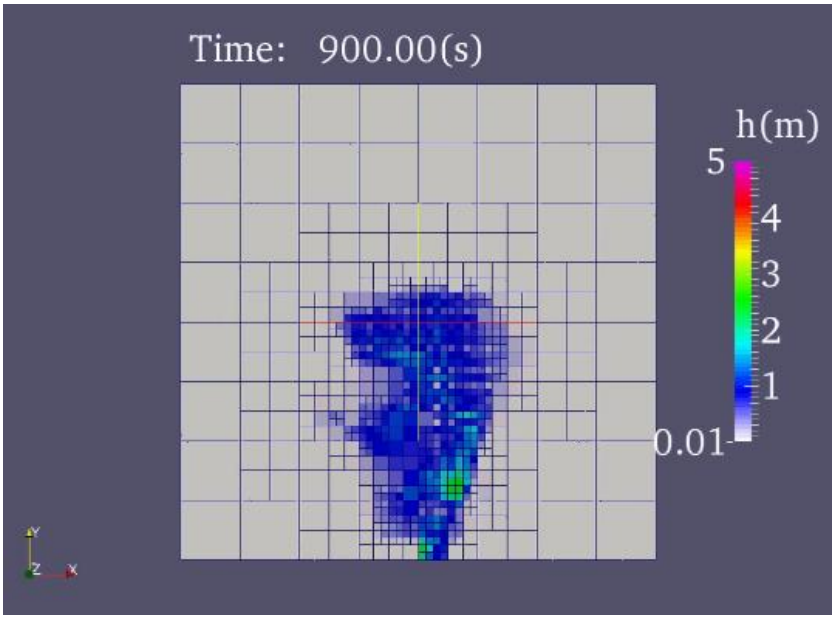
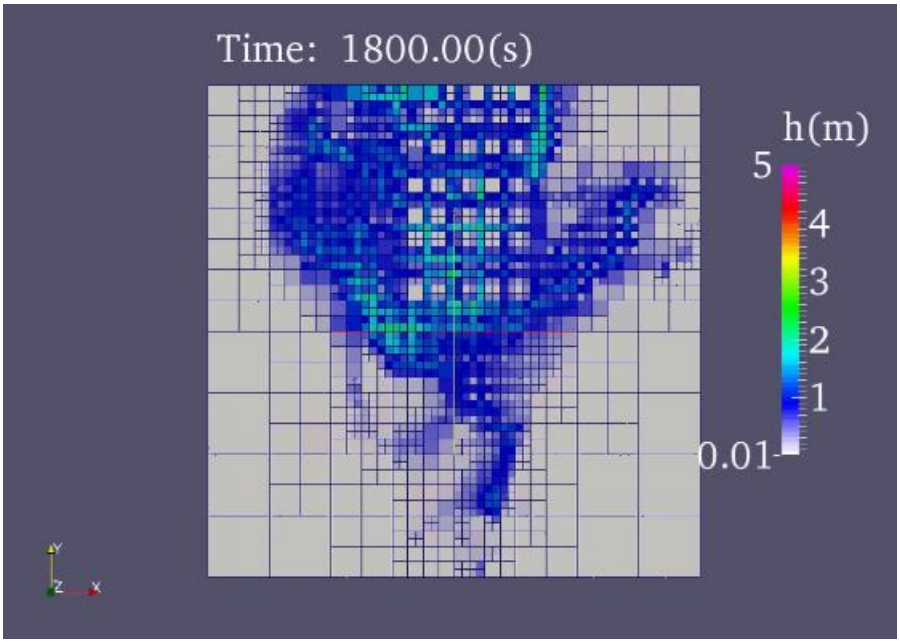
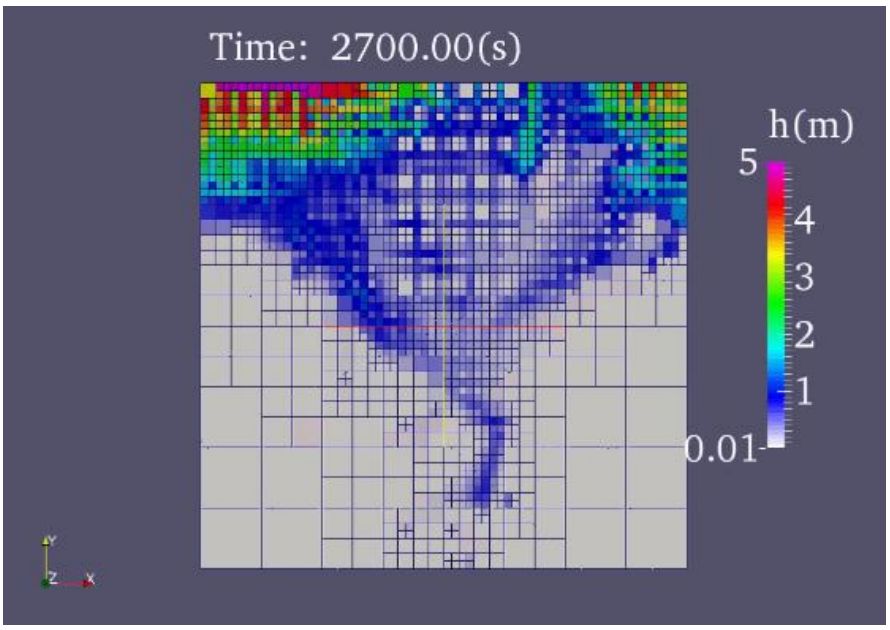


Figure 4.25. The water depths along the black line (showed in Fig. 1b) at 6000s bewteen AMR5 and the fine grid case on non-uniform discharge case.


Time: 900.00(s)
h(m)
5
4
3
2
1
0.01


Time: 1800.00(s)
h(m)
5
4
3
2
1
0.01



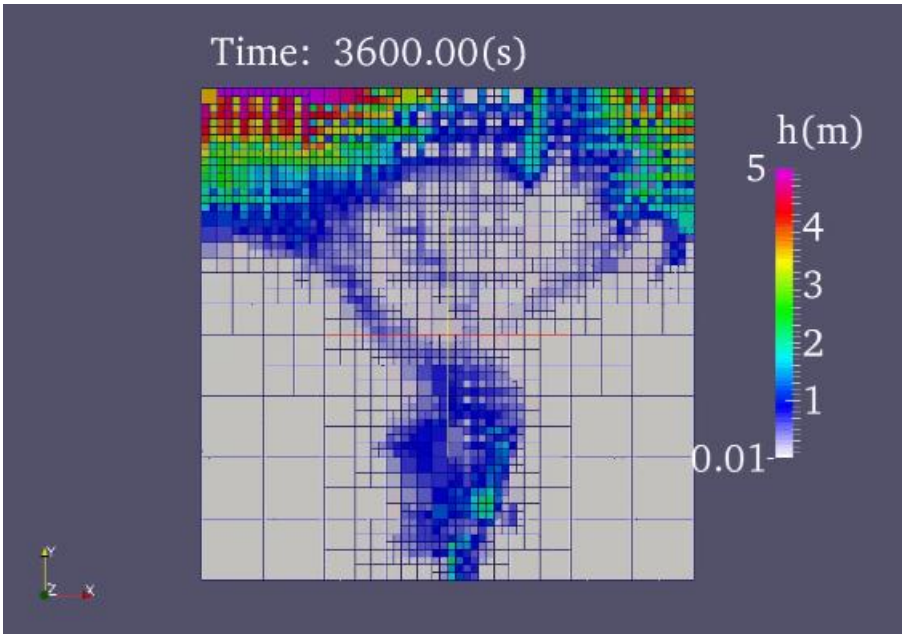


Figure 4.26. flood propagation

4.14 Test cases3: Multiple-inflow

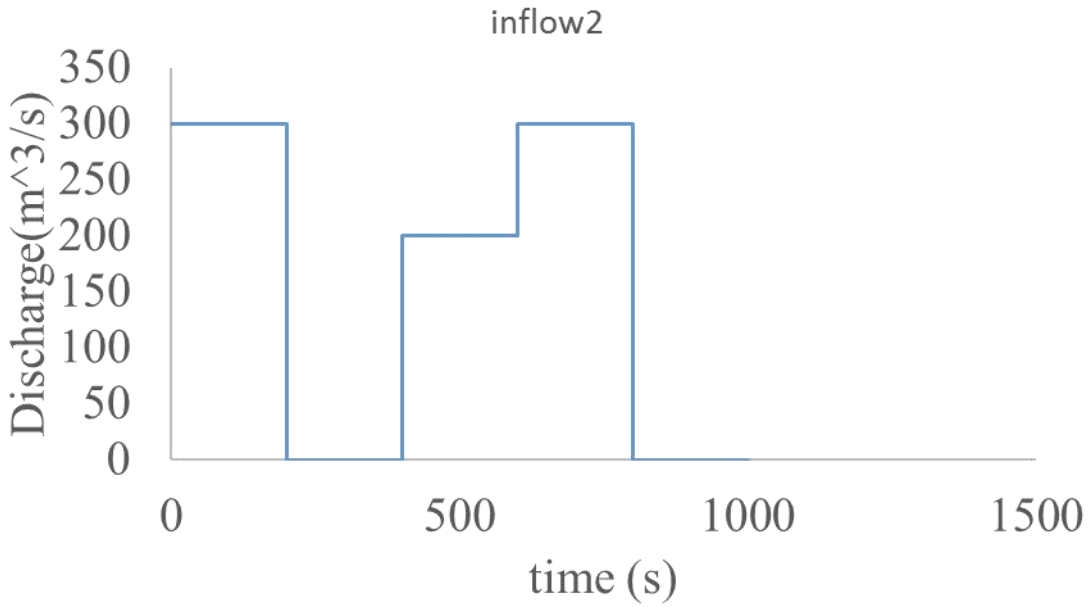


Figure 4.27. The discharge of second inflow changes along the time

4.14.1 Results and discussion

AMR case saves 77.4% time and without loss of accuracy.

Table Computational time

Method	Computational Time
AMR	2h25min
Fine	10h43min

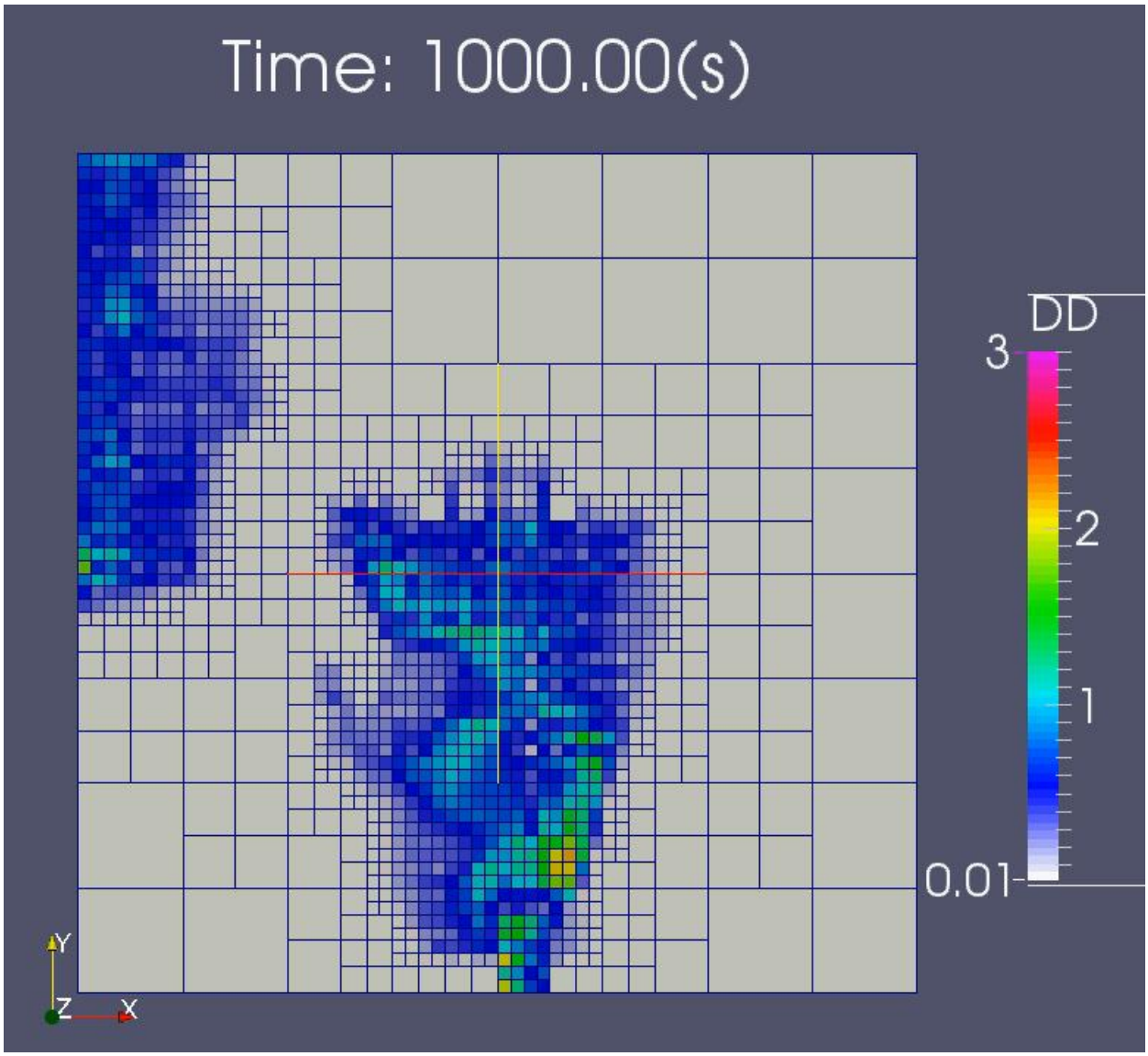


Figure 4.28 multiple inflow simulated by AMR model at 1000s

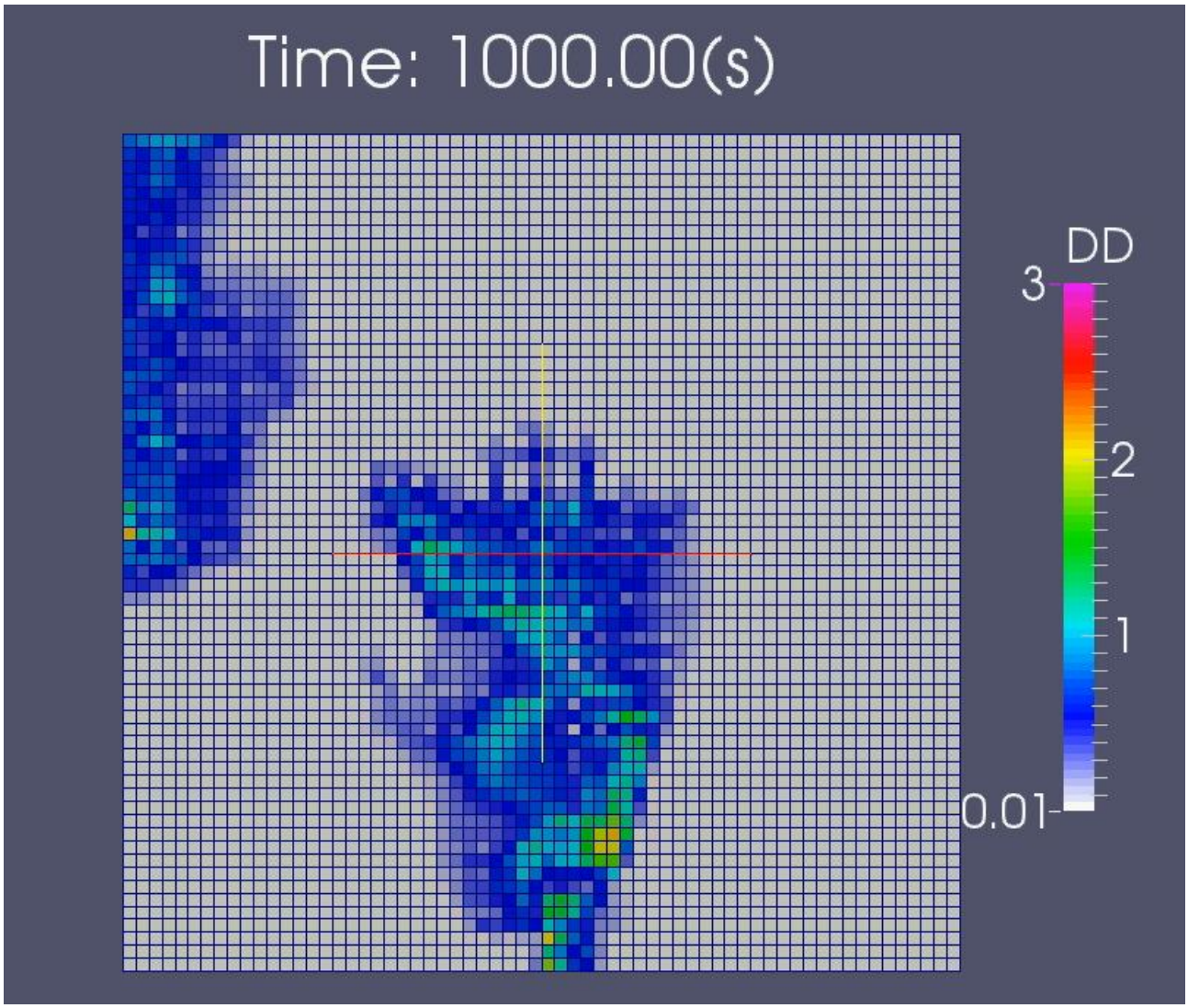


Figure 4.29 multiple inflow simulated by AMR model at 1000s

Test case 3 Shows the results between the AMR model and Fine grid model are almost same. The AMR model saves saves 77.4% time

4.2 The evaluation of the refinement criteria and the efficiency of the flood simulation

Table 2 shows several refinement criteria we tested (defined as AMR1 to AMR7) and resulted computational time for flood simulation. In addition, Figure 7 shows the temporal change of the number of the blocks on the various refinement criteria and the fine grid case, and Figure 8 shows simulated water depth in the downstream direction along the black line in the Fig. 1a. We firstly show the effect of threshold value and refinement level for the refinement

in same criteria in the runs of AMR2, 3, 4. Among these runs, AMR4 is the only case reproducing the result obtained by the fine grid model due to the appropriate threshold value and refinement level, respectively. AMR2 gives the shortest computational time among all computational run, however, the computational result is not acceptable as shown in Fig. 8. This is because that the maximum refinement level which is equal to 6 is insufficient to resolve the highly urbanized area in the computational domain. The AMR3 generates much computational blocks, so that it needed the computational time (efficiency compared with fine grid case is just 47%). In this ARM3, the lower value set for de-refinement is close to the upper value set for refinement (the lower value is too large), which leads to the grid coarsening and inaccurate water depth on the high density building area compared with the fine grid model. This result suggests that suitable values has to be calibrated for the refinement level and threshold value.

Table 2, Figs. 7 and 8 also show that AMR 1, 5, 6 and 7 obtained the similar computational time and accuracy, which results are also same with the fine grid model as shown in Fig. 8. Among them, AMR5 is the most efficient, which saves 68.2% time. The physical meaning of this criteria is water surface slope, so that it may be relatively easy to calibrate the parameter in the physical sense rather than other parameters we tested.

Chapter 5-Conclusions

Conclusions

1.To solve the shallow water equations, the fractional step was used. For the Nonadvection term, the implicit method is used for numerical stability and accuracy for unsteady fluid flows. For the advection term, the CIP method was used to prevent the numerical diffusion and keep it accuracy in grid cell.
2.For the uniform discharge, AMR can save 66.0%-68.2% computational time without significant loss of computational accuracy. The accuracy and efficiency differ compared with Fine grid case, depending on the refinement criteria, refinement levels, and the threshold values. The best refinement criteria is water surface slope in this case.
3.In the nonuniform discharge case, the water surface slope still can capture the flow characteristics well and saved 43% time.
4.In the multiple inflow case, the model with AMR saves 77.4% time without loss of accuracy compared with fine grid case

Future work

1.A novel wet-dry boundary tracking model should be proposed.
2.The interrelation between rainfall (driver) and the corresponding response processes, including surface runoff, pipe flow, overflow activity, as well as impacts on the adjacent compartments, such as sewage treatment plant, groundwater, and rivers, should be studied further
3.Combining the numerical modeling with modern sensors and innovative wireless data transfer technologies to solve the realistic urban management problem.

REFERENCES

https://web.archive.org/web/20021025100606/http://sdcd.gsfc.nasa.gov/RIB/repositories/inhouse_gsfc/Users_manual/amr.html

Ali, M. A., Kimura, I. and Shimizu, Y. (2016). Flood modelling using sub-grid based finite volume approach & constrained interpolation profile method. In Proceedings of the International Conference on Fluvial Hydraulics (River Flow 2016). Iowa City, USA: CRC Press, 1891-1895.

Bates, P. D., Horritt, M. S., and Fewtrell, T. J. (2010). A simple inertial formulation of the shallow water equations for efficient two-dimensional flood inundation modelling.Journal of Hydrology,387(1-2), 33-45.

Blaise, S., & St-Cyr, A. (2012). A dynamic hp-adaptive discontinuous Galerkin method for shallow-water flows on the sphere with application to a global tsunami simulation. Monthly Weather Review, 140(3), 978-996.

Berger, M.J., Colella, P. (1989). Local adaptive mesh refinement for shock hydrodynamics. J. Comput. Phys. 82, 64–84. doi:10.1016/0021-9991(89)90035-1

Berger, M.J., Oliger, J. (1984). Adaptive mesh refinement for hyperbolic partial differential equations. J. Comput. Phys. 53(3), 484–512. doi:10.1016/0021-9991(84)90073-1415– 427.

Bradford, S. F., & Sanders, B. F. (2002). Finite-volume model for shallow-water flooding of arbitrary topography. Journal of Hydraulic Engineering, 128(3), 289-298.

Caian, M., Mic, R. P., Corbus, C., Angearu, C. V., & Matreata, M. (2021). Extreme flood modeling and mechanism over Crisul Alb basin in Romania. CATENA, 196, 104923.

Chen, A. S., Evans, B., Djordjević, S., and Savić, D. A. (2012). Multi-layered coarse grid modelling in 2D urban flood simulations.Journal of Hydrology,470, 1-11.

Cobby, D. M., Mason, D. C., and Davenport, I. J. (2001). Image processing of airborne scanning laser altimetry data for improved river flood modelling.ISPRS Journal of Photogrammetry and Remote Sensing,56(2), 121-138.

Courant, R., Friedrichs, K., & Lewy, H. (1928).About the partial difference equations of mathematical physics.Mathematical annals,100(1), 32-74.

DESA, U. (2018). World urbanization prospects: the 2018 revision, key facts. New York: NY. Available online at: https://population. un. org/wup/Publications/(Accessed December 20, 2018).

Gentry, R.A., Martin, R.E., Daly, B.J., 1966. An Eulerian differencing method for unsteady compressible flow problems. Journal of Computational Physics 1, 87–118.. doi:10.1016/0021-9991(66)90014-3

GONG, W., SHIMIZU, Y., & IWASAKI, T., (2020). A Case Study of Flood Modeling with Adaptive Mesh Refinement.

Hankin, B., Waller, S., Astle, G., and Kellagher, R. (2008). Mapping space for water: screening for urban flash flooding.Journal of Flood Risk Management,1(1), 13-22.

Hoang, H. M. (2005). A parallel adaptive finite difference algorithm for petroleum reservoir simulation.

Hu, R., Fang, F., Salinas, P., & Pain, C. C. (2018). Unstructured mesh adaptivity for urban flooding modelling. Journal of hydrology, 560, 354-363.

Huang, W., Cao, Z., Pender, G., Liu, Q., and Carling, P. (2015). Coupled flood and sediment transport modelling with adaptive mesh refinement.Science China Technological Sciences, 58(8), 1425-1438.

Jang, C.L. and Shimizu, Y. (2005). Numerical simulation of relatively wide, shallow channels with erodible banks. Journal of Hydraulic Engineering, 131(7), 565-575.

Kajishima, T., & Taira, K. (2017). Computational fluid dynamics. Cham: Springer International Publishing, (1).

Koltakov, S., & Fringer, O. B. (2013). Moving grid method for numerical simulation of stratified flows. International Journal for Numerical Methods in Fluids, 71(12), 1524-1545.

Lin, P. (2007). A fixed-grid model for simulation of a moving body in free surface flows. Computers & fluids, 36(3), 549-561.

Liu, Y., and Pender, G. (2010). A new rapid flood inundation model. In proceedings of the first IAHR European Congress, 4-6.

Levy, D., Powell, K., Va, B., 1989. An implementation of a grid-independent upwind scheme for the Euler equations, in: .. doi:10.2514/6.1989-1931

Lowe, J. A., Bernie, D., Bett, P., Bricheno, L., Brown, S., Calvert, D., ... & Belcher, S. (2018). UKCP18 science overview report. Exeter, UK: Met Office Hadley Centre.

MacNeice, P., Olson, K.M., Mobarry, C., Fainchtein, R. and Packer, C. (2000). PARAMESH: A parallel Adaptive Mesh Refinement Community Toolkit. Computer Physics Communications, 126(3), 330-354.

Madzvamuse, A., Maini, P. K., & Wathen, A. J. (2005). A moving grid finite element method for the simulation of pattern generation by Turing models on growing domains. Journal of Scientific Computing, 24(2), 247-262.

Morikawa, G.(2019). Numerical analysis of flood with hyper grid model, dissertation for Master degree of Hokkaido University, Sapporo, Japan.

Morikawa, G. and Kimura, I. (2018). Numerical analysis of flood with a double grid model. In E3S Web of Conferences (Vol. 40, p. 05042). EDP Sciences.

Miura, S., Kawamura, I., Kimura, I., and Miura, A. (2011). Study on inundation flow analysis method in densely populated urban area on alluvial fan.Journal of Japan Society of Civil Engineers, Ser. B1 (Hydraulic Engineering),67, I_979-I_984.

Neal, J. C., Fewtrell, T. J., Bates, P. D., and Wright, N. G. (2010). A comparison of three parallelization methods for 2D flood inundation models.Environmental Modelling & Software,25(4), 398-411.

Olson, K. (2006). PARAMESH: A parallel, adaptive grid tool. In Parallel Computational Fluid Dynamics 2005 (pp. 341-348). Elsevier.

Petzold, L. R. (1987). Observations on an adaptive moving grid method for one-dimensional systems of partial differential equations. Applied Numerical Mathematics, 3(4), 347-360.

Pons, K., Golay, F., & Marcer, R. (2017). Adaptive mesh refinement method applied to shallow water model: A mass conservative projection. Topical problems of fluid mechanics.

Semper, B., & Liao, G. (1995). A moving grid finite-element method using grid deformation. Numerical Methods for Partial Differential Equations, 11(6), 603-615.

Takewaki, H., Nishigushi, A., & Yabe, T. (1985). Cubic interpolated pseudo-particle method (CIP) for solving hyperbolic type equation. National Institute for Fusion Science. NII-Electronic Library Services.

UN. 2018 World urbanisation prospects: the 2018 revision key facts. See https://esa.un.org/unpd/wup/Publications/Files/WUP2018-KeyFacts.pdf (accessed 18 March 2019).

Wang, J. P., and Liang, Q. (2011). Testing a new adaptive grid based shallow flow model for different types of flood simulations. Journal of Flood Risk Management, 4(2), 96-103.

Yabe, T., Xiao, F., Utsumi, T., 2001. The Constrained Interpolation Profile Method for Multiphase Analysis. Journal of Computational Physics 169, 556–593.. doi:10.1006/jcph.2000.6625

O'Donnell, E. C., & Thorne, C. R. (2020). Drivers of future urban flood risk. Philosophical Transactions of the Royal Society A, 378(2168), 20190216.

Zhao, D. H., Shen, H. W., Tabios III, G. Q., Lai, J. S., & Tan, W. Y. (1994). Finite-volume two-dimensional unsteady-flow model for river basins. Journal of Hydraulic Engineering, 120(7), 863-883.