Development of Cut Cell Methods for Barrier Simulations with Shallow Water Equations

Chanyang Ryoo

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2022

Abstract

Development of Cut Cell Methods for Barrier Simulations with Shallow Water Equations

Chanyang Ryoo

In this thesis we aim to provide computationally efficient methods of performing water barrier simulations. The innate challenge in simulations of structures such as sea or surge barriers is resolution. Because barriers tend to be long and thin compared to the surrounding landscapes they protect, one must put mesh refinement on the barrier region in order to even numerically recognize the barrier's presence. This is a costly computation due to the CFL condition which puts a strict limit on the size of time step proportional to the spatial mesh size. Another issue is the complexity of meshing near the barrier. Since barriers are most likely slanted or have certain shapes, the grid has to reflect this in the form of a grid mapping or an unstructured grid.

To mitigate the issue of resolution, we propose an approximation of the barrier with a line interface embedded on a Cartesian grid, reducing our problem to an embedded boundary problem. Then to avoid complex meshing, we develop three *cut cell methods* on two shapes of barriers: 1) the h-box method (HB), 2) the state redistribution method (SRD), and 3) the cell merging method (CM). Doing this two-step approach means that we can lower the resolution near the barrier region and still feel the presence of the barrier and capture its effect, which would otherwise not be the case if we relied on resolution for representation of the barrier. This does not mean that we are losing accuracy by lowering resolution, however. Rather, we are maintaining about the same accuracy while also lowering resolution (and thus cutting computational cost), which we show by comparison with a refined barrier. We solve the shallow water equations as our underlying PDEs to simulate water interaction with the barrier, as they are commonly used in tsunami and storm simulations. We implement our work on the PyClaw [1] framework, which is an objected oriented program that solves conservation laws.

[1] An open sourceware for solving hyperbolic conservation equations.

Table of Contents

List of Figures

List of Tables

Acknowledgements

"Fear of the Lord is the beginning of wisdom" - Proverbs 9:10

I would like to thank Jesus Christ my God who forgives me and guides me. I also thank my advisor Kyle Mandli, who has been both a mentor and a friend to me. I thank my dad for his daily check-ins and my roommates for their home-cooked meals when I was busy.

Chapter 1: Introduction and Background

1.1 Motivation

In light of sea level rises and projected increase in storms and surges [1], people around the world have been building barriers off deltas, between islands, and on rivers. The most notable example of this is in Netherlands, where numerous surge barriers have been built under a project called “Delta Works". Talks of possibly building a similar barrier in New York City are in progress (Fig. 1.1). The hope of building such barriers is to reduce flooding and its associated infrastructural and financial damages.

(a) Linear (b) V shaped

Figure 1.1: Two proposed shapes of barriers for NYC. (Left: Wikipedia, right: [2].)

To test their efficiency, however, one must run a simulation comparing how far a certain storm-caused flooding will reach with and without the barrier. The simulation must reveal what protection they can provide to the city as a whole. Such a task is an interdisciplinary one and an active area of research [3, 4]. Three-dimensional simulations are available online[1] of how the barrier may

[1] See videos here: video 1, video 2

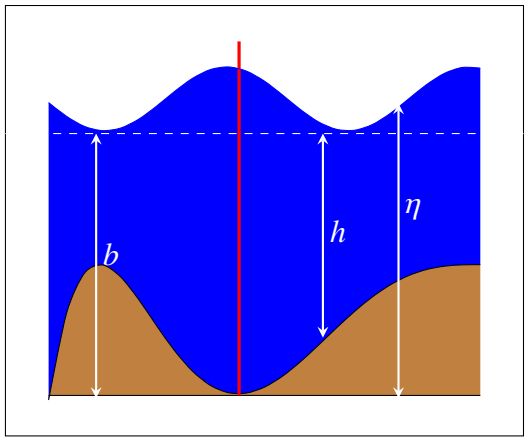


Figure 1.2: 1D diagram showing motivation and possible scenario of a storm barrier (red) in action, along with some measurement variables: the bathymetry b measured from fixed level (dashed), the height h of water column, and the surface level $\eta = b + h$.

weather a windy climate (e.g. by Arcadis and CH2M Hill, engineering consulting companies [5]). Three-dimensional simulations that evaluate the region-wide effect of a storm, however, is both unwarranted and computationally intractable. As a consequence modeling coastal flooding scenarios, especially with optimization, usually involves a reduction to two-dimensional equations.

Unfortunately another problem arises even in a two-dimensional simulation. If one represents the barrier as part of the ground level (i.e. via elevating the bathymetry), then the barrier is going to be very small and narrow in width compared to the overall city landscape, which then requires much higher resolution and much more computation around the barrier.

A work around this issue is where our model problem finds its origination: the problem will approximate the physical barrier with a *zero width* barrier. Instead of a bathymetric variation, it will be akin to an embedded boundary interface that mimics a fixed, stationary barrier which allows both reflection and overtopping of waves (see Fig. 1.2).

Another approximation we use that is different between our problem and some already existing 3D simulations is that the equations we solve will be the shallow water equations (SWE) instead of the more complicated Navier-Stokes equations. They have been used to model tsunamis and storms as well [6].

1.2 Embedded boundary and cut cells

Furthermore, SWE are an example of hyperbolic PDEs, and hyperbolic PDEs with embedded boundaries are an important class of problems in computational fluid dynamics, another example being the Euler equations [7, 8]. Embedded boundary problems are those in which the grid is Cartesian and has boundary elements overlapping it, producing *cut cells* or *small cells* (size $< 0.5h$, with h being size of regular mesh). A 1D example of a cut cell caused by a barrier is shown in Fig. 2.1. This is in contrast to problems using boundary *adapted* grids which rotate grids with respect to the boundaries and thereby avoid creating cut cells. Embedded boundary grids provide much simpler calculations away from the boundaries and require only special treatment near the boundaries, whereas using boundary adapted grids requires a *global* transformation of grid directions and is specific to the given boundaries. This provides incentives to prefer embedded boundary grids.

However, using embedded boundaries are still computationally challenging because of the cut cells they produce [9], which impose a strict time step condition due to the CFL condition (i.e. the prevention of wave propagation beyond mesh size [10]). Various methods have been developed to tackle this problem and are collectively called *cut cell methods* [11, 12, 13, 14, 15, 16, 17, 18, 19].

The novelty of our work is that here we are presenting a unique embedded boundary where the boundary allows flux across it. This is because the boundary represents an overtoppable barrier and is parameterized by a variable height. SWE with cut cells (no flux across boundary, or the case of “infinite" height) have been dealt with elsewhere [20].

1.3 Prior Work

There are largely two types of research done for barrier simulations. One is simulations done “up close", looking at the velocity profile of water through a barrier and the flow change near the barrier. Examples of this include research done by Army Corp of Engineers [21] and by Delta Works [22]. This type of research mostly uses the finite element method with unstructured grids

and is concerned about detailed design of the barrier, which we are unconcerned about. Some have even suggested what type of material should be used for building the barriers [23], where finite element method is again used for the barriers themselves.

A second type of research is for storm protective strategies. This type of research uses software such as ADCIRC, SLOSH, SWAN, and GEOCLAW to look at the more global scale of storms [24, 25]. ADCIRC uses the finite element method; SLOSH uses finite difference; while SWAN and GEOCLAW uses finite volume. It is to be remarked that SLOSH also approximates barriers as a line interface [26]. However, the difference is that it uses conformal mapping and makes sure that the barrier runs along the mapped grid edge. Our work allows flexibility of grid placement and does not require grid alignment of the barrier. Furthermore, we use finite volume methods which captures discontinuities such as shocks in the numerical solutions.

None of these work cited use cut cell methods whose benefits are freedom in barrier placement and relaxed CFL condition. We provide here the basics of our cut cell methods, their numerical accuracy, computational benefits and some realistic scenarios, and we remark that our methods can be adapted and further implemented in any finite volume storm simulating software in the future.

Chapter 2: Model Equations and Problem Setup

Shallow water equations (SWE) are a system of hyperbolic partial differential equations modeling horizontal, surface water motion. There are source terms to construct a geophysical version of SWE [27], for which solvers like GeoClaw exists, but for simplicity we only consider bathymetric source terms. Also a note is that in our numerical experiments, we perform dimensionless calculations and set $g = 1$, with the assumption that the physics of the equations remain the same regardless of our choice of g.

2.1 1D shallow water equations

First we consider the 1D problem. The set of equations that we will be solving is:

$$h_t + (hu)_x = 0 \tag{2.1}$$

$$(hu)_t + \left(\frac{1}{2}gh^2 + hu^2\right)_x = -ghb_x, \tag{2.2}$$

where h is the water height, u depth-averaged water velocity, b the bathymetry, and g the gravitational constant. In case there is no bathymetric change, these equations describe conservation of mass and momentum $q = [h, hu]$.

In vector notation, therefore, the SWE becomes:

$$q_t + f(q)_x = \Psi(q), \tag{2.3}$$

where $q = [h, hu]$, $f(q) = f(h, hu) = [hu, \frac{1}{2}gh^2 + (hu)^2/h]$, and $\Psi(q) = [0, -ghb_x]$.

A 1D diagram of the problem at hand is shown in (Fig. 2.1). As can be seen in the figure, the arbitrary barrier placement along the grid creates a cut of cell sized $\alpha\Delta x$ and $(1-\alpha)\Delta x$ with some

$\alpha > 0$.

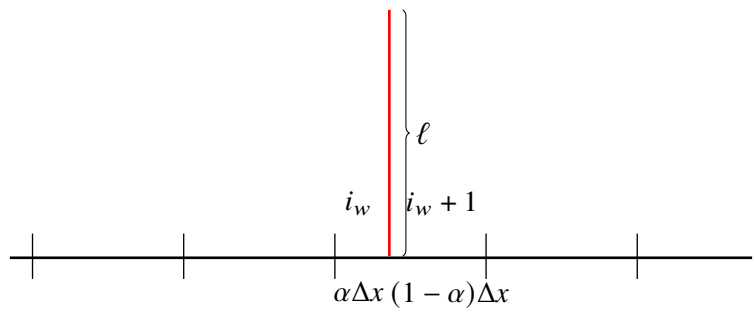


Figure 2.1: 1D setup, example of a barrier with variable height ℓ. Cut cells have index $i = i_w, i_w + 1$

2.2 2D shallow water equations

2D SWE can also be written as a set of conservation equations [28]:

$$h_t + (hu)_x + (hv)_y = 0 \tag{2.4}$$

$$(hu)_t + \left(\frac{1}{2}gh^2 + hu^2\right)_x + (huv)_y = -ghb_x \tag{2.5}$$

$$(hv)_t + (huv)_x + \left(\frac{1}{2}gh^2 + hv^2\right)_y = -ghb_y, \tag{2.6}$$

for $(x, y) \in \Omega \subset \mathbb{R}^2$, where h represents the height of water, u and v the x-velocity and y-velocity, g the gravitational constant, and $b = b(x, y)$ the bathymetry. In the first equation, we have conservation of mass h and in the second and third equations, if $\nabla b \equiv 0$, conservation of momentum. In the presence of bathymetric variation there is loss of conservation of momentum.

We can thus represent the SWE in the following form Eq. (2.7) as derived in [10]. If $q = [h, hu, hv]$, $f(q) = [q_2, \frac{1}{2}gq_1^2 + \frac{q_2^2}{q_1}, \frac{q_2 q_3}{q_1}]$, $g(q) = [q_3, \frac{q_2 q_3}{q_1}, \frac{1}{2}gq_1^2 + \frac{q_3^2}{q_1}]$, and $\Psi(q, b) = [0, -gq_1 b_x, -gq_1 b_y]$, then we have for our SWE:

$$q_t + f(q)_x + g(q)_y = \Psi(q, b). \tag{2.7}$$

In 2D, we will set up two types of barrier on our grid, the slanted and the V shaped barrier, as

suggested by Fig. 1.1 and seen in Fig. 2.2. Furthermore, we provide in the Appendix the algorithms used to find the cut cell data and the Fortran-Python module we built to automate and customize this process.

(a) Model problem 1: slanted barrier. (b) Model problem 2: V-barrier

Figure 2.2: Grid setup of two model barrier problems. 1, 2, and 3 denote the vertices of the barriers.

2.3 Properties of solutions to SWE

2.3.1 Speed vs. velocity

In SWE, the velocity of a water column is a state attribute denoted by either u or v. However, this is different from a speed of a wave. A wave can be thought of as a “group" of state values traveling together across the physical domain. Mathematically, a wave is the difference of two water states or flux vectors, and a speed is the eigenvalue corresponding to the eigen-decomposition of that wave, using eigenvectors of the Jacobians $f'(q), g'(q)$. We denote the speed by λ and we usually have $\lambda = u \pm \sqrt{gh}$. Note that even when initial u is zero, we can still have nonzero speed $\sqrt{gh}$. This is called a gravity wave. On the other hand, it is possible to have a wave speed of zero, where the water states involved have nonzero velocity. This occurs in steady state solutions.

This distinction is important in categorizations of types of flows, which is done using a parameter called the *Froude number*:

$$Fr = \frac{|u|}{\sqrt{gh}}.$$

If Froude number is greater than 1, then the velocity associated with a wave is greater than the speed. This is called a supercritical flow. If it is less than 1, then the velocity is less than the speed. This is called a subcritical flow. When it is equal to 1, we have critical flow.

2.3.2 Type of waves

Furthermore, there are four categorizations of waves. First is subsonic wave. This is when all the speeds associated with a wave are negative. Second is supersonic wave, which is when all the speeds are positive. Third is transonic wave, when there are both positive and negative speeds associated with a wave. Finally, there is the stationary wave, whose speed is zero.

As we will see in the following chapters, SWE allow nonlinear solution forms, which include shocks and rarefactions. These all arise because of the nonlinear nature of the waves' speed profile. Sometimes, the speed profile cross within itself (shocks) or fan out (rarefaction). Also, in 2D SWE there is a third linear type of wave called the linearly degenerate wave. This wave is what propagates the transverse momentum across the physical domain.

A final comment on the property of solutions to SWE is that depending on the source terms introduced to the equations, there arise completely different type of waves. We shall only consider variable bathymetry as our source term. This amounts to the solution only containing gravity waves, which are waves that are generated or influenced by the graviational force only. However, there are the following possible source terms and corresponding wave solutions:

- Coriolis force f: This adds source terms fhu and fhv. Poincare waves and Kelvin waves result from nonzero f
- Drag coefficient k: This adds the source terms khu and khv. This affects the speed of a wave.
- Viscosity coefficient ν: This adds the source terms $\nu((hu)_{xx} + (hu)_{yy})$, $\nu((hv)_{xx} + (hv)_{yy})$ and affects the momentum calculations.

Chapter 3: Finite volume methods

3.1 Flux Difference Form

We employ a finite volume method to solve SWE. Finite volume methods seek to approximate volume averages of state variables over a control volume V_i (a grid cell $[x_{i-1/2}, x_{i+1/2}] \times [y_{j-1/2}, y_{j+1/2}]$) and their evolution. This can be seen by taking volume integral of the original hyperbolic equations:

$$\int_{V_i} \left(q_t + f(q)_x + g(q)_y\right) dV = \int_{V_i} \Psi(q, b)\, dV \tag{3.1}$$

$$\frac{dQ_i}{dt}|V_i| + \int_{V_i} f(q)_x\, dV + \int_{V_i} g(q)_y\, dV = \int_{V_i} \Psi(q, b)\, dV \tag{3.2}$$

$$\frac{dQ_i}{dt} + \frac{1}{|V_i|} \oint_{S_i} (f(q) + g(q)) \cdot n\, dS = \frac{1}{|V_i|} \int_{V_i} \Psi(q, b)\, dV, \tag{3.3}$$

where $Q_i(t)|V_i| = \int_{V_i} q dV$ (via mean value theorem), $|V_i|$ the volume of V_i, S_i the surface area of V_i (via Gauss's theorem), and n is the normal surface vector of S_i. As can be seen by the scalar $1/|V_i|$, we now work with volume averages.

The usual finite volume method in 2D is then to divide the domain into grid cells V_i, compute (or initialize) state averages over V_i, calculate fluxes on each edge of the grid cell (S_i), and update the state average Q_i via those fluxes, while taking into account any source term averages.

In a uniform 2D Cartesian coordinate with $\Psi = 0$, Eq. (3.3) becomes:

$$\frac{dQ_i}{dt} + \frac{1}{\Delta x \Delta y}\Big[\int_{y_{j-1/2}}^{y_{j+1/2}} f(q(t, x_{i+1/2}, y))dy - \int_{y_{j-1/2}}^{y_{j+1/2}} f(q(t, x_{i-1/2}, y))dy \\ + \int_{x_{i-1/2}}^{x_{i+1/2}} g(q(t, x, y_{i+1/2}))dx - \int_{x_{i-1/2}}^{x_{i+1/2}} g(q(t, x, y_{i-1/2}))dx \Big] = 0. \tag{3.4}$$

Eq. (3.4) is then numerically approximated by integrating in time from t_i to t_{i+1} to get:

$$Q_i^{n+1} = Q_i^n - \frac{1}{\Delta x}(F_{i+1/2,j} - F_{i-1/2,j}) - \frac{1}{\Delta y}(G_{i,j+1/2} - G_{i,j-1/2}) \tag{3.5}$$

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left(\frac{F_{i+1/2,j} - F_{i-1/2,j}}{\Delta t}\right) - \frac{\Delta t}{\Delta y}\left(\frac{G_{i,j+1/2} - G_{i,j-1/2}}{\Delta t}\right), \tag{3.6}$$

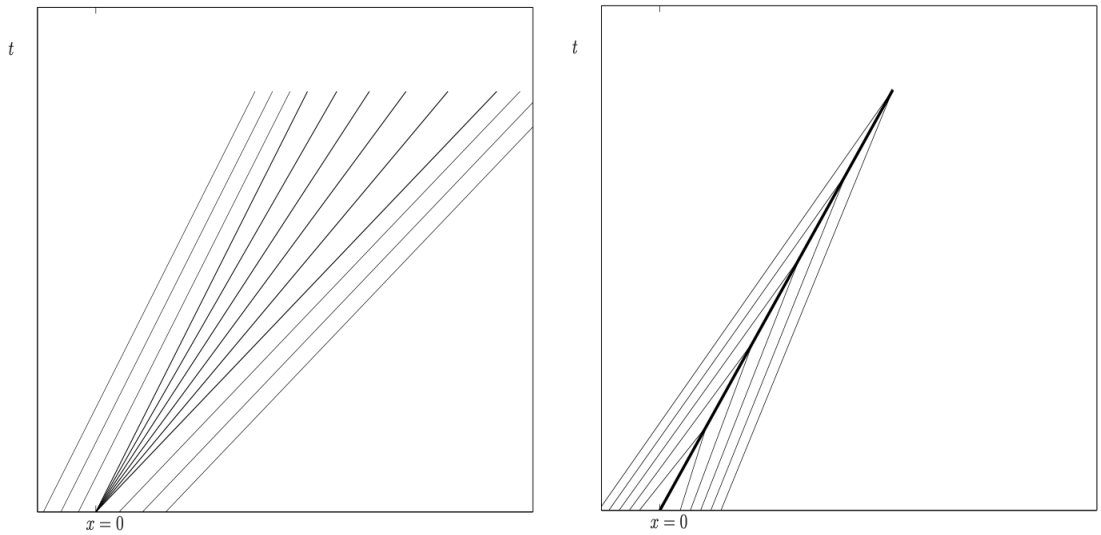
where $F_{i\pm1/2,j} \approx \frac{1}{\Delta y}\int_{t_i}^{t_{i+1}}\int_{y_{j-1/2}}^{y_{j+1/2}} f(q(x\pm1/2))dydt$ and $G_{i,j\pm1/2} \approx \frac{1}{\Delta x}\int_{t_i}^{t_{i+1}}\int_{x_{i-1/2}}^{x_{i+1/2}} g(q(y\pm1/2))dxdt$ and we added the Δt term on the denominator to get a time average of the integrals and on the numerator to cancel out the denominator. This seemingly suggests the timestep variable is artificially added into the numerical scheme. Note, however, that the time variable is implicit in the integral limits. Furthermore, if we treat the inner spatial integral as a constant with respect to time, the timestep $\Delta t = t_{i+1} - t_i$ will come out. This form shown in Eq. (3.6) is called the flux difference form. The flux integrals are approximated using many different ways, e.g. using Taylor's approximations or approximating $q(x \pm 1/2)$ (Godunov's method) [10].

3.2 Riemann Problem

The hyperbolic equation limited to just two states on either side of a grid edge is called a *Riemann problem* (RP). This means that in a 1D problem, two RP's must be solved to update one grid cell, whereas in 2D four must be solved. The bulk of the work done in a finite volume solver is therefore in solving RPs. There is much technical detail that can be discussed about Riemann problems which we do not include in this thesis, as it has been dealt with elsewhere (e.g. entropy, numerical viscosity, Riemann invariants, etc.) [29, 30]. We only briefly mention two interesting phenomena that occur in solutions to nonlinear hyperbolic RPs, such as in SWE.

3.2.1 Shocks

Shocks are a discontinuity in a solution that travel along the domain in time. They occur when the characteristics curves (where $dq/dt = 0$) collide with each other (Fig. 3.1). This means that in one region of the domain, mass is propagating in one particular direction **x** faster than the



(a) Rarefaction (b) Shock

Figure 3.1: Characteristic curves of different nonlinear wave (from [30])

mass lying in that direction. Difficulties with hyperbolic problems such as SWE is that shocks can form even with continuous initial condition, and that the PDE does not hold locally in such discontinuities. This is why finite volume methods are used, with integrals that hold true even with discontinuities. Another issue is that shocks travel with some speed and numerical solvers must track them. The speed at which they travel is given by the Rankine-Hugoniot condition:

$$s(q_r - q_l) = f(q_r) - f(q_l), \tag{3.7}$$

where s is the speed and q_l, q_r are the states on either side of the shock discontinuity, and $f(q)$ is the flux function. The equation above would be the shock speed in the x-direction in our notation of the 2D problem.

3.2.2 Rarefactions

Rarefactions occur when characteristics curve diverge away from each other and leave a gap in between, which is where the rarefaction occurs (Fig. 3.1). Here, the solution of the hyperbolic RP is continuous from one end of the diverging characteristic line to the other. This is accomplished

by a similarity solution $q(x,t) = \hat{q}(x/t)$. Inserting this solution form into the hyperbolic equation (Eq. (2.3) with $\Psi = 0$) gives the relation:

$$f'(\hat{q}(x/t))\hat{q}'(x/t) = \frac{x}{t}\hat{q}'(x/t). \tag{3.8}$$

This equation is solved for $\hat{q}$ using standard ODE methods as now we have a first order differential equation of just $\hat{q}$.

3.3 Wave Propagation

Throughout this paper we will sometimes use the standard flux difference form to solve SWE but also the wave propagation algorithm. It is this last algorithm we give notation and reference to here, as it is used in the actual numerical solvers.

Wave propagation is an algorithm [10] that uses the Jacobian of the hyperbolic equation and its eigenvectors to construct *speed* (the eigenvalues) and *waves* (the eigenvectors) from a Riemann problem (RP) and amasses the waves that contribute to the right state or the left called *fluctuations*. Their direction is determined by the sign of the corresponding eigenvalues or the speeds. The speeds are specially averaged eigenvalues of the left and right states and the waves are the corresponding eigenvectors. A note to be made is that the difference between the two state variables Q on either side of a RP is decomposed with the eigenvectors, but here we decompose the difference of *flux vectors*, which is sometimes called f-wave propagation. This method has the added benefit of always conserving mass, and we simply call this wave propagation.

3.3.1 1D Wave Propagation

The main part of the algorithm is taking the difference between the flux vectors of the right and left state and decomposing it using the speeds and waves [31]. For a RP between q_L, q_R, we have:

$$\sum_{i=1}^{M_\omega} \beta_i \mathbf{r}_i = f(q_R) - f(q_L), \tag{3.9}$$

where β_i are the weights of the eigenvectors $\mathbf{r}_i$ in the decomposition of the flux difference vector, $f(q_R) - f(q_L)$, and M_ω is the number of eigenvectors and speeds. In 1D SWE, M_ω is generally 2, although one can add another vector [14] and in 2D SWE, $M_\omega = 3$.

In 1D, the Jacobian of $f(q)$ takes the form

$$f'(q) = \begin{bmatrix} 0 & 1 \\ -u^2 + gh & 2u \end{bmatrix}, \tag{3.10}$$

whose eigenvectors $\mathbf{r}_i$ take the form

$$\mathbf{r}_i = \begin{bmatrix} 1 \\ \lambda_i \end{bmatrix},$$

where λ_i is the speed. For our work, we use the Einfeldt speed [32], which is: $\lambda_1 = \min(u_L - \sqrt{gh_L}, \hat{u} - \sqrt{g\overline{h}})$, $\lambda_2 = \max(u_R - \sqrt{gh_R}, \hat{u} + \sqrt{g\overline{h}})$, where $u_{L/R}$ is the velocity of the left/right state, $h_{L/R}$ the height of left/right state, $\overline{h}$ the arithmetic average of the left and right heights, and $\hat{u} = \frac{\sqrt{h_L}u_L + \sqrt{h_R}u_R}{\sqrt{h_L}+\sqrt{h_R}}$, also called Roe average velocity. Using this speed avoids producing entropy violating (i.e. nonphysical) solutions [33, 30].

Finally, we denote the right-going and left-going waves, respectively, as $\mathcal{A}^+\Delta Q = \sum_{i:\,\lambda_i>0} \beta_i \mathbf{r}_i$, $\mathcal{A}^-\Delta Q = \sum_{i:\,\lambda_i<0} \beta_i \mathbf{r}_i$. The right-moving waves are the addition of the decomposed eigenwaves whose corresponding speed is positive and the left-moving waves are the addition of those waves whose corresponding speed is negative. This means that in a regular grid, the update of a cell Q_i becomes:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}(\mathcal{A}^+\Delta Q_{i-1/2} + \mathcal{A}^-\Delta Q_{i+1/2}),$$

where $\mathcal{A}^{+/-}\Delta Q_{i-1/2}$ is the right or left fluctuations from the interface between cells q_i and q_{i-1}.

3.3.2 2D Wave Propagation

Let $Q_{i,j}$ denote the state variable $[H_{i,j}, (HU)_{i,j}, (HV)_{i,j}]$, $f(q)$ the flux vector in the x-direction and $g(q)$ the flux vector in the y-direction.

(a) Grid edges of cell (i, j) and its flux notation (b) Flux vectors used in wave propagation for blue cell

Figure 3.2: Flux scheme versus wave propagation

As shown in Fig. 3.2, instead of using fluxes at the edges of a cell, wave propagation linearly decomposes the difference of flux vectors from either side of each edge (Fig. 3.2b). The 2D Jacobians for $f(q)$ and $g(q)$ are given by:

$$f'(q) = \begin{bmatrix} 0 & 1 & 0 \\ -u^2 + gh & 2u & 0 \\ -uv & v & u \end{bmatrix}, \; g'(q) = \begin{bmatrix} 0 & 0 & 1 \\ -uv & v & u \\ -v^2 + gh & 0 & 2v. \end{bmatrix} \tag{3.11}$$

The matrices used for the linear decomposition for flux vectors $f(q)$ and $g(q)$ are given by the eigenvectors of the above matrices:

$$\mathcal{A} = \begin{bmatrix} 1 & 0 & 1 \\ U^* - \sqrt{gH^*} & 0 & U^* + \sqrt{gH^*} \\ V^* & 1 & V^* \end{bmatrix},$$
$$\mathcal{B} = \begin{bmatrix} 1 & 0 & 1 \\ U^* & 1 & U^* \\ V^* - \sqrt{gH^*} & 0 & V^* + \sqrt{gH^*} \end{bmatrix}, \tag{3.12}$$

respectively, where H^*, U^*, V^* represent special averages between $Q_{i,j}$ and neighbors, such as Roe

or Einfeldt average [32, 34]. The decomposition equations for the right and top edge are then

$$\mathcal{A}\beta = f(Q_{i+1,j}) - f(Q_{i,j}) \tag{3.13}$$

$$\mathcal{B}\hat{\beta} = g(Q_{i,j+1}) - g(Q_{i,j}), \tag{3.14}$$

where β, $\hat{\beta} \in \mathbb{R}^3$.

The columns of $\mathcal{A}$ and $\mathcal{B}$ Eq. (3.12) are the eigenvectors of the Jacobians $f'(q)$ and $g'(q)$, respectively. The eigenvalues corresponding to the pth column vector of $\mathcal{A}$ are $\{(\lambda_A)_p\}_{p=1}^3 = \{U^* - \sqrt{gH^*}, U^*, U^* + \sqrt{gH^*}\}$, and those corresponding to the pth column vector of $\mathcal{B}$ are $\{(\lambda_B)_p\}_{p=1}^3 = \{V^* - \sqrt{gH^*}, V^*, V^* + \sqrt{gH^*}\}$. Note that the second vector in the matrices above are the linearly degenerate waves that carry the transverse momentum huv.

Let $\{(\lambda_A)_p, \mathcal{A}_{(\lambda_A)_p}\}$ and $\{(\lambda_B)_p, \mathcal{B}_{(\lambda_B)_p}\}$ represent the pth eigenvalue-eigenvector pair of the Jacobians $f'(q)$ and $g'(q)$. Then we have the *$\mathcal{A}$-left* and *$\mathcal{A}$-right fluctuation waves* defined by

$$\mathcal{A}^-\Delta Q_{i+1/2,j} = \sum_{p:(\lambda_A)_p<0} \beta_p \mathcal{A}_{(\lambda_A)_p} \tag{3.15}$$

$$\mathcal{A}^+\Delta Q_{i-1/2,j} = \sum_{p:(\lambda_A)_p>0} \beta_p \mathcal{A}_{(\lambda_A)_p}, \tag{3.16}$$

where β is the difference coefficient vector in Eq. (3.13).

The *$\mathcal{B}$-up* and *$\mathcal{B}$-down fluctuation waves* are given by:

$$\mathcal{B}^-\Delta Q_{i,j-1/2} = \sum_{p:(\lambda_B)_p<0} \hat{\beta}_p \mathcal{B}_{(\lambda_B)_p} \tag{3.17}$$

$$\mathcal{B}^+\Delta Q_{i,j+1/2} = \sum_{p:(\lambda_B)_p>0} \hat{\beta}_p \mathcal{B}_{(\lambda_B)_p}, \tag{3.18}$$

where $\hat{\beta}$ is the difference coefficient vector in Eq. (3.14). The vectors in the summation $\beta_p \mathcal{A}_{(\lambda_A)_p}$ and $\hat{\beta}_p \mathcal{B}_{(\lambda_B)_p}$ represent how much of the flux difference is carried in the p eigenvector and are called weight vectors W_p. The sign of the speed $(\lambda_{A/B})_p$ indicates the direction in which the flux

difference is moved. Then the state update formula via the wave propagation method becomes:

$$Q_{i,j}^{n+1} = Q_{i,j}^{n} - \frac{\Delta t}{\Delta x}(\mathcal{A}^{-}\Delta Q_{i+1/2,j} + \mathcal{A}^{+}\Delta Q_{i-1/2,j}) \\ -\frac{\Delta t}{\Delta y}(\mathcal{B}^{-}\Delta Q_{i,j+1/2} + \mathcal{B}^{+}\Delta Q_{i,j-1/2}). \tag{3.19}$$

Relationship between flux differencing form and wave propagation

Although this wave propagation algorithm is different from standard flux differencing method, there is a relation between the two for hyperbolic conservation equations such as the SWE. In 1D the relation between numerically approximated flux $F_{i-1/2}$ and the waves is given by: $F_{i-1/2} = f(q_{i-1}) + \mathcal{A}^{-}\Delta Q_{i-1/2}$, $F_{i-1/2} = f(q_i) - \mathcal{A}^{+}\Delta Q_{i-1/2}$ [10], which gives us:

$$F_{i+1/2} - F_{i-1/2} = \mathcal{A}^{-}\Delta Q_{i+1/2} + \mathcal{A}^{+}\Delta Q_{i-1/2}.$$

In 2D SWE, we have two sets of fluxes that contribute to the update of cell (i, j), which we denote $F_{i+1/2,j}, F_{i-1/2,j}$ and $G_{i,j+1/2}, G_{i,j-1/2}$, and we have:

$$F_{i+1/2,j} - F_{i-1/2,j} = \mathcal{A}^{-}\Delta Q_{i+1/2,j} + \mathcal{A}^{+}\Delta Q_{i-1/2,j}$$
$$G_{i,j+1/2} - G_{i,j-1/2} = \mathcal{B}^{-}\Delta Q_{i,j+1/2} + \mathcal{B}^{+}\Delta Q_{i,j-1/2}.$$

Essentially, the wave propagation algorithm works because it is another way of approximating the numerical fluxes $F_{i+1/2,j}, G_{i,j+1/2}$ using spectral methods.

With this relation we can go back and forth from the flux difference form of the numerical update to the wave propagation form akin to what is done in [35]. We use the flux difference form to do conservation calculations later on in our h-box methods and use the wave propagation method for wave redistribution and implementation of examples. Also note that the terminology of “flux" is used for flux difference form but “fluctuation" is used for the wave propagation algorithm, but we will use them interchangeably.

The reason for using the wave propagation method is that it is a versatile method that can also be applied to nonconservative hyperbolic equations [10] and also an easily implemented method as long as the decomposing matrices are known and well-behaved.

Source terms: bathymetric variation

Another merit of the wave propagation method, other than the guarantee of conservation, is that presence of source terms is easily handled [31]. Suppose we have the hyperbolic equation

$$q_t + f(q)_x = \Psi(q),$$

as in Eq. (2.3). In our problems, we will have $\Psi(q) = [0, -ghb_x]$, where b is the bathymetry. All we need to modify in our wave propagation method in order to handle bathymetric source terms is by subtracting the discretization of the source term as follows:

$$\sum_{p=1}^{2} \beta_p r_p = f(Q_R) - f(Q_L) - \Delta x \Psi(Q_R, Q_L) \tag{3.20}$$

where $\Psi(Q_R, Q_L)$ is approximated by some averaging of the bathymetric variation (slope, e.g. $b_R - b_L/\Delta x$). This form of handling source terms can be hinted at by the following approximations:

$$\int_{x_{i-1/2}}^{x_{i+1/2}} q_t \approx (Q^{n+1} - Q^n)/\Delta t$$
$$\int_{x_{i-1/2}}^{x_{i+1/2}} f(q)_x - \Psi(q) \approx (f(Q_R) - f(Q_L)) - \Delta x \Psi(Q)$$

Then, we can write Eq. (2.3) as

$$q_t + (f(q)_x - \Psi(q)) = 0,$$

and treat $\int_{x_{i-1/2}}^{x_{i+1/2}} f(q)_x - \Psi(q)$ as our “flux" and use wave propagation to approximate this flux via linear, eigen-decomposition as shown in Eq. (3.20).

3.3.3 Second order

Second order finite volume methods are made from using a linear approximation on each cell. From the first order wave propagation method, however, we can easily move towards a second order method by careful use of the speeds (eigenvalues) and the eigenbasis vectors. The wave propagation algorithm thus provides an equivalent way of getting the higher order correction terms necessary for a second order numerical method without approximating solution slopes [36, 33].

This is achieved by adding to the first order fluctuations a correction term $\tilde{F}_{i\pm1/2,j}, \tilde{G}_{i,j\pm1/2}$:

$$\begin{aligned} Q_{i,j}^{n+1} = Q_{i,j}^{n} &- \frac{\Delta t}{\Delta x}(\mathcal{A}^{-}\Delta Q_{i+1/2,j} + \mathcal{A}^{+}\Delta Q_{i-1/2,j}) - \frac{\Delta t}{\Delta x}(\tilde{F}_{i+1/2,j} - \tilde{F}_{i-1/2,j}) \\ &- \frac{\Delta t}{\Delta y}(\mathcal{B}^{-}\Delta Q_{i,j+1/2} + \mathcal{B}^{+}\Delta Q_{i,j-1/2}) - \frac{\Delta t}{\Delta y}(\tilde{G}_{i,j+1/2} - \tilde{G}_{i,j-1/2}). \end{aligned} \tag{3.21}$$

The correction terms are then given by the following:

$$\tilde{F}_{i-1/2,j} = \frac{1}{2}\sum_{p=1}^{m}|(\lambda_A)_p|\big(1 - \frac{\Delta t}{\Delta x}|(\lambda_A)_p|\big)\tilde{\beta}_p\mathcal{A}_{(\lambda_A)_p}, \tag{3.22}$$

$$\tilde{G}_{i,j-1/2} = \frac{1}{2}\sum_{p=1}^{m}|(\lambda_B)_p|\big(1 - \frac{\Delta t}{\Delta x}|(\lambda_B)_p|\big)\tilde{\hat{\beta}}_p\mathcal{B}_{(\lambda_B)_p}, \tag{3.23}$$

where m is the dimension of the eigenspectrum, and the tilde over $\beta_p, \hat{\beta}_p$ is a limited version of the decomposition coefficient which we discuss in the next subsection. In a linear hyperbolic system $q_t + Lq_x = 0$, with L some constant matrix, this formulation Eq. (3.21) (without the limiting over the coefficients) can be shown to be equal to the Lax-Wendroff method in each dimension [10].

Wave limiting

Limiting of gradients or slopes in the linear approximations is necessary in the second order methods in order to prevent numerical instability such as oscillations, a common problem in higher accuracy finite volume methods. The tilde over the decomposition coefficients $\tilde{\beta}_p, \tilde{\hat{\beta}}_p$ denote limited decomposition coefficients used in wave propagation. Instead of limiting slopes, these work

by limiting the eigenbasis vectors decomposed from the flux vector differences from the Riemann problem. The limiting of the pth element of these coefficients are performed by comparing the the magnitude of pth eigenvectors arising from neighboring Riemann problems and applying the typical limiting functions to the ratio of magnitudes.

In symbols, we have the following: for a given Riemann problem between $Q_{i-1,j}$ and $Q_{i,j}$, let $W^p_{i-1/2,j} = \beta_p A_{(\lambda_A)_p}$ (or $\hat{\beta}_p B_{(\lambda_B)_p}$), or the weight vector of the flux difference. Then we have

$$\theta^p_{i-1/2,j} = \frac{W^p_{I-1/2,j} \cdot W^p_{i-1/2,j}}{\|W^p_{i-1/2,j}\|^2} \tag{3.24}$$

$$\tilde{\beta}_p = \phi(\theta^p_{i-1/2,j})\beta_p, \tag{3.25}$$

where $I = i - 1$ if $(\lambda_A)_p > 0$ and $I = i + 1$ if $(\lambda_A)_p < 0$, and $\phi : \mathbb{R} \to \mathbb{R}$ is a typical limiting function, such as minmod, MC, etc. The parameter $\theta^p_{i-1/2,j}$ is the component of the projection vector $\overrightarrow{\mathbf{proj}}_{W^p_{i-1/2,j}} W^p_{I-1/2,j}$ and measures how much the neighboring weight vector aligns with the weight vector at the interface. This roughly estimates how smooth the solution is at the interface and is the parameter used to determine how much limiting should be applied in case of a lower value, which shows steep discontinuity of solution.

In our problems, we apply the minmod limiter on all the horizontal and vertical grid edges. The minmod limiter is given by:

$$\phi_{minmod}(\theta) = \max(0, \min(1, \theta)). \tag{3.26}$$

Minmod limiters are symmetric: $\phi_{minmod}(1/\theta) = \phi_{minmod}(\theta)/\theta$, a desired property especially as our V-barrier problem is symmetric.

We note that the wave limiting method described above is suitable only for Cartesian edges, where the neighboring index I is defined. For the barrier cut edge, we will resort to using gradient limiters as we will need to employ limiters suitable for non-Cartesian cell edges. We discuss further in the cell merging method, where the second order method is developed.

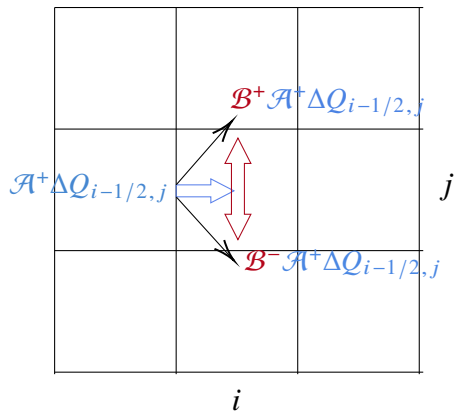


Figure 3.3: Transverse waves from edge $(i-1/2, j)$ affect cells $(i, j+1)$ and $(i, j-1)$.

3.3.4 Transverse wave propagation

Another way to achieve higher (than one) order accuracy in our numerical solutions is by introducing transverse solvers. The waves generated at the interface of two states as described above are directed *normal* to the interface, affecting those two states. They are thus called normal waves. However, each of these normal waves can further be decomposed in the direction parallel to the interface, i.e. transverse, and thus affecting the cells diagonally above and below the original cells (Fig. 3.3).

Essentially, this amounts to solving another linear equation as follows:

$$\mathcal{B}\tilde{\tilde{\beta}} = \mathcal{A}^+\Delta Q \tag{3.27}$$

$$\mathcal{B}^+\mathcal{A}^+\Delta Q = \sum_{\lambda_B>0} \tilde{\tilde{\beta}}\mathcal{B}_{\lambda_B} \tag{3.28}$$

$$\mathcal{B}^-\mathcal{A}^+\Delta Q = \sum_{\lambda_B<0} \tilde{\tilde{\beta}}\mathcal{B}_{\lambda_B}. \tag{3.29}$$

The vector $\mathcal{B}^\pm\mathcal{A}^+\Delta Q$ is called a transverse wave of $\mathcal{A}^+\Delta Q$. (We omit indices for simplicity in our notation.) The same can be carried out for the transverse wave of $\mathcal{A}^-\Delta Q$. Note that transverse solver does not decompose a difference of wave vectors, but a single wave vector, e.g. either $\mathcal{A}^+\Delta Q$ or $\mathcal{A}^-\Delta Q$. Note that in Fig. 3.3, our transverse waves would be used to update $Q_{i,j\pm1}$.

For our cut cell methods, we use the transverse calculations away from the cut interface. This

is because at the cut barrier interface, another method called wave redistribution for fluctuation calculation would need to be employed, which we discuss in the next chapter. Not employing the transverse solver at the barrier amounts to using wave redistribution only once. As we will see, however, using transverse solvers increases order of accuracy by approximately 0.5.

3.3.5 CFL Condition

Now with the notations of the finite volume method explained, we finally describe the Courant-Friedrichs-Lewy (CFL) condition as this is what necessitates the cut cell method on a restricted grid cell. The condition states that fluctuation cannot propagate into a cell more than its size. In the 1D problem, we have

$$\frac{\lambda \Delta t}{\Delta x} \leq 1, \tag{3.30}$$

where λ denotes the speed of a wave, Δt the time duration of the wave propagated, and Δx the width of a cell. This CFL number (the left hand term) can be seen to appear in the wave propagation form Eq. (3.15) to Eq. (3.19): the $\lambda_{A/B}$ in the summation terms will multiply with the $\Delta t/\Delta x$ term in front. Essentially, we want to only take fraction of each weight vector $W_p = \beta_p \mathcal{A}_p$ (or $\hat{\beta}_p \mathcal{B}_p$).

In 2D, we have

$$\frac{\lambda_A \Delta t}{\Delta x} + \frac{\lambda_B \Delta t}{\Delta y} \leq 1, \tag{3.31}$$

where λ_A, λ_B are speeds of the waves in x direction and y direction, respectively, and $\Delta t, \Delta x, \Delta y$ denote the time step and the width and height of a grid cell. Usually each term of the sum in Eq. (3.31) is limited to 0.5 in order to ensure the condition is met.

Chapter 4: Wave Redistribution

In order to compute the fluctuation at the barrier edge, we develop a special method called wave redistribution (WR). This method is repeatedly used in all our cut cell methods, and we devote this section to develop the method. The 2D methods to be presented in this chapter are original work of the author.

4.1 1D Wave redistribution

The method in 1D was originally developed in [37]. The initial idea for this method can be seen by considering the following. The arbitrary placement of a barrier on the grid (Fig. 2.1) creates small cells iw and iw+1 with sizes $\alpha > 0$ and $1 - \alpha$. In other words, when $\alpha = 0$, this means that the barrier is directly on a grid edge. WR was developed to handle the case $\alpha = 0$. However, as will be seen in our cut cell methods, WR is used not only to solve the case when $\alpha = 0$, but in general to set the flux at the barrier.

Since our barrier has no width, it cannot be treated as a cell with a bathymetric jump. The fluctuation between the cells on each side of the barrier must be calculated differently than that arising from a standard RP. It must mimic fluctuation that arise from hitting, reflecting, and/or overtopping a barrier with height ℓ. WR accomplishes this by redistributing waves arising from *two* ghost Riemann problems.

We will use the setup notation shown in Fig. 2.1 and denote cell averages of q as Q and bathymetric averages as B. The ghost problems of WR are set by introducing a middle, third ghost cell Q^* representing the barrier, in addition to the two cells Q_{iw}, Q_{iw+1} on either side of the barrier. We denote $Q^L_{iw+1/2} = Q_{iw}$ and $Q^R_{iw+1/2} = Q_{iw+1}$ to highlight the fact that we are calculating the fluctuation at the barrier edge.

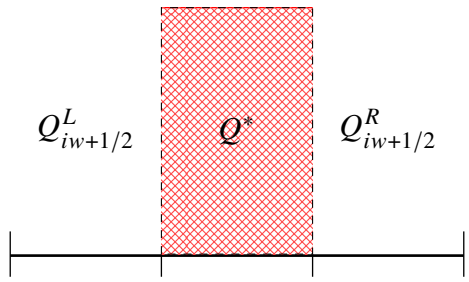


Figure 4.1: Introduction of ghost cell q^* in WR. The red hatched cell represents the barrier ghost cell.

The ghost cell average Q^* is set as a cell with bathymetry jump the size of the barrier, ℓ, and with $Q = 0$, i.e. dry state, to mimic a barrier with no water on top (left of Fig. 4.2):

$$Q^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$B^* = \min(B_{iw}, B_{iw+1}) + \ell.$$

There is a special case where Q^* is set differently, however. This is when water height on both sides of the barrier is higher than ℓ. This may be the result of there being simply more water on both sides of barrier, or the result of overtopping from both sides. In either case, we still set the bathymetry $B^* = \min(B_{iw}, B_{iw+1}) + \ell$, but we set Q^* as

$$Q^* = \begin{bmatrix} \min(h^L_{iw+1/2} - B^*, h^R_{iw+1/2} - B^*) \\ \min(hu^L_{iw+1/2}, hu^R_{iw+1/2}) \end{bmatrix}$$

where $Q^{L/R}_{iw+1/2} = [h^{L/R}_{iw+1/2}, hu^{L/R}_{iw+1/2}]$. The height being set here is the minimum of the "skimmed crop" of the two h-box heights off the barrier (Fig. 4.2), and the momentum is the minimum of the two h-box momentum. This is to ensure a continuous flow at the top of the barrier arising from the overtopping water. Furthermore, in the case where barrier is submerged and the water is at rest, not having this ghost water state would evidently create fictitious waves, as the left and right states would try to fill in the void.

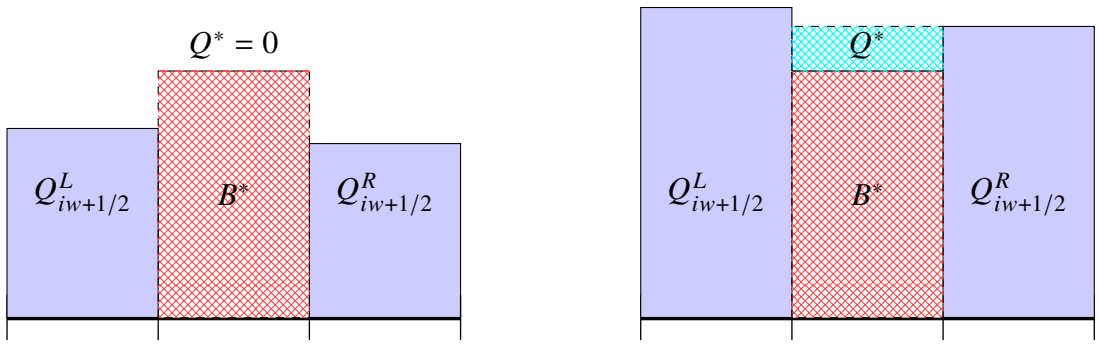


Figure 4.2: Cases of ghost cell q^* in WR. The red hatched cell here represents the ghost bathymetry and the aqua hatched cell represents the ghost water state.

Finally we discuss how to determine whether water overtops and how to redistribute the waves arising from the ghost problems. Whether water overtops from either side or not is calculated by solving a Riemann problem between $Q^{L/R}_{iw+1/2}$ with its inverted momentum state, $\mathcal{R}(Q^{L/R}_{iw+1/2}) = [h^{L/R}_{iw+1/2}, \text{-}(hu)^{L/R}_{iw+1/2}]$. The intermediate height h_m arising from this RP is then compared to the barrier height B^*, and if h_m is higher than B^*, we have overtopping, and if not, we have a reflection.

The two ghost RP's are solved by the wave propagation algorithm, which gives us:

$$\sum_{i=1}^{2} \beta_i \mathbf{r}_i = f(Q^*) - f(Q^L_{iw+1/2}) - \Psi_L$$
$$\sum_{i=1}^{2} \tilde{\beta}_i \tilde{\mathbf{r}}_i = f(Q^R_{iw+1/2}) - f(Q^*) - \Psi_R,$$

where $\{\mathbf{r}_i, \lambda_i\}$, $\{\tilde{\mathbf{r}}_i, s_i\}$ are the eigenvectors and their corresponding speeds arising from each RP and $\Psi_L = [0, -g(B^* - b_L)(h_L + h^*)/2]$ and $\Psi_R = [0, -g(b_R - B^*)(h^* + h_R)/2]$. Then we introduce a *new* set of eigenvectors $\{\hat{\mathbf{r}}_i\}_{i=1,2}$ and eigenvalues $\{\hat{s}_i\}_{i=1,2}$ which will propagate the mass and momentum transfer generated by both ghost RPs. For our new eigenvalues $\hat{s}_i$, we set them as averages of the ghost speeds, which gives us:

$$\hat{s}_i = \frac{1}{2}(\lambda_i + s_i) \tag{4.1}$$

$$\hat{\mathbf{r}}_i = \begin{bmatrix} 1 \\ \hat{s}_i \end{bmatrix}. \tag{4.2}$$

So redistribution solves:

$$\left[\mathbf{r}_1|\mathbf{r}_2|\tilde{\mathbf{r}}_1|\tilde{\mathbf{r}}_2\right]\tilde{\beta}_1 = \left[\hat{\mathbf{r}}_1|\hat{\mathbf{r}}_2\right]\hat{\beta}_2,$$

where the boldfaced $\mathbf{r}_i, \tilde{\mathbf{r}}_i, \hat{\mathbf{r}}_i$ are column eigenvectors, $\tilde{\beta}_1 = [\beta_1, \beta_2, \tilde{\beta}_1, \tilde{\beta}_2]$ from the two ghost RP decompositions, and $\hat{\beta}_2 \in \mathbb{R}^2$ are the new weights that the new eigenvectors $\hat{\mathbf{r}}_i$ will take. Then we let

$$\mathcal{A}^+\Delta Q_{iw+1/2} = \sum_{i:\ \hat{s}_i>0} \hat{\beta}_2^i \hat{\mathbf{r}}_i$$

and

$$\mathcal{A}^-\Delta Q_{iw+1/2} = \sum_{i:\ \hat{s}_i<0} \hat{\beta}_2^i \hat{\mathbf{r}}_i.$$

For $\hat{s}_i = 0$, we divide $\hat{\beta}_2^i \hat{\mathbf{r}}_i$ into half and distribute each to $\mathcal{A}^+\Delta Q_{iw+1/2}$ and $\mathcal{A}^-\Delta Q_{iw+1/2}$.

4.2 2D Wave Redistribution

In 2D, wave redistribution works in the same way as in 1D, with the exception of the additional transverse element now being carried by the second linearly degenerate eigenvector of the flux Jacobian $f'(q)$ and $g'(q)$ (Eq. (3.12)). As in 1D, a ghost state is introduced that represents the barrier. The determination of overtopping is done in the same way by using the *normal* momentum to the barrier. We present here therefore the ghost state and the new set of eigenvectors used to redistribute the waves.

Without loss of generality, consider WR between cells $q_{i,j}$ and $q_{i+1,j}$ where the barrier lies on their shared edge $x = x_{i+1/2}$ (see Fig. 4.3).

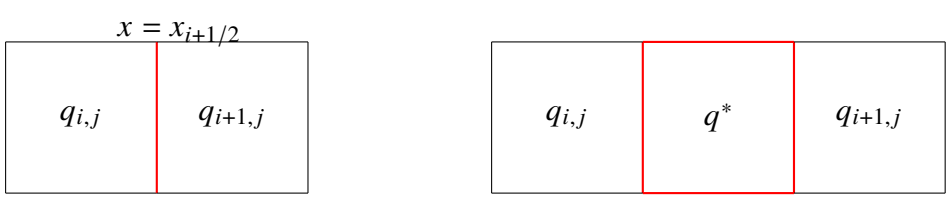


Figure 4.3: Example of WR in 2D between cells $q_{i,j}$ and $q_{i+1,j}$. On the left is shown barrier on edge $x = x_{i+1/2}$ and on the right is shown the ghost cell setup.

In all cases except when water overtops from both sides, we set $Q^* = 0$ and $B^* = \min(B_{i,j}, B_{i+1,j})$

$+\ell$, where ℓ is the length of the barrier. In case where water overtops from both sides, we set

$$Q^* = \begin{bmatrix} \min(h_{i,j} - B^*, h_{i+1,j} - B^*) \\ \min(hu_{i,j}, hu_{i+1,j}) \\ \min(hv_{i,j}, hv_{i+1,j}) \end{bmatrix}$$

$$B^* = \min(B_{i,j}, B_{i+1,j}) + \ell.$$

Finally, let $\{\lambda_i, \mathbf{r}_i\}$ and $\{s_i, \tilde{\mathbf{r}}_i\}$ be the x direction eigenpairs arising from the two RPs. This allows us to solve for the coefficients $\beta, \tilde{\beta}$:

$$[\mathbf{r}_1|\mathbf{r}_2|\mathbf{r}_3]\beta = f(Q^*) - f(Q_{i,j}) - \Psi_L \tag{4.3}$$

$$[\tilde{\mathbf{r}}_1|\tilde{\mathbf{r}}_2|\tilde{\mathbf{r}}_3]\tilde{\beta} = f(Q_{i+1,j}) - f(Q^*) - \Psi_R, \tag{4.4}$$

where $\Psi_{L/R}$ is the bathymetric variation between B^* and $B_{i/i+1,j}$.

We then set the eigenvectors for redistribution as: $\hat{\mathbf{r}}_1 = [1, \hat{s}_1, \hat{v}]$, $\hat{\mathbf{r}}_2 = [0, 0, 1]$, and $\hat{\mathbf{r}}_3 = [1, \hat{s}_3, \hat{v}]$, where $\hat{s}_i = \frac{1}{2}(s_i + \lambda_i)$ are the speeds and $\hat{v}$ is the Roe average of the transverse velocities. Then the following equation is solved for $\hat{\beta} \in \mathbb{R}^3$, which are the new redistributed wave coefficients:

$$[\mathbf{r}_i|\tilde{\mathbf{r}}_i](\beta : \tilde{\beta}) = [\hat{\mathbf{r}}_i]\hat{\beta},$$

where $\beta : \tilde{\beta}$ is the concatenated vector of eigendecomposition coefficients from the two RPs (total of 6 values) shown in the right subfigure of Fig. 4.3. Finally we collect the right going components with themselves and the left going ones with themselves:

$$\mathcal{A}^+\Delta Q_{i+1/2,j} = \sum_{p:\hat{s}_p>0} \hat{\beta}^p \hat{r}_p$$

$$\mathcal{A}^-\Delta Q_{i+1/2,j} = \sum_{p:\hat{s}_p<0} \hat{\beta}^p \hat{r}_p.$$

4.2.1 2D Rotated Wave redistribution

Now we discuss how to set the fluctuation at the barrier edge that is at an angle to the grid. Later in Section 7.1.2 we will describe how to do a second order correction at the barrier edge.

On a cut cell (i, j), we use $Q^h_{i,j}$, which we denote as the small cells' state averages as they are on either side ($h = L$ or U: see Fig. 4.4) of the barrier edge and rotate them as follows:

$$\breve{Q}^h_{i,j} = R_{i,j} Q^h_{i,j} = [H, H\breve{U}, H\breve{V}], \tag{4.5}$$

and

$$R_{i,j} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \hat{n}_1 & \hat{n}_2 \\ 0 & \hat{t}_1 & \hat{t}_2 \end{bmatrix}.$$

The rotation vectors are simply the orthonormal pair with respect to the barrier Fig. 4.4.

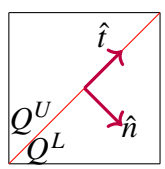


Figure 4.4: Rotation vectors used to rotate states: orthonormal pair.

Once we have rotated the states, we apply the WR algorithm as before with the ghost states being set in the same way. Namely, we have the following decompositions to perform:

$$\mathcal{A}_L \beta = f(\breve{Q}^L_{i,j}) - f(Q^*) - \Psi_L \tag{4.6}$$

$$\mathcal{A}_U \tilde{\beta} = f(Q^*) - f(\breve{Q}^U_{i,j}) - \Psi_U, \tag{4.7}$$

where $\mathcal{A}_L$ is the matrix similar to Eq. (3.12) for the Riemann problem between $\breve{Q}^L_{i,j}$ and Q^* with associated eigenvalues and eigenvectors $\{\lambda^i_L, \mathbf{r}^i_L\}$ and $\mathcal{A}_U$ is that for the Riemann problem between

$\breve{Q}^U_{i,j}$ and Q^* with eigenvalues and eigenvectors $\{\lambda^i_U, \mathbf{r}^i_U\}$. Without loss of generality, we choose the Jacobian form $f'(q)$ for $\mathcal{A}$, considering the rotated normal direction to be the (rotated) x direction. The terms Ψ_L and Ψ_U represent the source term vector as before.

Then wave redistribution sets a new set of eigenvalues and eigenvectors $\{s_i, \mathbf{r}_i\}$ as

$$s_i = \frac{1}{2}(\lambda^i_U + \lambda^i_L) \tag{4.8}$$

$$\hat{\mathbf{r}}_i = [1, s_i, \bar{v}] \text{ for } i = 1, 3 \tag{4.9}$$

$$\hat{\mathbf{r}}_2 = [0, 0, 1], \tag{4.10}$$

where $\bar{v} = \frac{1}{2}(\breve{V}^L + \breve{V}^U)$ (or the Roe average can be taken instead).

Finally, wave redistribution solves

$$[\mathcal{A}_L \| \mathcal{A}_U](\beta : \tilde{\beta}) = \mathbf{A}\hat{\beta}, \tag{4.11}$$

where $[\mathcal{A}_L \| \mathcal{A}_U]$ is the augmented matrix of two matrices $\mathcal{A}_L$ and $\mathcal{A}_U$, $\beta : \tilde{\beta}$ is the augmented vector of coefficient vectors from the two RPs, and $\mathbf{A}$ is the matrix $[\hat{\mathbf{r}}_1, \hat{\mathbf{r}}_2, \hat{\mathbf{r}}_3]$. The coefficient vector $\tilde{\beta}$ is the unknown to be solved for and once solved, we have for our redistributed waves at the barrier edge,

$$\mathcal{A}^+ \Delta\breve{Q}_{i,j} = \sum_{p:s_p>0} \hat{\beta}_p \hat{\mathbf{r}}_p \tag{4.12}$$

$$\mathcal{A}^- \Delta\breve{Q}_{i,j} = \sum_{p:s_p<0} \hat{\beta}_p \hat{\mathbf{r}}_p. \tag{4.13}$$

Once we have computed the positive and negative fluctuations arising from the rotated Riemann problem, we follow the algorithm in [27] and rotate them back into the original coordinate directions by the following linear transformation:

$$\mathcal{A}^\pm \Delta Q_{i,j} = R^T_{i,j} \mathcal{A}^\pm \Delta\breve{Q}_{i,j}. \tag{4.14}$$

Chapter 5: H-box Methods

The h-box method was originally developed in [8]. Like most cut cell methods, this method was first applied to solid embedded boundary problems and achieves second order accuracy at the boundary as well as in the interior for advection problems [7]. It works by enlarging the domain of dependence in calculation of fluxes by creating on either side of a cut cell edge a “box" of length h, which is either Δx or some specially derived length dependent on the direction of flow.

We attempt to solve our problem using h-box methods. The differences between this work and that in [7] and other embedded boundary grid problems are (1) the extra flux condition at the boundary that allows for mass transfer across it and (2) the relative simplicity of the proposed method. The proposed method provides a simpler way to solve the embedded grid and small cell problem than previously developed methods such as the large time stepping method, which includes multi-step updates, or the original h-box method, which involves the generation of many h-boxes at the edges of small cells [7, 17, 29].

5.1 Prior Work

5.1.1 Large Time Stepping Method in 1D

A barrier problem in 1D with cut cell has been solved [37] using large time stepping method (LTS) [17]. This is an h-box type method, and we present the work briefly here to give background of what has been done and what added benefit we provide with our work, namely the simplicity of our algorithm.

LTS is a very complicated algorithm, involving multi-step updates. The idea is to take small incremental time steps Δt on each side of the barrier when updating the cut cells (see Fig. 5.1), according to the CFL condition $\frac{s\Delta t}{\Delta x_i} < 1$, where s is the speed of a wave and Δx_i the size of cell i.

Since one side of the barrier is larger than the other, this means that CFL violation does occur on the smaller cell. However, the fluxes at cut cell edges are used to update an *extended* average off the barrier cut called h-box is to avoid a CFL violation. This h-box is also used to find the flux or fluctuation at the barrier. Essentially the LTS method handles the small cell problem by tracking the fluxes' movement in incremental time periods. This is done until the incremental time periods sum up to a full time step Δt and the incrementally updated h-box averages are used to produce a full time step update, which was not allowed for the small cells.

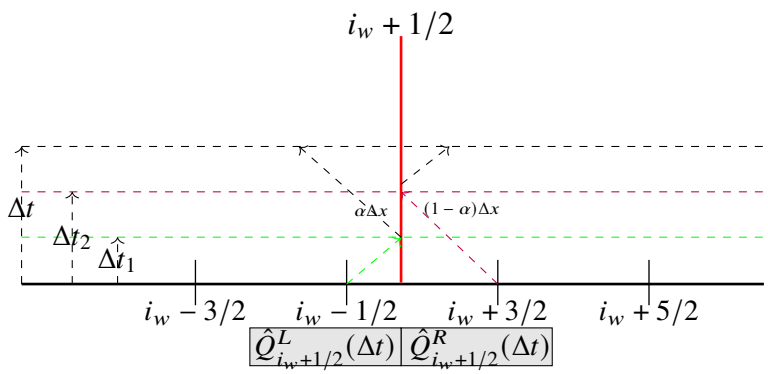


Figure 5.1: LTS Schematics: two small updates are made before the full time step update. The green dashed line corresponds to $\mathcal{A}^+\Delta Q_{i_w-1/2}$, the two black lines $\mathcal{A}^\pm\Delta Q_{i_w+1/2}$ and the magenta line $\mathcal{A}^-\Delta Q_{i_w+3/2}$.

We refer the reader to [37] for details but offer a two-increment update for the small cell i_w. Analogous update must be done for cell i_w+1. Important features here are the increment Δt_1, which is the maximum time step allowed for small cell i_w and increment Δt_2, which is the maximum time step allowed for small cell i_w+1. Once each time increment Δt is taken to update the cells, a new h-box average update $\hat{Q}^{L/R}_{i_w+1/2}(\Delta t)$ is calculated to update the fluctuation at the wall. The superscript Δt_i in Eq. (5.1) indicates the updated speed or wave after that increment, and the absence of the superscript indicates the speed and wave at time t^n. Also, λ_i is the left or right ($i = 1, 2$) speed of

the waves at the wall:

$$
\begin{aligned}
Q_{i_w}^{n+1} = Q_{i_w}^{n} &- \frac{\Delta t_1}{\alpha \Delta x} \mathcal{A}^{+} \Delta Q_{i_w - 1/2} \\
&- \frac{\Delta t}{\alpha \Delta x} \left(\frac{\min(-\lambda_1 \Delta t, \alpha \Delta x)}{-\lambda_1 \Delta t} \mathcal{A}^{-} \Delta Q_{i_w + 1/2} \right) \\
&- \frac{\Delta t - \Delta t_1}{\alpha \Delta x} \left(\frac{\min(-\lambda_1^{\Delta t_1} (\Delta t - \Delta t_1), \alpha \Delta x)}{-\lambda_1^{\Delta t_1} (\Delta t - \Delta t_1)} \mathcal{A}^{-} \Delta Q_{i_w + 1/2}^{\Delta t_1} \right) \\
&- \frac{\Delta t - \Delta t_2}{\alpha \Delta x} \left(\frac{\min(-\lambda_1^{\Delta t_2} (\Delta t - \Delta t_2), \alpha \Delta x)}{-\lambda_1^{\Delta t_2} (\Delta t - \Delta t_2)} \mathcal{A}^{-} \Delta Q_{i_w + 1/2}^{\Delta t_2} \right).
\end{aligned}
\tag{5.1}
$$

The first fractional term $\frac{\Delta t_1}{\alpha \Delta x} \mathcal{A}^{+} \Delta Q_{i_w - 1/2}$ comes from the right fluctuation at edge $i_w - 1/2$; the second fractional term is from left wave of the initial WR at the barrier, the third term is from left wave of the second WR (since the right wave from $i_w - 1/2$ hits the barrier); and the last term arises from left wave of third WR (since the left wave form $i_w + 3/2$ hits the barrier).

5.1.2 H-box method in 2D

Finally we conclude this section with a brief introduction to the original h-box method in 2D as this will later contrast to our method in 2D. For the method in 1D we refer readers to [7]. The method is best described with the help of a diagram, which shows wall-like embedded boundaries (Figs. 5.2 and 5.3).

The h-box length can be set as the regular grid size $h = \Delta x = \Delta y$ (assuming uniform grid for simplicity) and the values that each h-box takes is the volume weighted average of the cells it covers, rotated with respect to boundary's normal and transverse direction. The normal h-box cells that go into the boundary (colored black in Fig. 5.2) are set by negating the rotated normal velocity of the small cell (colored yellow in same figure), and the overall value in the whole h-box cell is again weighted by volume [4] with this negated average. Once the Riemann problems in both normal and transverse directions are computed, the resulting fluxes are combined and appropriately rotated back in the original grid directions (for details refer to [7]), and the small cell is updated using those fluxes.

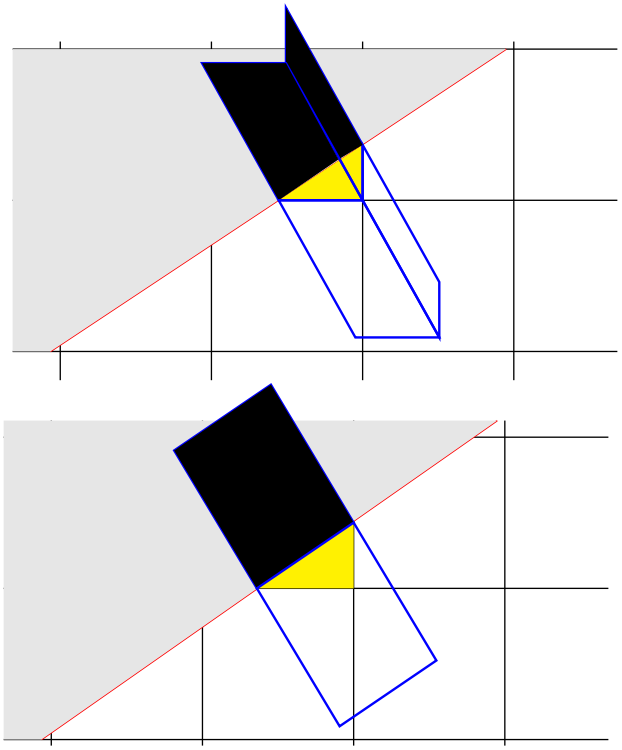


Figure 5.2: Diagram showing example of h-boxes used in setting normal (to boundary) direction flux/fluctuation at edges of yellow cell. The setup is a 2D grid with a wall-like embedded boundary. The blacked out portions of the h-boxes are the ghost cell portions where the normal velocity of the yellow cell with respect to the wall is negated. The diagram above shows the normal h-boxes associated with the non-barrier edges, while the one below those for the barrier edge.

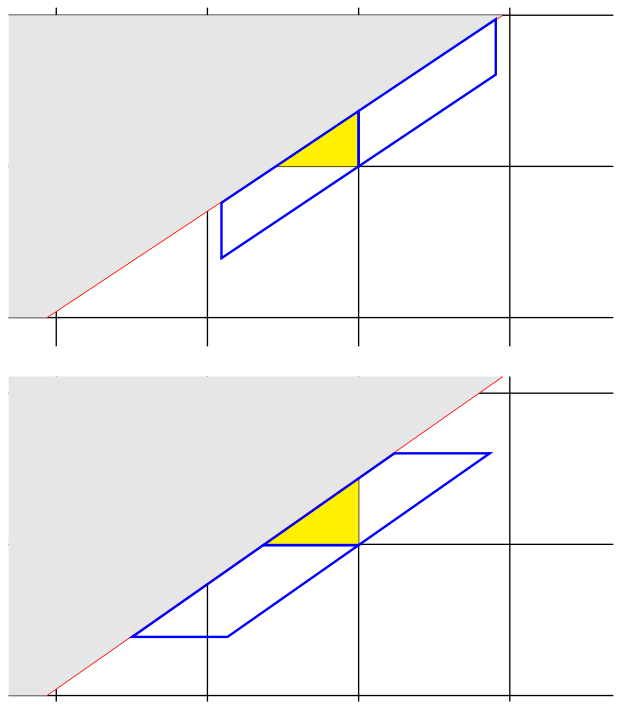


Figure 5.3: Diagram showing example of h-boxes used in setting transverse (to boundary) direction flux/fluctuation at edges of yellow cell. Two pairs of h-boxes are shown here on the horizontal and vertical grid edge.

The stability of the method is guaranteed by the overlap of the h-boxes (consider the three pairs of normal h-boxes in Fig. 5.2, for example), which leads to a *cancellation property* in a cell i:

$$\oint_{S_i} (f(\tilde{q}) + g(\tilde{q})) \cdot n \, dS = O(|V_i|)$$

$$O(|V_i|) << 1, \tag{5.2}$$

where $\tilde{q}$ represents the h-box state averages.

This means that the flux differences become on the order of the size of the small cell, which allows for a controlled update when taking volume average over the net fluxes. The flux differences are as small as the size of the cell due to the Lipschitz continuity of the numerical fluxes:

$$|F(Q_i, Q_j) - F(Q_j, Q_k)| < K| \max(|Q_i - Q_j|, |Q_j - Q_k|),$$

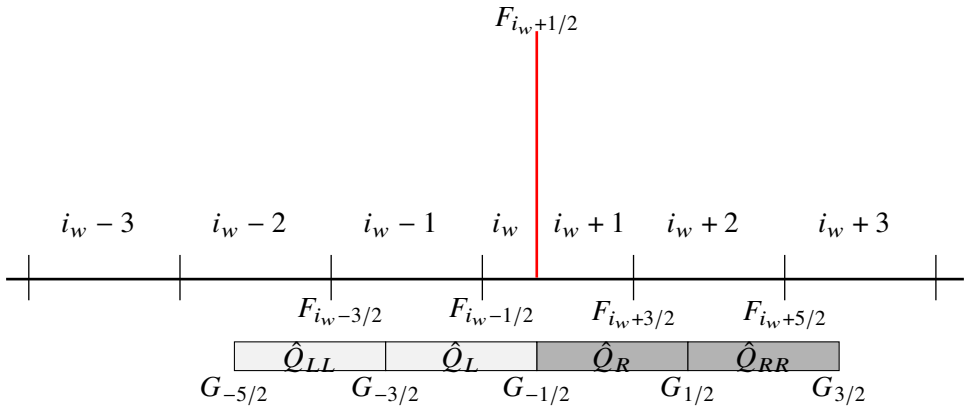


Figure 5.4: The double h-boxes off the barrier in 1D setup. The F_i denote the fluxes at the physical grid edges, while G_i denote the fluxes between the h-boxes.

where K is some constant and Q_i, Q_j, Q_k are state averages for neighboring cells. Since the h-box averages are overlapping, the term on the right hand side will be small. One can now see why enlarging the domain of dependence by using h-boxes is useful in the calculation of the fluxes at the edges.

We show the original 2D h-boxes and mention these properties for comparison with our new method. We highlight the fact that for the original h-box method, there are numerous h-boxes (e.g. Figs. 5.2 and 5.3) that need to be calculated, but as will be seen, our new method only calculates the normal h-box at the boundary edge and instead creates a grid along the boundary. Also, because of this absence of creating h-boxes on every edge, our new method will not explicitly exhibit cancellation property but still use the enlarging of domain of dependence for stability.

5.2 1D problem with a double h-box method

Now we present the new method in solving the stated 1D problem, in contrast to the LTS method. Instead of tracking the waves, we simply introduce two h-boxes off the barrier in both directions (see Fig. 5.4).

The general idea of the double h-box method is that we extend enough to cover the cut cells and update the h-boxes and the corresponding physical grid cells according to the h-box fluctuations. The LTS method is accurate but cumbersome, and the double h-box method tries to find the balance

between accuracy and complexity. Namely, the double h-box method effectively pushes away the cut cell edges from the barrier interface. Consider $F_{i_w-1/2}$ in Fig. 5.4 for example. The right wave from this edge will reach the barrier edge in a short time Δt_s. However, the wave from h-box edge $G_{-3/2}$ approximates this wave further away and allows for an update using regular time step Δt.

The h-box values $\hat{Q}$ are again simply volume weighted averages of the underlying cells at time t^n and update the physical grid cells proportionally by volume:

$$\begin{aligned}
\hat{Q}_{LL} &= (1-\alpha)Q_{i_w-2} + \alpha Q_{i_w-1} \\
\hat{Q}_{L} &= (1-\alpha)Q_{i_w-1} + \alpha Q_{i_w} \\
\hat{Q}_{R} &= (1-\alpha)Q_{i_w+1} + \alpha Q_{i_w+2} \\
\hat{Q}_{RR} &= (1-\alpha)Q_{i_w+2} + \alpha Q_{i_w+3}.
\end{aligned}$$

Once h-box averages are set, standard RPs are solved between $\hat{Q}_L$ and $\hat{Q}_{LL}$ and between $\hat{Q}_R$ and $\hat{Q}_{RR}$ to set $G_{-3/2}, G_{1/2}$, respectively. Between $\hat{Q}_L$ and $\hat{Q}_R$, WR is used to obtain right and left fluctuations, which set $G_{-1/2}$ and also $F_{i_w+1/2}$. The only remaining fluxes to be set are at the leftmost and rightmost edges of the h-boxes denoted $G_{-5/2}, G_{3/2}$ in Fig. 5.4.

To determine these outermost fluxes, we use the conservation principle. The conservation principle is that we need to have the same mass at time t^{n+1} via the regular flux differencing update as at time t^{n+1} via the h-box update, given a large enough x-axis window (large enough range disallowing outflow of mass under CFL condition).

The updated mass at time t^{n+1} of the surrounding cells near the barrier using standard flux differencing is given by:

$$\sum_{i=i_w-3}^{i_w+4} \alpha_i Q_i^{n+1} = \sum_{i=i_w-3}^{i_w+4} \alpha_i Q_i^{n} - \frac{\Delta t}{\Delta x_i}\left(F_{i+1/2} - F_{i-1/2}\right), \tag{5.3}$$

where $\alpha_{i_w} = \alpha$, $\alpha_{i_w+1} = 1-\alpha$, and $\alpha_{i\neq i_w, i_w+1} = 1$, and $\Delta x_{i_w} = \alpha\Delta x$, $\Delta x_{i_w+1} = (1-\alpha)\Delta x$ and $\Delta x_{i\neq i_w, i_w+1} = \Delta x$ according to Fig. 5.4.

On the other hand, the h-box update of the small cells and neighboring cells is given by:

$$Q_{i_w-1}^{n+1} = \alpha \hat{Q}_{LL}^{n+1} + (1-\alpha)\hat{Q}_{L}^{n+1} \tag{5.4}$$

$$Q_{i_w}^{n+1} = \hat{Q}_{L}^{n+1} \tag{5.5}$$

$$Q_{i_w+1}^{n+1} = \hat{Q}_{R}^{n+1} \tag{5.6}$$

$$Q_{i_w+2}^{n+1} = (1-\alpha)\hat{Q}_{R}^{n+1} + \alpha \hat{Q}_{RR}^{n+1} \tag{5.7}$$

$$Q_{i_w-2}^{n+1} = \alpha \overline{Q}_{i_w-2}^{n+1} + (1-\alpha)\hat{Q}_{LL}^{n+1} \tag{5.8}$$

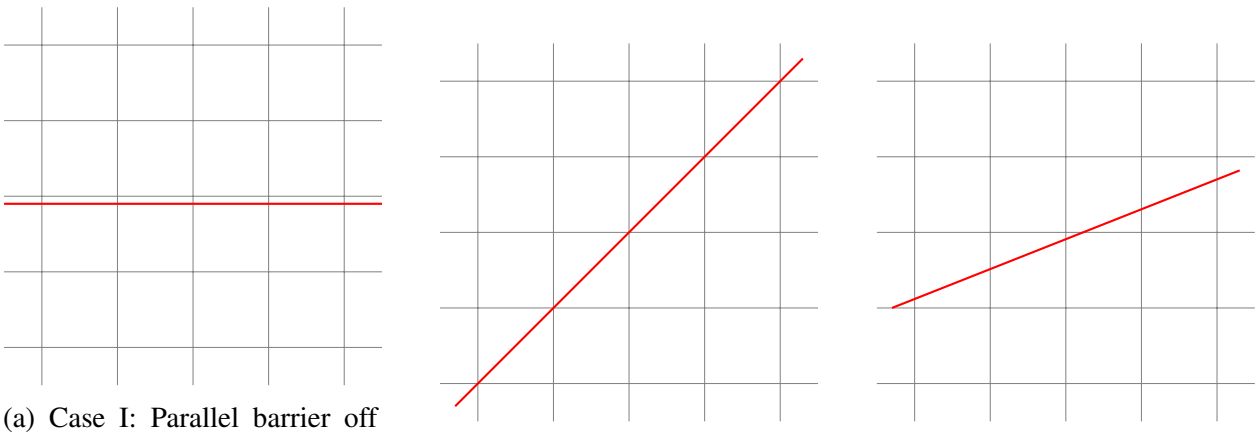
$$Q_{i_w+3}^{n+1} = (1-\alpha)\overline{Q}_{i_w+3}^{n+1} + \alpha \hat{Q}_{RR}^{n+1}, \tag{5.9}$$

where $\overline{Q}_i^{n+1}$ indicates the regular update of that cell. Given these updates, the updated mass at time t^{n+1} of the surrounding cells is given by:

$$\begin{aligned}
\sum_{i=i_w-3}^{i_w+4} \alpha_i Q_i^{n+1} &= Q_{i_w-3}^{n} - \frac{\Delta t}{\Delta x}\left(F_{i_w-5/2} - F_{i-7/2}\right) + Q_{i_w+4}^{n} - \frac{\Delta t}{\Delta x}\left(F_{i_w+9/2} - F_{i_w+7/2}\right) \\
&+ \alpha\left(Q_{i_w-2}^{n} - \frac{\Delta t}{\Delta x}\left(F_{i_w-3/2} - F_{i_w-5/2}\right)\right) \\
&+ (1-\alpha)\left(Q_{i_w+3}^{n} - \frac{\Delta t}{\Delta x}\left(F_{i_w+7/2} - F_{i_w+5/2}\right)\right) \\
&+ \hat{Q}_{LL}^{n+1} + \hat{Q}_{L}^{n+1} + \hat{Q}_{R}^{n+1} + \hat{Q}_{RR}^{n+1} \\
&= \overline{Q}_{i_w-3}^{n+1} + \overline{Q}_{i_w+4}^{n+1} + \alpha \overline{Q}_{i_w-2}^{n+1} + (1-\alpha)\overline{Q}_{i_w+3}^{n+1} \\
&+ \hat{Q}_{LL}^{n} - \frac{\Delta t}{\Delta x}\left(G_{-3/2} - G_{-5/2}\right) + \hat{Q}_{L}^{n} - \frac{\Delta t}{\Delta x}\left(G_{-1/2} - G_{-3/2}\right) \\
&+ \hat{Q}_{R}^{n} - \frac{\Delta t}{\Delta x}\left(G_{1/2} - G_{-1/2}\right) + \hat{Q}_{RR}^{n} - \frac{\Delta t}{\Delta x}\left(G_{3/2} - G_{1/2}\right),
\end{aligned} \tag{5.10}$$

where $G_{i-1/2}$ represents the flux between h-boxes, as also seen in Fig. 5.4.

Now equating the two expressions in Eq. (5.3) and Eq. (5.10) gives us what $G_{3/2}$ and $G_{-5/2}$



(a) Case I: Parallel barrier off edge

(b) Case II: diagonal barrier

(c) Case III: 20° barrier

Figure 5.5: The three examples of the 2D barrier problem being studied.

should be:

$$G_{3/2} = \alpha F_{i_w+7/2} + (1-\alpha) F_{i_w+5/2}$$
$$G_{-5/2} = (1-\alpha) F_{i_w-5/2} + \alpha F_{i_w-3/2}. \tag{5.11}$$

Since we use wave propagation algorithm in solving the SWE, we translate these fluxes in wave form in the actual numerical solutions. We provide the 1D numerical results in section Section 5.4.

5.3 2D problem with the h-box grid

We develop three h-box type methods in the 2D case, none of which uses LTS or the many sided h-boxes of the original method. First we extend the double h-box method to the case that barrier's orientation is parallel to grid. Second, we develop a single h-box layer method in the case that barrier is at 45° to the grid. Finally, we develop also a single h-box layer method in the case that barrier is at 20° to the grid, which can theoretically be extended to other angles (see Figs. 5.5a to 5.5c for the setup). Note that these are instances of the first type of barrier problem as described in Fig. 2.2a.

5.3.1 2D Problem I: Parallel barrier

The simplest extension of the 1D problem to 2D is when we have a straight barrier that cuts through either a row or column of cells as in Fig. 5.5a. The small cells are all in one row or column, and the ratio of their areas to regular cell area are also α and $1 - \alpha$, where α is the proportion of the smaller cut or equivalently, the distance from the nearest edge.

Because of the added dimension, now we need double h-boxes off the barrier on every column or row, creating a grid of h-boxes. Without loss of generality, we will only consider a horizontal barrier that cuts through a row of cells. Then we create a double h-box grid as in Fig. 5.6. Note that the figure shows a cut-out of a bigger domain, so the the h-box grid can continue in the lateral direction as long as the physical domain extends.

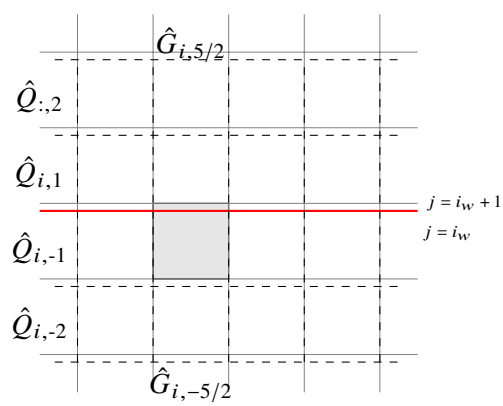


Figure 5.6: Double h-box grid for grid aligned barrier problem. Each γ-row of h-boxes are denoted by $\hat{Q}_{i,\gamma}$. The jth index of the small cells are i_w and $i_w + 1$.

Before we discuss how to update these h-boxes, we first remark the difference between this proposed method and the original h-box method. If the same principles of the original 2D h-box method were to be applied to our problem, we would have to calculate h-boxes off every small cell edge.

In our problem, however, this includes calculating the value of an h-box that lies on top of the barrier, namely the h-box off edge $y = y_{i_w - 1/2}$ that coincides with the sliced cell (shaded in Fig. 5.6). The value for this crossed over h-box is ambiguous, so our method bypasses the need to set a crossed over value on such h-box and is also the reason why the double h-box method was

developed for our model problem. To compensate for this absence of h-box calculation and loss of flux at an edge, we add the second h-box. As we will see, however, in the angled barrier case, we cannot use the two full h-box layers as the conservation calculations quickly get complicated.

The h-box values in this grid method are calculated as they were in the 1D case, by weighting the covered cells by volume. We number the h-box rows underneath the barrier with γ = -1, -2, and the ones above the barrier with $\gamma = 1, 2$. Then we have:

$$\begin{aligned}
\hat{Q}_{i,-2} &= \alpha Q_{i,i_w-2} + (1-\alpha) Q_{i,i_w-1} \\
\hat{Q}_{i,-1} &= \alpha Q_{i,i_w-1} + (1-\alpha) Q_{i,i_w} \\
\hat{Q}_{i,1} &= \alpha Q_{i,i_w+1} + (1-\alpha) Q_{i,i_w+2} \\
\hat{Q}_{i,2} &= \alpha Q_{i,i_w+2} + (1-\alpha) Q_{i,i_w+3}.
\end{aligned} \tag{5.12}$$

Once the values are set at time t^n, WR is used between h-boxes $\hat{Q}_{i,-1}$ and $\hat{Q}_{i,1}$ for each column i. Then standard RPs are solved between the h-boxes $\hat{Q}_{i,-2}$ and $\hat{Q}_{i,-1}$, and between $\hat{Q}_{i,1}$ and $\hat{Q}_{i,2}$ for each column i. Finally, the same formula as in the 1D double h-box method is used to set the fluctuation at the outermost edges.

Let $\hat{G}_{i,5/2}$ denote the fluxes at the top edge of h-boxes $\hat{Q}_{i,2}$, and let $\hat{G}_{i,-5/2}$ denote the fluxes at the bottom edge of h-boxes $\hat{Q}_{i,-2}$. Also, let $G_{i,j-1/2}$ denote the flux in y-direction at the bottom edge of grid cell $C_{i,j}$. Then we have:

$$\begin{aligned}
\hat{G}_{i,5/2} &= \alpha G_{i,i_w+7/2} + (1-\alpha) G_{i,i_w+5/2} \\
\hat{G}_{i,-5/2} &= (1-\alpha) G_{i,i_w-5/2} + \alpha G_{i,i_w-3/2}.
\end{aligned} \tag{5.13}$$

As for the flux in the x (transverse) direction $\hat{F}_{i\pm1/2,\gamma}$ between the h-boxes, we resort to the appropriately scaled flux that arises from the small cells, instead of computing them out of the h-box cells. This is because in the transverse direction, small cells cause no stability problems in this special case since flow is in the x-direction where the cell width is always Δx as also discussed

in [38]. In equation form, we have that for each h-box cell i in row $\gamma = -2, -1, 1, 2$:

$$\hat{Q}^{n+1}_{i,\gamma} = \hat{Q}^{n}_{i,\gamma} - \frac{\Delta t}{\Delta x}\left(\hat{F}_{i+1/2,\gamma} - \hat{F}_{i-1/2,\gamma}\right) - \frac{\Delta t}{\Delta y}\left(\hat{G}_{i,\gamma+1/2} - \hat{G}_{i,\gamma-1/2}\right), \tag{5.14}$$

where

$$\begin{aligned}
\hat{F}_{i+1/2,-1} &= (1-\alpha)F_{i+1/2,i_w} + \alpha F_{i+1/2,i_w-1} \\
\hat{F}_{i+1/2,-2} &= (1-\alpha)F_{i+1/2,i_w-1} + \alpha F_{i+1/2,i_w-2} \\
\hat{F}_{i+1/2,1} &= \alpha F_{i+1/2,i_w+1} + (1-\alpha)F_{i+1/2,i_w+2} \\
\hat{F}_{i+1/2,2} &= \alpha F_{i+1/2,i_w+2} + (1-\alpha)F_{i+1/2,i_w+3}.
\end{aligned} \tag{5.15}$$

Once the h-box cells are updated, the small cells update are by volume and as follows:

$$\begin{aligned}
Q^{n+1}_{i,i_w} &= \hat{Q}_{i,-1} \\
Q^{n+1}_{i,i_w+1} &= \hat{Q}_{i,1} \\
Q^{n+1}_{i,i_w-1} &= \alpha\hat{Q}_{i,-1} + (1-\alpha)\hat{Q}_{i,-2} \\
Q^{n+1}_{i,i_w+2} &= (1-\alpha)\hat{Q}_{i,1} + \alpha\hat{Q}_{i,2} \\
Q^{n+1}_{i,i_w-2} &= (1-\alpha)\overline{Q}^{n+1}_{i,i_w-2} + \alpha\hat{Q}_{i,-2} \\
Q^{n+1}_{i,i_w+3} &= (1-\alpha)\hat{Q}_{i,2} + \alpha\overline{Q}^{n+1}_{i,i_w+3},
\end{aligned} \tag{5.16}$$

where $\overline{Q}^{n+1}_{i,j}$ represents the standard regular update, i.e.

$$\overline{Q}^{n+1}_{i,j} = Q^{n}_{i,j} - \frac{\Delta t}{\Delta x}\left(F_{i+1/2,j} - F_{i-1/2,j}\right) - \frac{\Delta t}{\Delta y}\left(G_{i,j+1/2} - G_{i,j-1/2}\right). \tag{5.17}$$

When $\Delta x = \Delta y$, we can also simplify the notation for the regular update as:

$$\overline{Q}^{n+1}_{i,j} = Q^{n}_{i,j} - k\Delta Q^{n+1}_{i,j},$$

where $k = \frac{\Delta t}{\Delta x}$ and $\Delta Q_{i,j}^{n+1} = (F_{i+1/2,j} - F_{i-1/2,j}) + (G_{i,j+1/2} - G_{i,j-1/2})$ to isolate the mass change from time step n to $n+1$. We will use this notation in the next section. We also remark that in the actual implementation, the normal fluctuations and the transverse fluctuations are calculated separately, in a dimensional split way akin to work in [39].

5.3.2 2D Problem II: Diagonal barrier

The second case we study is the barrier that is at 45° to the grid (Fig. 5.5b). In this case the small cells all have half the size of regular cells and occur along the diagonal of the grid. Although this example is the simplest of the angled barrier problem, some new complexities arise.

Rotation of states

First, the obvious complexity is the need to rotate the states with respect to the barrier to calculate the flux between the two small cells on either side of the barrier. The water momentum hu, hv are given in the Cartesian coordinate directions. To rotate these in directions $\hat{n}, \hat{n}^{\perp}$, we can apply the rotation matrix $R_{\hat{n}}$ shown below. Let $Q_{i,j}$ be the Cartesian coordinate aligned state. Then we have for the rotated state $\breve{Q}_{i,j}$ [10]:

$$\breve{Q}_{i,j} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \hat{n}_x & \hat{n}_y \\ 0 & -\hat{n}_y & \hat{n}_x \end{bmatrix} Q_{i,j}. \tag{5.18}$$

To rotate back to the Cartesian coordinate, we apply the inverse:

$$Q_{i,j} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \hat{n}_x & -\hat{n}_y \\ 0 & \hat{n}_y & \hat{n}_x \end{bmatrix} \breve{Q}_{i,j}. \tag{5.19}$$

Now to determine these rotated grid directions $\hat{n}, \hat{n}^{\perp}$, we choose the unit normal vector to the diagonal barrier, $\hat{n} = (\pm\frac{1}{\sqrt{2}}, \mp\frac{1}{\sqrt{2}})$ and its orthogonal $\hat{n}^{\perp} = (\mp\frac{1}{\sqrt{2}}, \pm\frac{1}{\sqrt{2}})$. The reason for the

alternating signs is that the rotated grid directions must correspond to the water's interaction with the barrier. In the case that water bounces off the barrier and does not overtop, the normal direction vector to the barrier must point in the direction of the *reflection*. In the case that water overtops the barrier, the normal direction vector must point towards the direction of the *overtopping*, while the corresponding transverse vector also switches sign. In Fig. 5.7, we show the normal and transverse

(a) No overtopping case (b) Overtopping case

Figure 5.7: Determining the direction vectors of rotation.

direction vectors used in rotating the *lower* half cell, assuming the water moves from right-bottom to left-top and either reflects (a) or overtops (b), as the blue dotted line indicates.

Motivation for monolayer h-box grid setup

Since these direction vectors will be used to rotate the h-boxes covering the small cells, we now discuss how the h-box grid is set up. In the previous case, constructing the h-box grid and determining the fluctuations at the outer edges of the h-boxes were straightforward extensions of the 1D case. If we apply the same double h-boxes grid to this diagonal barrier, however, the conservation calculation required to determine the outer edge fluxes (c.f. Eq. (5.11) for 1D) becomes much more complicated, which is another added complexity in the angled case. However, if we only form one layer of h-boxes on either side of the barrier, the calculation is more manageable with the method still producing reasonable results. We first draw the monolayer grid in Fig. 5.8, and show the calculations.

In Fig. 5.8, we consider a grid with only two cells in both the x direction and y direction, with two ghost cells all around, to also show what must be done at the boundary. Note that the barrier only extends to first ghost cell outside the inner domain, since otherwise the h-boxes will

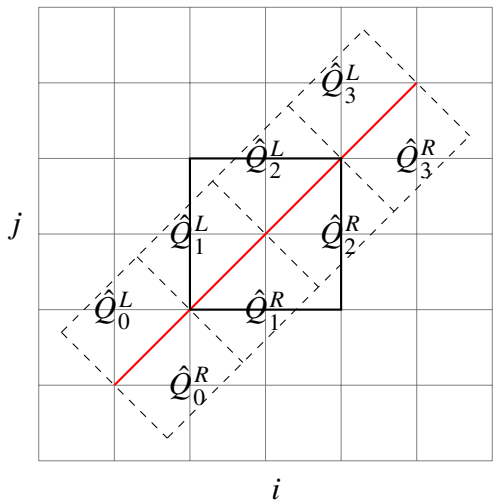


Figure 5.8: Monolayer of h-boxes for diagonal barrier problem. Example with a small grid (2×2) setup, with the outer layer being ghost cells of thickness 2.

lie outside the whole domain if the barrier were to extend all the way to the second layer of ghost cells. As can be seen in the diagram, the proportion of cells that comprise the h-boxes are not as simple as in case I, and now additional geometrical calculations are required. However, in this diagonal case, there is a repeated pattern for each h-box and the area proportion of each cell can be easily calculated (see Fig. 5.9). This gives us:

$$\hat{Q}_{i,i} = \frac{1}{\sqrt{2}\Delta x \Delta y}\left(a_1 Q_{i,i} + 2a_2 Q_{i+1,i} + a_3 Q_{i+1,i-1}\right). \tag{5.20}$$

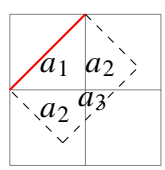


Figure 5.9: Area weights of cells that make up each h-box: $\{a_1 = \frac{1}{2}, a_3 = (1-\frac{\sqrt{2}}{2})^2, a_2 = \frac{\sqrt{2}-a_1-a_3}{2}\}$. Note that the area of the h-box is $\sqrt{2}\Delta x \Delta y$, and the weights will be effectively normalized by $\sqrt{2}$.

Fluctuation calculations

Once these h-box cells are rotated according to Eq. (5.18), two sets of fluctuations calculations are performed, namely the WR at the barrier between the h-boxes on either side of the barrier, ($\hat{Q}_i^L$ and $\hat{Q}_i^R$) and the transverse fluctuation calculations in between the h-boxes on one side of the barrier ($\hat{Q}_{i-1}^M$ and $\hat{Q}_i^M$, where $M = L$ or R). Since SWE is rotationally invariant, the fluctuation calculations between rotated states are exactly the same as that between two grid-aligned states. After the h-box cells are updated according to the fluctuations, they are rotated back into the Cartesian coordinate grid for the small cells and neighboring cells' final update.

Outer fluxes

To update the h-boxes completely, we must identify the outer fluxes as well. In order to calculate the outer fluxes, we perform a similar conservation calculation as in the 1D case to balance mass between the update according to the monolayer h-box method and that according to the standard flux difference method.

In the standard flux difference method, Eq. (5.17) is used for regular sized cells. For the half-size small cells $Q_{i,i}^{L/R}$ along the diagonal, there is an extra barrier flux $H_{i,i}$, and so the following update is used:

$$\begin{aligned} Q_{i,i}^{R,n+1} &= Q_{i,i}^{R,n} - \frac{\Delta t}{0.5\Delta x} F_{i+1/2,i} + \frac{\Delta t}{0.5\Delta y} G_{i,i-1/2} - \frac{\Delta t\sqrt{2}}{0.5\Delta y} H_{i,i} \\ Q_{i,i}^{L,n+1} &= Q_{i,i}^{L,n} + \frac{\Delta t}{0.5\Delta x} F_{i-1/2,i} - \frac{\Delta t}{0.5\Delta y} G_{i,i+1/2} + \frac{\Delta t\sqrt{2}}{0.5\Delta y} H_{i,i} \,. \end{aligned} \tag{5.21}$$

The $\sqrt{2}$ factor comes in because of the length of the barrier segment in each cut cell [40].

In the monolayer h-box method, we can generalize from Fig. 5.8 and have the following up-

dates:

$$
\begin{aligned}
Q_{i,i}^{R,n+1} &= \hat{Q}_i^{R,n+1} \\
Q_{i,i}^{L,n+1} &= \hat{Q}_i^{L,n+1},
\end{aligned}
\tag{5.22}
$$

and for cells at the either tip of the barrier,

$$
\begin{aligned}
Q_{0,-1}^{n+1} &= a_2 \hat{Q}_0^{R,n+1} + (1 - a_2)\overline{Q}_{0,-1}^{n+1} \\
Q_{N_x+2,N_y+1}^{n+1} &= a_2 \hat{Q}_{N_x+1}^{R,n+1} + (1 - a_2)\overline{Q}_{N_x+2,N_y+1}^{n+1} \\
Q_{-1,0}^{n+1} &= a_2 \hat{Q}_0^{L,n+1} + (1 - a_2)\overline{Q}_{-1,0}^{n+1} \\
Q_{N_x+1,N_y+2}^{n+1} &= a_2 \hat{Q}_{N_x+1}^{L,n+1} + (1 - a_2)\overline{Q}_{N_x+1,N_y+2}^{n+1},
\end{aligned}
\tag{5.23}
$$

and for cells simultaneously covered by two adjacent h-boxes ($i = 1, N_x$),

$$
\begin{aligned}
Q_{i+1,i}^{n+1} &= a_2 \hat{Q}_i^{R,n+1} + a_2 \hat{Q}_{i+1}^{R,n+1} + (1 - 2a_2)\overline{Q}_{i+1,i}^{n+1} \\
Q_{i-1,i}^{n+1} &= a_2 \hat{Q}_i^{L,n+1} + a_2 \hat{Q}_{i+1}^{L,n+1} + (1 - 2a_2)\overline{Q}_{i-1,i}^{n+1},
\end{aligned}
\tag{5.24}
$$

and finally for the cells diagonally across to the small cells:

$$
\begin{aligned}
Q_{i+1,i-1}^{n+1} &= a_3 \hat{Q}_i^{R,n+1} + (1 - a_3)\overline{Q}_{i+1,i-1}^{n+1} \\
Q_{i-1,i+1}^{n+1} &= a_3 \hat{Q}_i^{L,n+1} + (1 - a_3)\overline{Q}_{i-1,i+1}^{n+1} .
\end{aligned}
\tag{5.25}
$$

The update for the h-box cells looks like:

$$\begin{aligned}
\hat{Q}_i^{R,n+1} &= \frac{1}{\sqrt{2}}\big(a_1 Q_{i,i}^{R,n} + a_2(Q_{i+1,i}^n + Q_{i,i-1}^n) + a_3 Q_{i+1,i-1}^n\big) \\
&\quad - \frac{\Delta t}{\Delta x}(\hat{F}_{i,1/2} - \hat{F}_{i,-1/2}) - \frac{\Delta t}{\sqrt{2}\Delta x}(\hat{G}_{i+1/2}^R - \hat{G}_{i-1/2}^R) \\
\hat{Q}_i^{L,n+1} &= \frac{1}{\sqrt{2}}\big(a_1 Q_{i,i}^{L,n} + a_2(Q_{i,i+1}^n + Q_{i-1,i}^n) + a_3 Q_{i-1,i+1}^n\big) \\
&\quad - \frac{\Delta t}{\Delta x}(\hat{F}_{i,-1/2} - \hat{F}_{i,-3/2}) - \frac{\Delta t}{\sqrt{2}\Delta x}(\hat{G}_{i+1/2}^L - \hat{G}_{i-1/2}^L),
\end{aligned} \tag{5.26}$$

where $\hat{G}_{i-1/2}^{L/R}$ denotes the (grid oriented) transverse fluxes in between h-box cells i and $i-1$ either above (L) or below (R) barrier and $\hat{F}_{i,-1/2}$, $\hat{F}_{i,-3/2}$ and $\hat{F}_{i,1/2}$ denote the normal fluxes at: the barrier, the outer left flux, and the outer right flux, respectively (see Fig. 5.10), for h-box i. Note that these fluxes are grid oriented for purposes of conservation calculations.

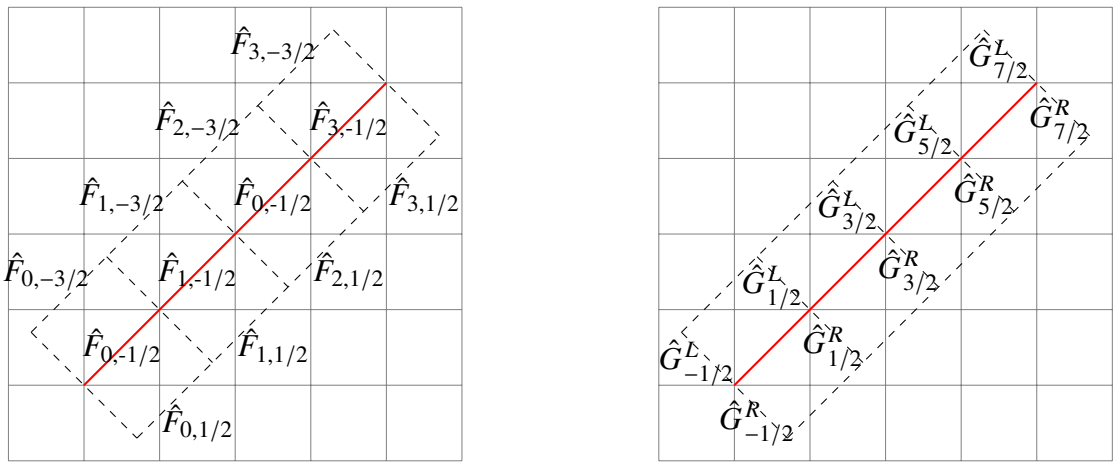
Now we select a neighborhood around the small cells to sum up the updated mass to compare it between the two methods. To do so, we select the cells whose updates are affected by the h-box updates, which is again large enough range to consider given the CFL condition. We look at the mass on each side of the barrier at a time:

$$\begin{aligned}
M_L^{n+1} &= \sum_{i=0}^{N_x+1}\left(0.5\, Q_{i,i}^{L,n+1} + Q_{i-1,i}^{n+1} + Q_{i,i+1}^{n+1} + Q_{i-1,i+1}^{n+1}\right) \\
M_R^{n+1} &= \sum_{i=0}^{N_x+1}\left(0.5\, Q_{i,i}^{R,n+1} + Q_{i+1,i}^{n+1} + Q_{i,i-1}^{n+1} + Q_{i+1,i-1}^{n+1}\right),
\end{aligned} \tag{5.27}$$

where the 0.5 is coming from the volume of the cut cell.

For regular flux difference method, we apply Eqs. (5.17) and (5.21) in the mass formula and we apply Eqs. (5.22) and (5.25) for the h-box method. Setting $M_{R,\text{regular}}^{n+1} + M_{L,\text{regular}}^{n+1} = M_{R,h\text{-box}}^{n+1} + M_{L,h\text{-box}}^{n+1}$ cancels out like terms and leaves only the following:

$$0.5\left(-\sum_{i=0}^{N_x+1}\left(\hat{F}_{i,1/2} - \hat{F}_{i,-3/2}\right) + \frac{1}{\sqrt{2}}\hat{G}_{-1/2}^L - \frac{1}{\sqrt{2}}\hat{G}_{N_x+1/2}^L + \frac{1}{\sqrt{2}}\hat{G}_{-1/2}^R - \frac{1}{\sqrt{2}}\hat{G}_{N_x+1/2}^R\right)$$



(a) Normal fluxes. (b) Transverse fluxes.

Figure 5.10: Notations of fluxes on h-box edges.

$$= a_2(\Delta Q^{n+1}_{0,-1} + \Delta Q^{n+1}_{N_x+2,N_y+1}) + a_2(\Delta Q^{n+1}_{-1,0} + \Delta Q^{n+1}_{N_x+1,N_y+2})$$
$$+a_3 \sum_{i=0}^{N_x+1} \left(\Delta Q^{n+1}_{i-1,i+1} + \Delta Q^{n+1}_{i+1,i-1} \right) + 2a_2 \left(\sum_{i=1}^{N_x+1} \Delta Q^{n+1}_{i-1,i} + \sum_{i=0}^{N_x} \Delta Q^{n+1}_{i+1,i} \right). \tag{5.28}$$

That is, all fluxes except the outer fluxes of h-box edges cancel (left hand side of Eq. (5.28)) and the fluxes of cells affected by the h-box cells remain in proportion to how much area it overlaps with the h-box (right hand side of Eq. (5.28)).

Given this condition for conservation Eq. (5.28), we can assign each of the h-box outer fluxes to a combination of the remaining fluxes on the right hand side. We group the remaining fluxes by their vicinity to the h-box outer edge whose flux we desire to set. Specifically, we set:

$$\hat{F}_{i,1/2} = \text{-}2\left(a_2 \Delta Q^{n+1}_{i+1,i} + a_2 \Delta Q^{n+1}_{i,i-1} + a_3 \Delta Q^{n+1}_{i+1,i-1} \right) \text{ for } i = 1, N_x$$
$$\hat{F}_{i,-3/2} = 2\left(a_2 \Delta Q^{n+1}_{i-1,i} + a_2 \Delta Q^{n+1}_{i,i+1} + a_3 \Delta Q^{n+1}_{i-1,i+1} \right) \text{ for } i = 1, N_x \tag{5.29}$$

for the normal outer fluxes in the interior h-boxes, and for normal and transverse fluxes at either

end of h-box monolayer, we set:

$$
\begin{aligned}
\hat{F}_{0,1/2} &= \text{-}2a_2\Delta Q^{n+1}_{1,0} \\
\hat{F}_{0,-3/2} &= 2a_2\Delta Q^{n+1}_{0,1} \\
\hat{F}_{N_x+1,1/2} &= \text{-}2a_2\Delta Q^{n+1}_{N_x+1,N_x} \\
\hat{F}_{N_x+1,-3/2} &= 2a_2\Delta Q^{n+1}_{N_x,N_x+1} \\
\hat{G}^R_{-1/2} &= 2\sqrt{2}(a_2\Delta Q^{n+1}_{0,-1} + a_3\Delta Q^{n+1}_{1,-1}) \\
\hat{G}^R_{N_x+1/2} &= 2\sqrt{2}(a_2\Delta Q_{N_x+2,N_x+1} + a_3\Delta Q^{n+1}_{N_x+2,N_x}) \\
\hat{G}^L_{-1/2} &= 2\sqrt{2}(a_2\Delta Q_{-1,0} + a_3\Delta Q_{-1,1}) \\
\hat{G}^L_{N_x+1/2} &= 2\sqrt{2}(a_2\Delta Q_{N_x+1,N_x+2} + a_3\Delta Q^{n+1}_{N_x,N_x+2}).
\end{aligned}
\tag{5.30}
$$

Note that the remaining fluxes are set using cells' incremental updates $\Delta Q_{i,j}$. With these the h-boxes are then updated according to Eq. (5.26) and rotated back into the original grid.

5.3.3 2D Problem III: General angled barrier

In this section we carry the same monolayer and conservation idea to the general case of having an arbitrarily angled barrier as in Fig. 5.11. We create normal h-boxes off the barrier and perform wave redistribution at the barrier edge, updating the h-boxes and underlying cells.

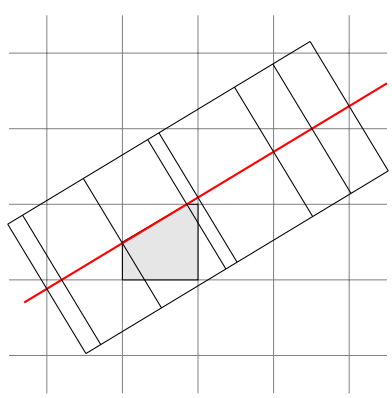


Figure 5.11: Case of arbitrary angled barrier. Note both the fragments within each h-box and the widths of each h-box, how they all differ across h-boxes. The update formula for highlighted cell will require updated averages of three h-boxes.

The added complexities are twofold. First, the update formula for the underlying cells become more complicated as multiple h-boxes can cover a small cell (see highlighted cell in Fig. 5.11). Also, they can either fully or partially cover the small cell. This means that in some cases, the h-box averages will update the small cells completely, and in other cases, the original cell regular updates will also contribute partially to the update values of the small cells. As can be observed in Fig. 5.11, however, the small cells that will be only partially covered will be those that have area larger than $0.5\Delta x\Delta y$ for which regular updates are not unstable or CFL-violating.

Updates

Let Q_s^{n+1} denote the updated average of a small cell fully covered by h-box(es) $\{\hat{Q}_i\}$, whose index(es) (numbered from leftmost h-box to the right) is in a set denoted by $\{i_s\}$. Let α_i denote the areas of the h-boxes that cover the small cell s. Then we have:

$$Q_s^{n+1} = \sum_{i \in i_s} \alpha_i \hat{Q}_i^{n+1}. \tag{5.31}$$

Now let $Q_{p_s}^{n+1}$ denote the updated average of a small cell partially covered by h-box(es) $\{\hat{Q}_i\}$, whose index(es) are again denoted by $\{i_s\}$. Let α_i denote the areas of the h-boxes that cover the small cell p_s. Also, let A_{p_s} be the area of the small cell p_s. If $\hat{Q}_{p_s}^{n+1}$ is the regular update of the cell, then we have:

$$Q_{p_s}^{n+1} = (A_{p_s} - \sum_{i \in i_s} \alpha_i)\hat{Q}_{p_s}^{n+1} + \sum_{i \in i_s} \alpha_i \hat{Q}_i^{n+1}. \tag{5.32}$$

Computationally, we do this multiple h-box update by first isolating each cell affected by h-boxes, finding which h-box covers it, and updating it by the area-weighted new h-box averages and the regular averages if necessary (for the partially covered cells). The isolation of which h-boxes contribute to which cells are part of a pre-processed geometrical computation, whose data is then retrieved on the run of the simulation.

Conservative Calculations

Second, the conservation calculation needs to be applied cell by cell. The correction terms required to conserve mass is applied to the h-box values after updating them using fluctuations and is the same as in the diagonal barrier case in Section 5.3.2, where we take the regular update increment $\Delta Q_{p_s}^{n+1}$ of each cell partially covered by h-boxes and weight them by α_i and distribute as out-fluxes of the ith h-box to balance the mass equation, $M_{R,\text{regular}}^{n+1} + M_{L,\text{regular}}^{n+1} = M_{R,h\text{-box}}^{n+1} + M_{L,h\text{-box}}^{n+1}$ (see Fig. 5.12a). The coefficients in front of the increments $\Delta Q_{i,j}$ (e.g. $\sqrt{2}, 0.5$) were coming from the small cell areas and barrier segment lengths, and for sake of notation, we include the weights together with $\Delta Q_{i,j}$ to define overall effective flux $\mathcal{F}$. For example, in Eq. (5.28), we can define $\mathcal{F}_i = 0.5\hat{F}_{i,1/2}$.

In equation form, let $\mathcal{F}_o^i$ denote the total effective out-flux that needs to be determined for the ith h-box to conserve mass. We call this total, since we combine both x-flux F and y-flux G and define as $\mathcal{F}_o^i$ as in the diagonal example. Also, let p_s denote all the cells that contribute to the ith h-box, with α_i denoting the area that cell p_s contributes to the ith h-box. Then we have:

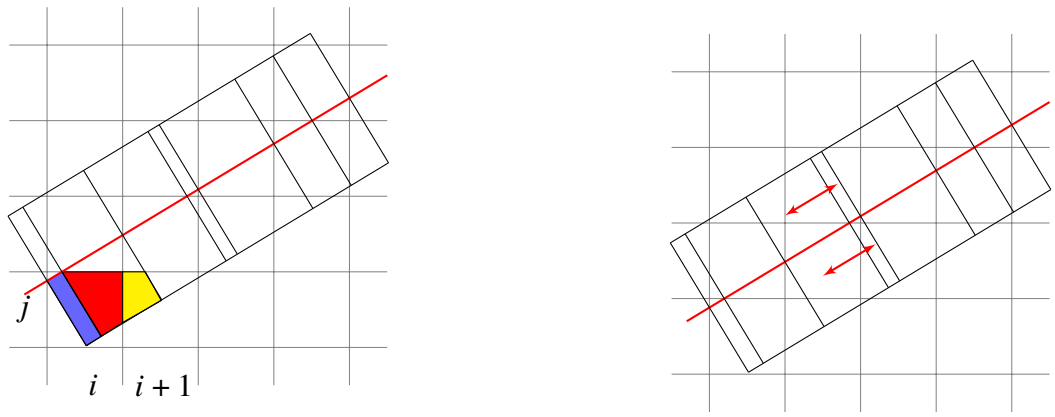
$$\mathcal{F}_o^i = -\sum_{p_s} \alpha_i \Delta Q_{p_s}^{n+1}. \tag{5.33}$$

This is the corrective term to each updated h-box average.

5.3.4 Limitation of the monolayer slanted method

As seen in Fig. 5.12b, we cannot take regular time step updates when doing transversal computation because of the irregular h-box widths. We attempted to apply state redistribution (another small cell method explained in the following chapter) in the transverse direction, but the result was that this only works for reflection only problems and not for overtopping problems. Applying yet another h-box method in the transverse direction here will be too complicated.

We are not certain why this is the case, but we can suspect that in the reflection only case, the most important wave direction is the normal to barrier direction. However, in the overtopping



(a) In the first leftmost lower h-box, the total effective outer flux will be the area of blue region times $\Delta Q_{i,j}^{n+1}$, and that of the second lower h-box will be the area of red region times $\Delta Q_{i,j}^{n+1}$ plus the area of yellow region times $\Delta Q_{i+1,j}^{n+1}$. This term is then used to preserve mass.

(b) Small cell problems within the h-box layer in the transverse direction. Standard updates using transverse fluctuations computed between the h-box cells will violate CFL condition. State redistribution (described in Chapter 6) was unsuccessfully used to get around this.

Figure 5.12: Computational considerations for h-box method in general angle

case, the wave motion depends also heavily in the transverse direction along the barrier, which a simple fix with SRD does not suffice. Furthermore, with a monolayer of h-boxes, we no longer have an approximation of the wave at the cut cell edges that will hit the barrier edge (as explained in the 1D double h-box method). In Section 5.4.5, we will show an example of using this WR and state redistribution hybrid method on a complete blockage problem at 20° to the x-axis and also its convergence to numerical solution computed via mapped grid.

5.4 Model problems

5.4.1 1D computational results

We provide here a 1D example simulating a barrier on a sloping beach with dry state condition on the right side of the barrier to test inundation. The boundary conditions on either end of the domain are wall boundary conditions, to also test conservation. We observe conservation of mass to machine precision. The number of cells is $N = 400$, with $\alpha = 0.1$. The CFL was 1. The barrier height is $\ell = 0.35$. The initial condition is shown in Fig. 5.13.

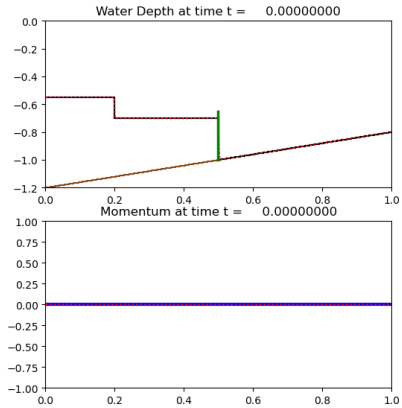
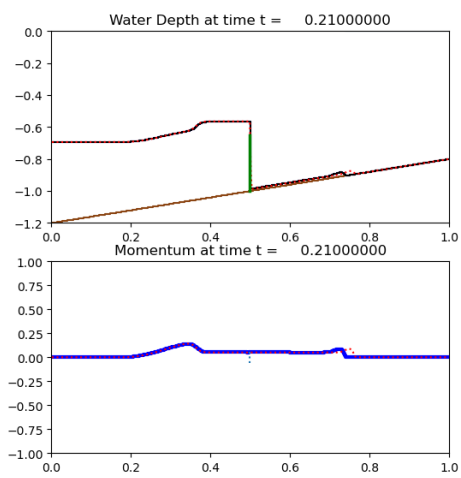
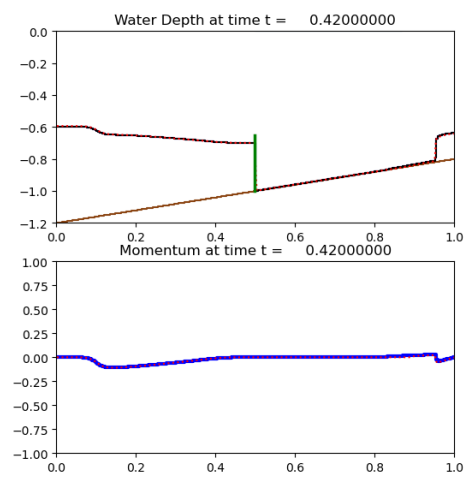


Figure 5.13: Initial condition of sloping beach example, with $N = 400$, $\alpha = 0.1$, $\ell = 0.35$.

In the water depth plot, the green vertical line at the center represents the flux-allowed embedded boundary, or the zero-width barrier, and the sloping line on the bottom represents the sloping bathymetry. The water is on the left of the barrier, and we have a dam-break that initiates the wave. The figure on the bottom shows the initial momentum of the water, which is zero. The two solutions are plotted on top of each other, with the solid blue and black line representing the h-box solution, and the dotted red line representing the LTS solution.



(a) H-box solution in black and blue solid line and LTS solution in red dotted line. Overtopping and inundation are both captured here.



(b) H-box solution in black solid line and LTS in red dotted line. Wave has overtopped and bounced off the right boundary.

Figure 5.14: 1D Overtopping on sloping beach

In Fig. 5.14a, we present the overtopping action and inundation. We note that since the equations are 1D SWE, there is no vertical motion in the y direction simulating the water's flowing

down the barrier. Instead there is only lateral motion in the x direction once water overtops the barrier. In Fig. 5.14b, we present the simulation at time $t = 0.42$. Here at time $t = 0.42$, the wave initiated by the dam break has overtopped the barrier and reflects off the right endpoint. The LTS solution and the h-box method solution to problem in Fig. 5.13 line up closely, while the h-box method requires no tracking of waves.

5.4.2 2D Case I: Horizontal barrier

Here we present a 100×100 grid from $[0, 0]$ to $[1, 1]$ with water height $h = 1.0$ all round with the exception of a square shaped step of positive 1.5 in the left lower corner. CFL was 1 here as well. There is no initial momentum. This is to produce a dam-break like wave at an oblique angle to the barrier, to highlight the method's adaptability to two dimensions. The height of the barrier is 1.5 and the barrier is located just above the midpoint of the y-axis, $y_{bar} = 0.502$ ($\alpha = 0.2$). We will see that this is an overtopping example. The boundary conditions are extrapolation conditions all around.

For comparison we provide the results of the same height and momentum initial conditions with the barrier of same height on a grid edge, $y_{bar} = 0.5$. The numerical solution is computed using only WR in y-direction exactly at a grid edge $y = 0.5$ ($\alpha = 0.0$). This is when the h-boxes are precisely the regular sized cells on either side of the barrier and only changes the y-flux at the edge where the barrier lies. We compare our results with this example because other than the 0.02 distance difference in the location of the barrier, all else remain the same, and results should be comparable.

For clarity and better comparison, we plot the contours of the water height normalized to 0.0 and the initial condition is shown in Fig. 5.15. The step in water height rescinds and propagates a wave at an angle to the barrier and hits the barrier around time $t = 0.2$. In Fig. 5.16, we see the waves overtopping the barrier at time $t = 0.3$. The two examples give very similar results, as expected. We do not provide actual error, but show qualitative similarity. We show more detailed error analysis for our slanted example later.

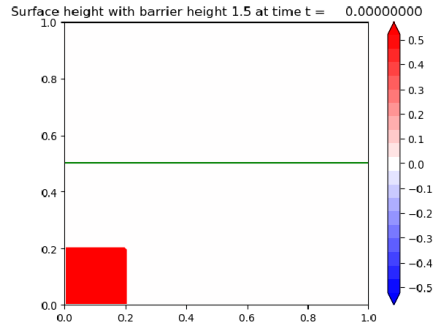
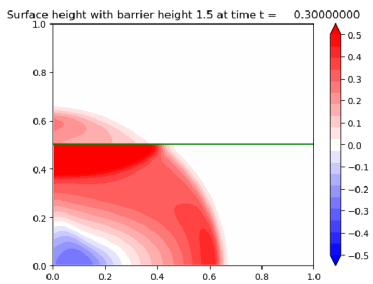
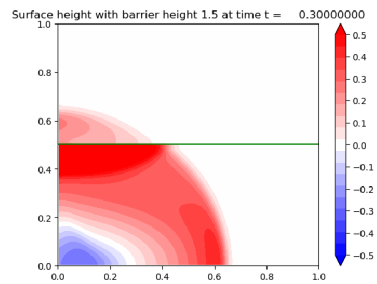


Figure 5.15: Contour plot of initial condition of oblique wave example with parallel barrier: $N = 100$, $\alpha = 0.2$, wall height $\ell = 1.5$, and jump $\Delta h = 1.5$.

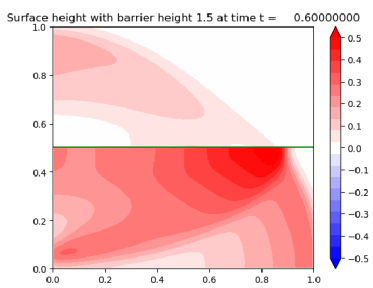


(a) $y_{bar} = 0.5$ at time $t = 0.3$.

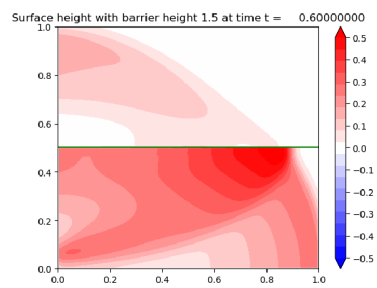


(b) $y_{bar} = 0.502$ at time $t = 0.3$.

Figure 5.16: Contour plot I of WR only result vs. h-box method result for parallel barrier example



(a) $y_{bar} = 0.5$ at time $t = 0.6$.



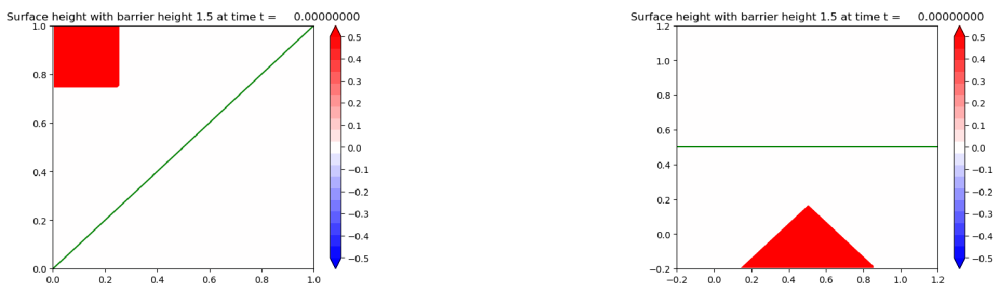
(b) $y_{bar} = 0.502$ at time $t = 0.6$.

Figure 5.17: Contour plot II of WR only result vs. h-box method result for parallel barrier example

Not only is the overtopping captured but also the reflection from the barrier, as seen in the u-shaped contour below the barrier. The wave continues the overtop and reflect and partially exits the domain at time $t = 0.6$ in Fig. 5.17.

5.4.3 2D Case II: Diagonal barrier

Here we present again the same grid with same dimension and number of cells, but now with a diagonal barrier at 45°. The height of the barrier is also the same, $\ell = 1.5$, and the height of water is $h = 1.0$ all throughout, with the exception of a raise of 1.5 over a small square region in the left-upper corner (see Fig. 5.18a). CFL is 1. The boundary conditions are wall boundary conditions on the left and top boundaries while the right and bottom boundaries are extrapolated for visual clarity in the waves.



(a) Diagonal problem initial state. (b) Rotated equivalent for comparison.

Figure 5.18: Initial condition of the diagonal problem and its equivalent problem with horizontal barrier.

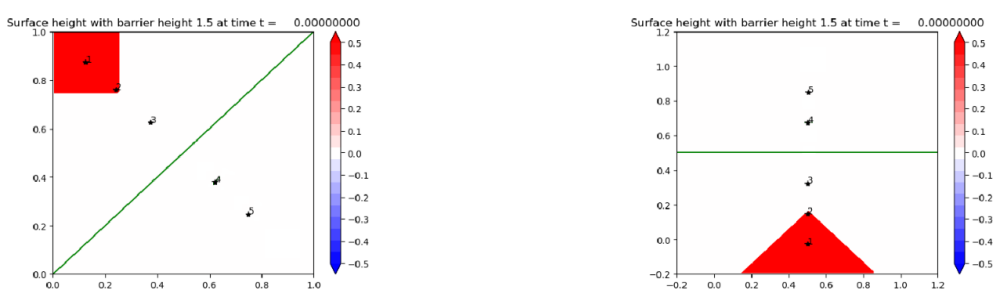
For comparison we provide a “rotated" version of the same initial conditions, namely a horizontal barrier problem that is located on the middle grid edge so we can only use WR and which has a half-square shaped raise on the bottom center of the grid (see Fig. 5.18b). The half-square (i.e. triangular) shape is used in comparison instead of full square as in the diagonal problem because of the shape of the overall grid and the fact that the upper half of the square step is the part that actually generates the wave that hits the barrier. Also, note the domain is extended to 1.4 to match the distance between the corner of the square step to the barrier ($\sqrt{2}/2 \approx 0.7 = 1.4/2$).

We see in Fig. 5.26 (in subsection “Larger Figures" at the end of this chapter) the progress of the propagating wave, which hits the barrier and overtops it in both instances around $t = 0.4$. Again, we see both reflection and overtopping captured in the contours on both problems. We also provide the 1D slices at the center of the motion (across the line $y = -x$ for diagonal case and across the line $x = 0.7$ for rotated equivalent) and compare them in Fig. 5.27 (in appendix). As one

can see especially from the 1D slices, the two results are virtually identical, as expected.

For a closer comparison between the two examples, we put “gauges" at special locations, which are time profiles of the height at fixed points. This is akin to the comparison made in [41], where they test the results of parabolic bowl tsunami. We can put gauges at points that are located proportionally to the barrier in each example (Fig. 5.19) and compare the heights at each time. We compare the parallel barrier (rotationally equivalent) problem results with the diagonal barrier results. Furthermore, we compare these two against results from GEOCLAW, a shallow water simulator with many available parametrizations [41], where the problem has a diagonal barrier with cell-wide thickness.

In Fig. 5.20, we provide the gauge heights' time profile and observe close results. Note that the GEOCLAW comparison problem has thickness on the barrier and can have water on top of the barrier, whereas our model problem does not. This explains the discrepancy in gauge 4, where the GEOCLAW result seems to have less height during $t = 0.5$ to $t = 0.6$ than the parallel and diagonal barrier results in black and dotted red. Some of the missing water is on the barrier in the GEOCLAW result, whereas all the overtopping water is on the other side of the barrier in the model problems. Also note that gauge 6 is flat throughout as the wave has not reached it by $t = 0.6$. This shows that the speed of the two results are consistent with each other. Furthermore, note the scale of the second gauge result, that the difference between GEOCLAW, flat, and diagonal barrier is $O(\Delta x)$.



(a) The rotated example.

(b) The rotated equivalent example.

Figure 5.19: Gauges for closer comparison.

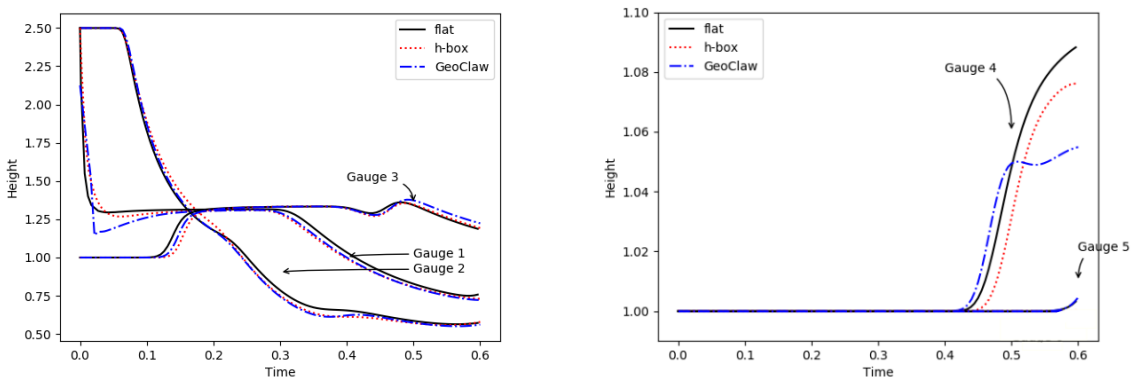


Figure 5.20: Gauge comparisons between the actually rotated problem and rotated equivalent problem.

5.4.4 2D Case III: Diagonal barrier with sloping beach

Here we provide an example similar to case II, except now with a sloping beach. The motivation for this example is to mimic the real-life example of a wave rushing up against a barrier on a beach. A note to be made is that in this example the local bathymetry near the barrier is flat (two cells wide to both sides of barrier), as the conservation calculation made in Section 5.3.2 is only for hyperbolic problems without a source term (e.g. bathymetric variation). To form an h-box monolayer method that is also conservative for varying bathymetry is a work in need of further research.

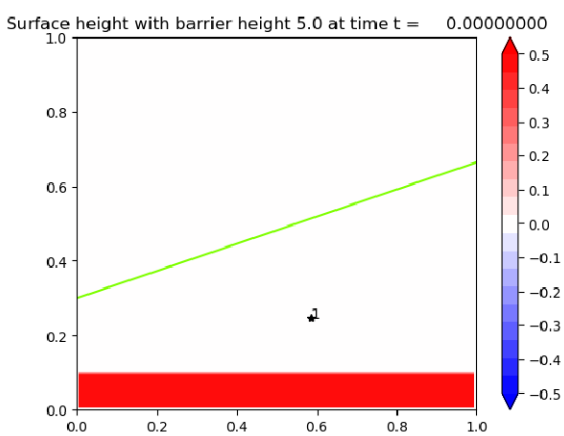
We provide the results in Fig. 5.25 (in "Larger figures"). CFL is 1. The boundary conditions are as same as before, with left and top boundaries being wall conditions and with right and bottom boundaries are extrapolation conditions. The initial condition is a square step in the upper left corner with jump size of 1.2, and the bathymetry is given by $b(x, y) = 0.01x - 0.01y - 2$. The barrier height is $\ell = 1.6$. We present the surface height, normalized again at 0.0. Note how the sloping beach causes the overtopping wave to become more skewed towards the right as it moves closer to “shore". This is in accordance to the dependence of the waves’ speed on the height of water, which decreases as the bathymetry slopes up, causing the accumulation to the right.

5.4.5 Case IV: Arbitrary angled barrier with flat bathymetry

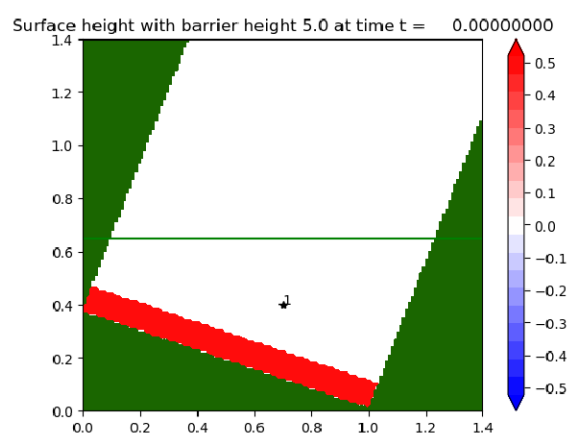
Finally we provide an example of an angled barrier that is 20° to the x-axis. This is to show the method's flexibility and ability to handle different barriers and very small cells. In the diagonal barrier example, the small cells were at the border line of what is considered 'small', i.e. $0.5\Delta x\Delta y$. Here, we have small cells that are as small as 1.85E-05 $\Delta x\Delta y$. The grid is 100×100 cells from $[0, 0]$ to $[1, 1]$. CFL here due to the unsplit nature of the update (update with fluctuation in x and y direction simultaneously) was 0.5.

Complete blockage

We do a dam break problem with a planar wave that comes toward the barrier at an oblique angle. The height of the barrier ℓ is set to be 5.0, and the overall height of water h across the domain is set to be 1.0. The jump of the dam is 0.5. The bathymetry b is flat and set at -2.0. Fig. 5.21 shows the initial setup and displays the water height h. The boundary conditions are all wall conditions all around the domain. We put a gauge point (marked with star and number 1) for later comparison with a parallel barrier simulation (Fig. 5.21b), and it is placed at a distance of 0.25 from the midpoint of the barrier.



(a) Initial condition for 20° barrier. $\ell = 5$, high enough for complete blockage.

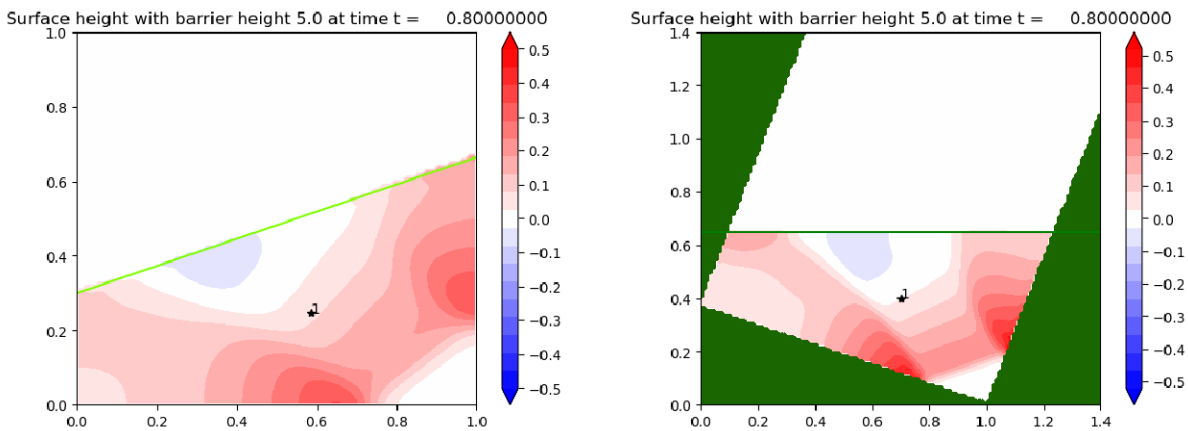


(b) Initial condition for parallel rotated problem, with $\ell = 5$. The green is raised bathymetry acting as wall boundaries.

Figure 5.21: Complete blockage example with 20° barrier and rotated parallel barrier

We compare the 20° barrier results with the result of parallel barrier simulation with the barrier at $y_{bar} = 0.65$, which is on a grid edge and where WR is used. The location y_{bar} is set to match gauge results. The domain $[0, 0]$ to $[1.4, 1.4]$ was expanded for easier geometrical considerations in comparison, but $dx = dy = 0.01$ is the same. The dam jump of 0.5 is rotated appropriately (white in Fig. 5.21b). We compare the two results as before, by way of gauge comparison. We include the GEOCLAW results as well in the blue.

At $t = 0.8$ (Fig. 5.22) the wave that collided with the barrier reaches the corner where the right side of domain and the barrier meet, and is pinched and has reflected downward. Also, the reflected wave from the barrier reaches the bottom of the domain and reflects once more, since the boundary condition is a wall condition (Figs. 5.22a and 5.22b).



(a) $t = 0.8$: the slanted barrier problem.

(b) $t = 0.8$: the parallel rotated problem.

Figure 5.22: Blockage for angled barrier (20.0°)

The difference in the two gauge results (Fig. 5.23) is due to the diffusive nature of our method. Since the state redistribution in the transverse direction adds diffusivity to our method, we observe slightly lower peak in the second hump (the reflected wave) for the rotated problem.

Source of diffusivity in the general angled method and comparison to original h-box method

The reasons for the diffusivity in the results (Fig. 5.23) may be attributed to (a) using only the normal h-boxes in the grid, which can cover multiple small cells at once, (b) using the SRD method in the transverse direction, and (c) correcting for conservation by using the update increments

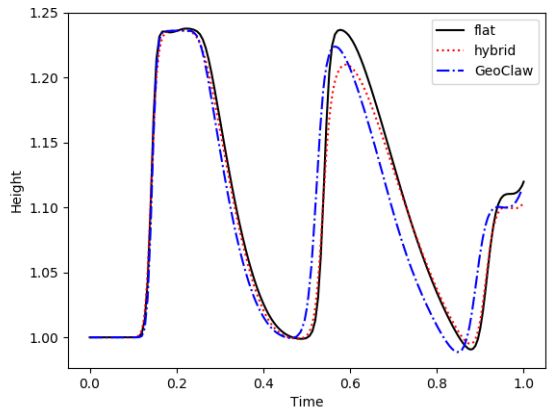


Figure 5.23: Blockage comparisons with WR results using gauge.

from surrounding cells. All these effects are most likely compounded to produce the diffusive result. Also we note that we have only developed a first order method that does not account for slope-limiting at the barrier.

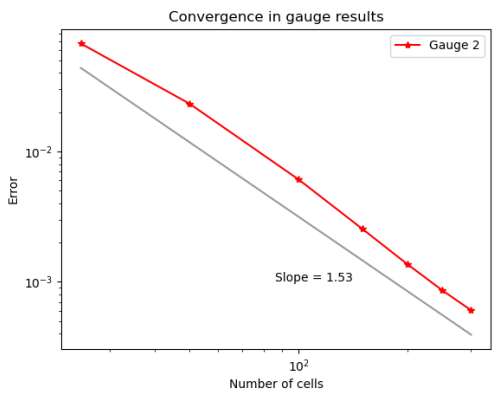
In the original h-box method, multiple h-boxes are used at each edge of the small cell to calculate fluxes, which then update the small cell. Our method only uses *one* rotated edge (the normal barrier edge) of the small cell to calculate the update of the small cell. This makes for a simpler method but as can be seen, comes at the cost of diffusivity. However, this can be made up by increasing the resolution and applying second order updates away from the barrier.

5.5 Convergence for hybrid method

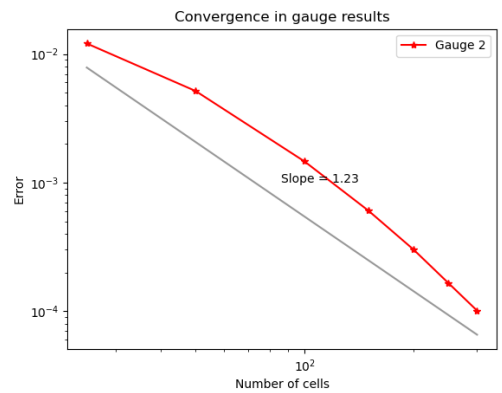
To confirm our slanted barrier method, we run convergence analysis. As before, we set an example with $\ell = 5.0$, and set the dam break jump to be 0.8. We ran the method on multiple dimensions Δx = 1/25, 1/50, 1/100, 1/150, 1/200, 1/250, 1/300 and compared the result against the same parameters solved on a mapped grid result (further explained in). We observe convergence order of 1.5 , as seen in Fig. 5.24a, and observe close match in the gauge result between a lower and higher resolution, as seen in Fig. 5.24c.

Δx	N_x, N_y	L_1 Error	L_∞ Error
4.e-2	25	6.72e-2	1.21e-2
2.e-2	50	2.32e-2	5.15e-3
1.e-2	100	6.10e-3	1.45e-3
0.666e-2	150	2.55e-3	6.06e-4
0.5e-2	200	1.37e-3	3.02e-4
0.4e-2	250	8.6e-4	1.65e-4
0.333e-2	300	6.0e-4	1.0e-4

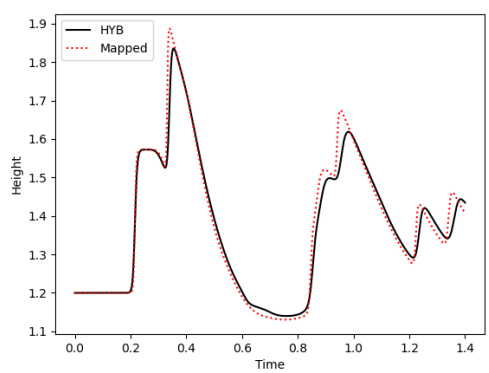
Table 5.1: L_1 and L_∞ errors computed at gauge point (0.5, 0.39).



(a) 1st order L_1 convergence on 20° barrier example with hybrid method



(b) 1st order L_∞ convergence on 20° barrier example with hybrid method



(c) Gauge comparison. Hybrid 300×300 and mapped 600×600.

Figure 5.24: Hybrid method performance on reflection only problem.

5.6 Computational savings compared to GEOCLAW

If we compare the h-box hybrid method on the complete blockage example against GEOCLAW , where we refine the barrier to be one cell wide, we observe the following reduction in time steps. For a simulation on grid 250×250, we see that GEOCLAW took 4574 time steps, whereas the hybrid h-box method took 2141, for about a half reduction. The minimum Δt observed in GEOCLAW was 7.7e-06, whereas in the hybrid method we observed 1.0e-04. The average Δt for GEOCLAW was 6.1e-4, whereas that for h-box method was 9.8e-4. For a simulation on grid 300×300, we observed 5459 time steps with GEOCLAW , compared to 2582 with the h-box method, and minimum Δt of 3.0e-05 with GEOCLAW , compared to 1.5e-04 with h-box. The average Δt for GEOCLAW was 5.1e-4, whereas that for h-box method was 8.1e-4. Thus we can see that h-box does relieve the CFL timestep restriction and reduces computational cost than barrier refinement.

Figures

In the following pages we show the plots for the diagonal barrier problem in sequence. First we see the sloping beach problem for the diagonal barrier. Then we show the flat bottom example with its comparison to the equivalent parallel barrier example.

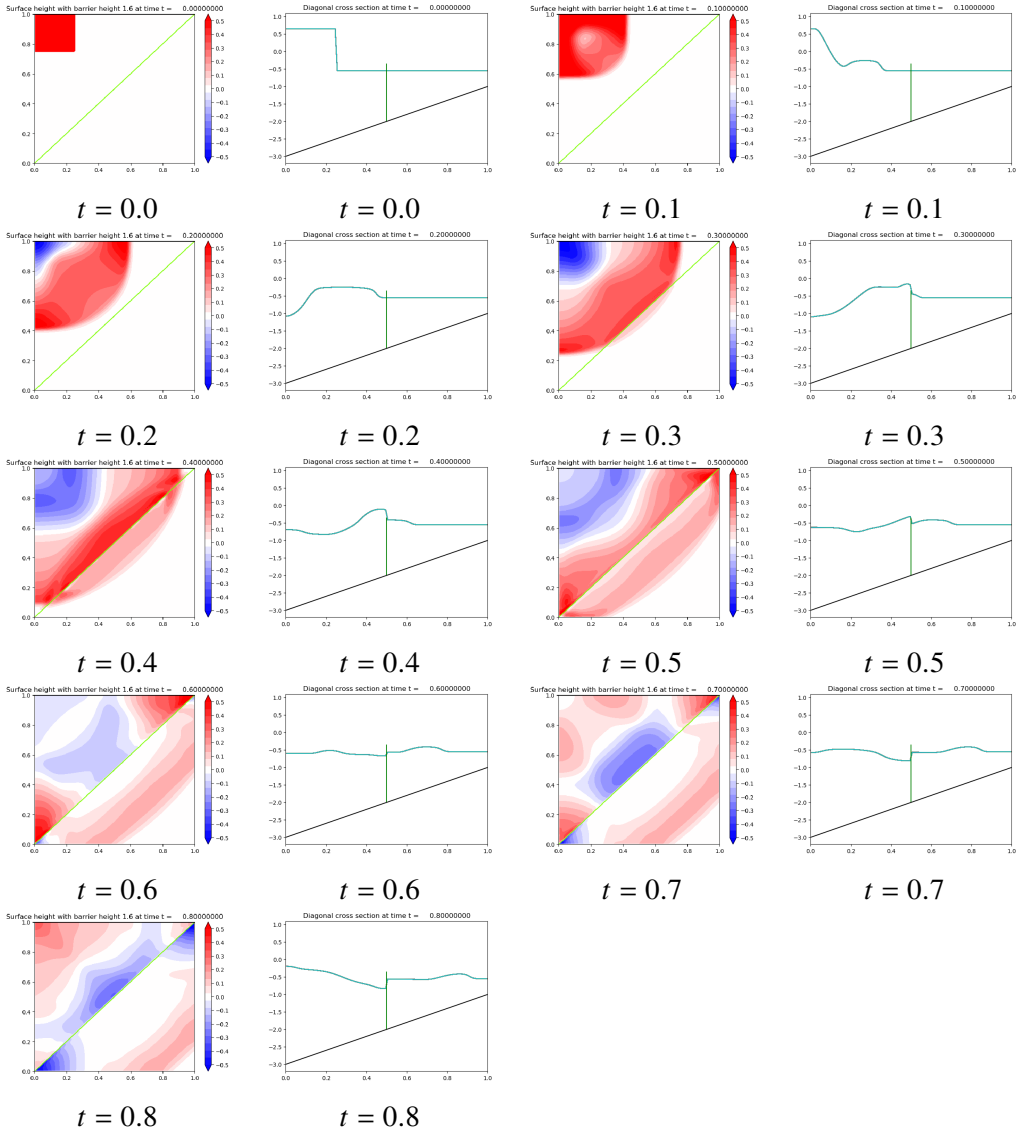


Figure 5.25: Progression of rotated barrier with sloping beach problem. On left we see the colored contour plot and on the right, the diagonal 1D slice.

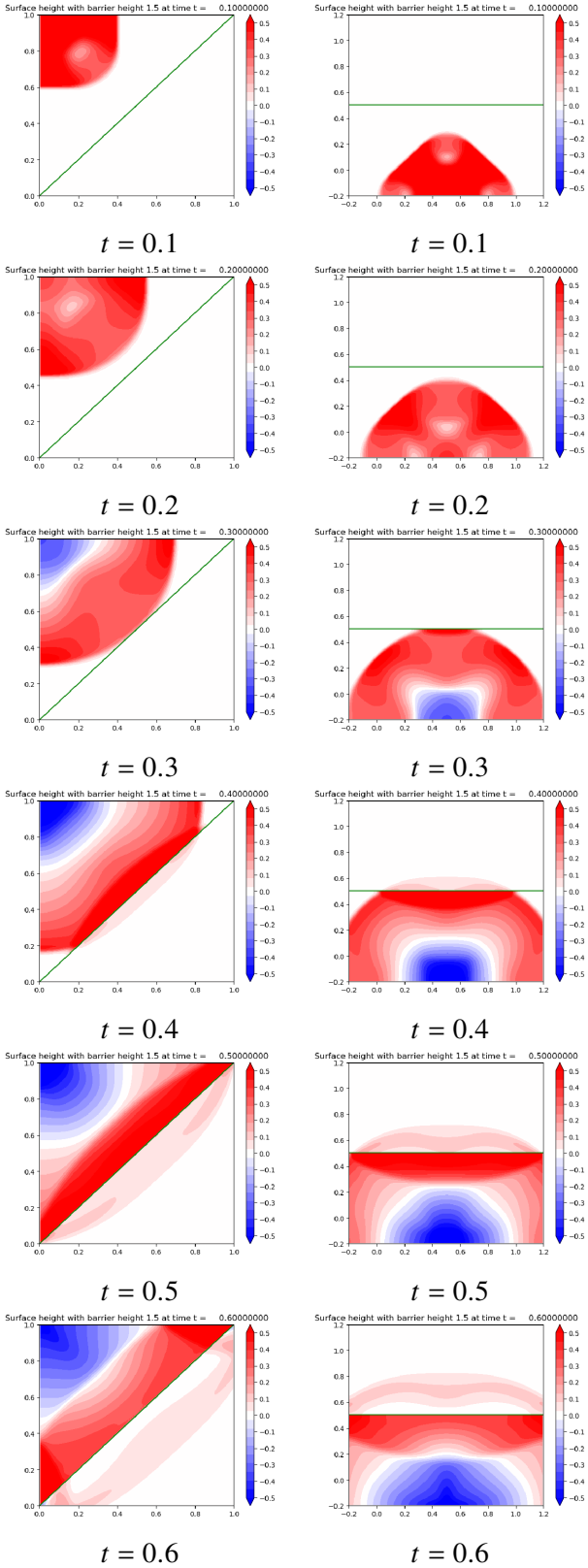


Figure 5.26: Wave progression in diagonal barrier problem (flat bathymetry) and its rotated equivalent.

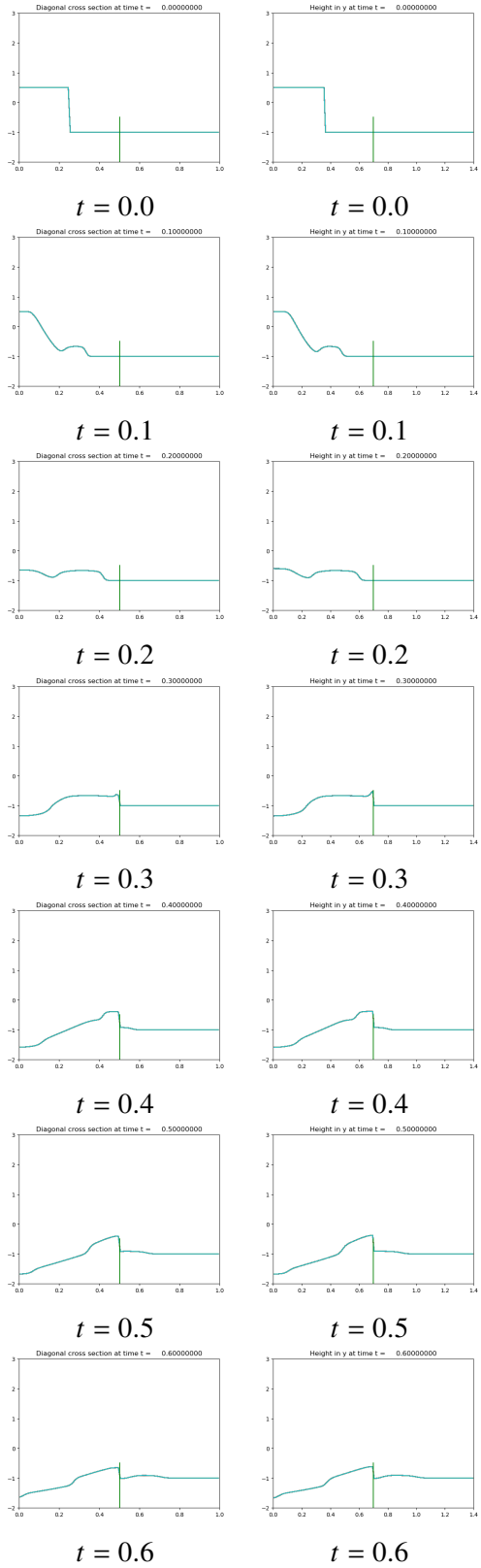


Figure 5.27: 1D slice of the progression of wave in diagonal barrier problem in Fig. 5.26 (left) and its rotated equivalent (right).

Chapter 6: State Redistribution

State redistribution (SRD) is a post-processing method, which seeks to relieve CFL restriction by averaging out an unstable update *a posteriori*. The idea is to take a CFL violating, unstable update and fix it by redistributing the update with neighboring cells. Because the method happens after the update and does not involve calculations on the fly, it is a faster algorithm than the h-box method. From this chapter forward, we will devote our cut cell methods to the two 2D model problems, the 20° linear barrier and the V barrier. Furthermore, the numerical method on the regular sized cells is what has been delineated in Chapter 3, so we will discuss the numerical method on cut cells only, namely SRD.

6.1 Numerical method on cut cells

6.1.1 Attempt with h-boxes: Motivation for Using State Redistribution (SRD)

As discussed in the previous chapter, we have attempted to solve this model problem using h-box type methods, with parallel, and diagonal barriers to deal with the small cell problem. The evident difficulties of the h-box method applied to our model problem are (1) cumbersome geometrical calculations, (2) complicated update formulas, and (3) ambiguity in more complex cases. The biggest challenge of the h-box method is finding fluxes at the non-barrier edges of the cut cells (Fig. 6.1). This is because h-box extensions off those edges will necessarily cross over the barrier, making it difficult to calculate their average. We will show how SRD overcomes all these challenges and can simulate arbitrary angled barriers and the V-shaped barriers.

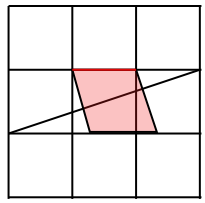


Figure 6.1: The shape of one h-box applied to problem at hand, extruding from edge in red. Note how it crosses over the barrier. LTS avoids this by taking normal h-box averages (e.g. in Fig. 5.11) at multiple time steps, all the while tracking waves from every edge. In 2D, this will become much more complicated, as there will be waves from multiple edges that will hit the barrier at different times, calling for a simpler method.

6.1.2 State Redistribution (SRD)

The equation Eq. (3.6) or Eq. (3.19) shown previously does not work on cut cells because of the geometry of the cut cells. We have different kinds of cut cells as shown in Fig. 6.2.

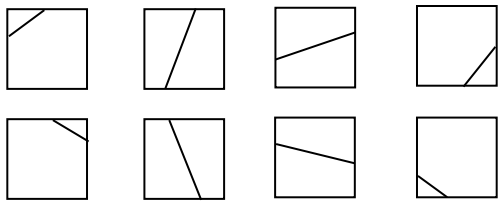


Figure 6.2: All possible types of cut cells for a barrier segment [20].

Instead, we apply the state redistribution method on these types of cut cells in 2D. The main idea of the method is to first do a conservative but unstable update on all cut cells and correct for the instability by doing a stable redistribution of those updates with neighboring cells.

6.1.3 1D SRD

The main components of the method are best explained in 1D. In Fig. 6.3 we have an irregular grid with two small cells, Q_1 and Q_3, with areas $\alpha\Delta x$ ($\alpha < 0.5$) [42], where a normal update cannot be taken due to the CFL condition. Every other grid cell has length Δx. This is the same grid setup as in the example in [12], but we provide a slightly different approach for neighborhood selection which will also apply to our 2D model problems.

Before applying the state redistribution, each cell, including the small cells, is given a regular

update with the Δt prescribed by the CFL condition with a regular Δx, *regardless* of individual cell size:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\alpha_i \Delta x}(\mathcal{A}^+ \Delta Q_{i-1/2} + \mathcal{A}^- \Delta Q_{i+1/2}),$$

where $\alpha_i = \alpha$ when $i = 1, 3$ and $\alpha_i = 1$ when $i \neq 1, 3$. This will be an unstable yet conservative update for the small cells because all the fluctuation $(\mathcal{A}^+ \Delta Q_{i-1/2} + \mathcal{A}^- \Delta Q_{i+1/2})$ is taken into account.

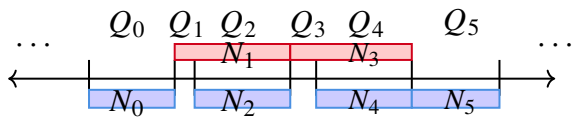


Figure 6.3: An irregular grid with small cells and the neighborhoods used in SRD in blue and red. A variation from [12].

Then we define *neighborhoods* for each cell, $\{N_i\}$, which are collection of cells to achieve a collective area greater than $0.5\Delta x$. This means that for cells with size $> 0.5\Delta x$, they are their own neighborhood, e.g. $N_0 = Q_0$ (blue in Fig. 6.3). For the small cells, we can select the regular sized cell immediately to their right and this achieves a collective area larger than $0.5\Delta x$. The two red boxes in Fig. 6.3 are such neighborhoods for Q_1 and Q_3. This is different from the example shown in [12] where neighboring cells are taken from both sides. However, as we shall see, forming this one sided neighborhood is conservative and will be needed in our model problem due to the presence of the zero width barrier.

The neighborhoods also have a special average. The average of a neighborhood is calculated by using its comprising cells' averages, areas, and their *overlap counts*, which is the total number of neighborhoods lying over them. For instance, cells Q_2 and Q_4 both have overlap count of 2

(N_1, N_2 and N_3, N_4 in Fig. 6.3). This gives the following neighborhood averages:

$$N_1 = \frac{1}{\alpha + \frac{1}{2}}(\alpha Q_1^{n+1} + \frac{1}{2}Q_2^{n+1}), \tag{6.1}$$

$$N_3 = \frac{1}{\alpha + \frac{1}{2}}(\alpha Q_3^{n+1} + \frac{1}{2}Q_4^{n+1}), \tag{6.2}$$

$$N_i = Q_i^{n+1} \text{ for } i \neq 1, 3. \tag{6.3}$$

Note that Δx factors out and weights for Q_2, Q_4 have overlap count 2 in the denominator.

Finally, stabilized updates of the cells are found by using the neighborhood average values lying over the cells and averaging them if there are multiple neighborhoods:

$$Q_2^{n+1} = \frac{1}{2}(N_1 + N_2) \tag{6.4}$$

$$Q_4^{n+1} = \frac{1}{2}(N_3 + N_4) \tag{6.5}$$

$$Q_i^{n+1} = N_i \text{ for } i \neq 2, 4. \tag{6.6}$$

Note that the discounting by overlap count in (Eqs. (6.1) and (6.2)) gives the effective volume of the neighborhood and is used for conservation purposes as can be seen by checking the mass from before to after SRD stabilization:

$$\overbrace{Q_0 + \alpha Q_1 + Q_2 + \alpha Q_3 + Q_4 + Q_5}^{\text{pre-SRD}}$$

$$= Q_0 + \underbrace{\alpha N_1 + \frac{1}{2}(N_1 + N_2) + \alpha N_3 + \frac{1}{2}(N_3 + N_4)}_{SRD} + Q_5$$

$$= Q_0 + (\alpha Q_1 + \frac{1}{2}Q_2) + \frac{1}{2}Q_2 + (\alpha Q_3 + \frac{1}{2}Q_4) + \frac{1}{2}Q_4 + Q_5$$

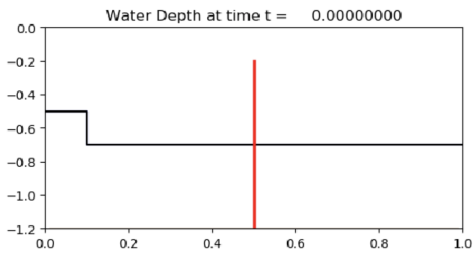
$$= \underbrace{Q_0 + \alpha Q_1 + Q_2 + \alpha Q_3 + Q_4 + Q_5}_{\text{post-SRD}}.$$

We show numerically that this 1D SRD method indeed does provide mass conservation. We

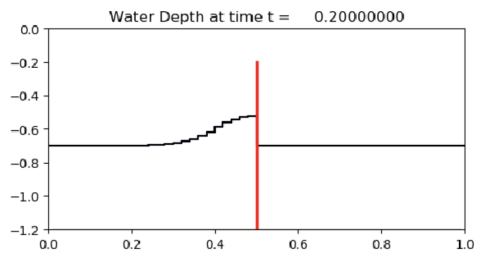
implement the method on a problem with barrier height ℓ that is set to 1.0, with water height h at 0.5, and a dam break height h_d of 0.7. The small cell ratio α is 0.01, located at $i = 25$. The water is contained in one side of the barrier due to the tall height of barrier (Fig. 6.4). We show the conservation of height calculated as

$$M_T = \sum_{i=1}^{50} \Delta x_i h_i,$$

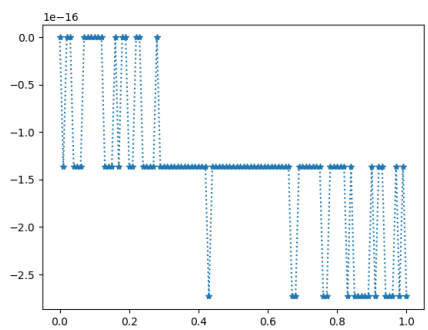
and one can see that the relative difference of total height oscillates around -1.5e-16.



(a) Initial



(b) Reflection



(c) Total mass changes across time.

Figure 6.4: 1D dam break blockage problem in SRD with 50-cell grid from [0,1] with $\alpha = 0.01$ and $\ell = 1.0$.

6.1.4 2D SRD method

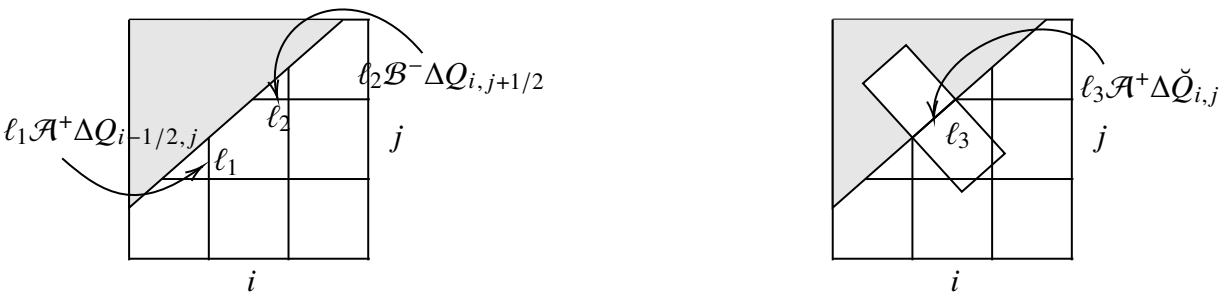
In this section we describe the original 2D SRD method applicable for impermeable solid boundaries using wave propagation and fluctuations. The method here will apply to our case with the permeable barrier interface, with the exception of the barrier edge handling.

The added complexity is now finding the conservative but unstable updates of the geometrically

specific cut cells during the first step of SRD method. There are 8 total types of cut cells (4 if we disregard the sign of the cut slope) as seen in Fig. 6.2. This means that we must update the cut cells in at most 4 different ways depending on their shapes.

Conservative but unstable update

The way we compute the conservative but unstable updates is by first identifying the cut cell edge lengths. Using the side lengths, we weight the waves arising from those edges as seen in Fig. 6.5. The waves from the vertical/horizontal edges are computed by using wave propagation method with the two adjacent cell averages.



(a) The weighted waves used to update the cut cell (i, j). The wave from the left edge is weighted by l_1 and the wave from the top edge is weighted by l_2.

(b) The normal averages used to produce waves at the barrier edge to update the cut cell (i, j). The wave from the barrier edge is weighted by l_3.

Figure 6.5: Ratio of edge lengths to mesh size as weights on waves from edges of a cut cell.

At the barrier edge, the original SRD method computes fluctuation waves by first creating normal "h-boxes" [7] extending towards both sides of the barrier edge as in Fig. 6.5b. The states of these h-boxes are computed by volume weighting the averages of the covered cells and by rotating their momentum with respect to the barrier direction, denoted by $\breve{Q}$ in Fig. 6.5b. The average of the h-box intruding into the solid region is computed by simply negating the velocity in the normal direction to the barrier (Fig. 6.5b). Since SWE is rotationally invariant, the waves at the barrier edge are calculated using the method described in Section 3.3 with these two h-box averages.

Once the cut cells are updated conservatively using the weighted waves at each edge, neighborhoods are found for each cell to compute the neighborhood averages for the SRD updates. The determination of overlap counts and small cell criteria ($< 0.5\Delta x\Delta y$) are all the same as in the 1D case, so we only show how neighborhoods are formed in the 2D case in Section 6.1.4.

Finding neighborhoods

The two possible types of neighborhood are the *normal* and the *grid neighborhood*. As can be seen in the left subfigure of Fig. 6.6, a normal neighborhood of cell (i, j) only uses the cell directly below the cut cell. Note that the neighboring cell could also be a cut cell. As long as the total area of the neighborhood exceeds $0.5\Delta x\Delta y$, it is a valid neighborhood. There are cases, however, when just including the directly normal cell does not result in a neighborhood of size bigger than $0.5\Delta x\Delta y$, as in the right subfigure in Fig. 6.6. In this case, the neighborhood of cell (i, j) needs not only cell $(i, j-1)$ but also the cells $(i-1, j-1)$ and $(i-1, j)$ to be included in their neighborhood. This type of neighborhood is used for cut cells that are especially constricted as in the figure and we will not use this type as our V barrier is obtuse.

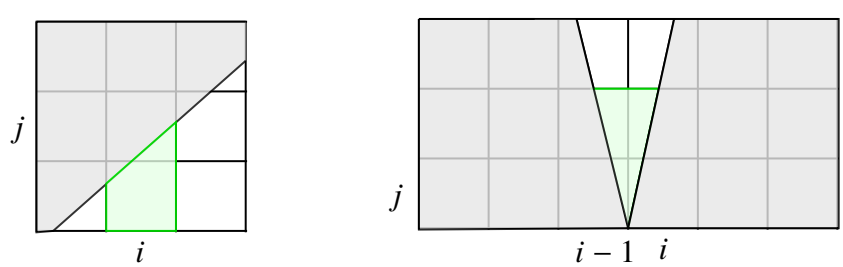


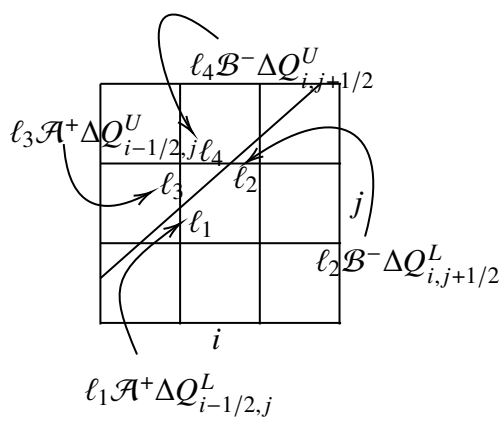
Figure 6.6: Two types of neighborhoods in 2D SRD. The solid region is made opaque for clarity in identifying the indices.

6.1.5 SRD applied to model problem

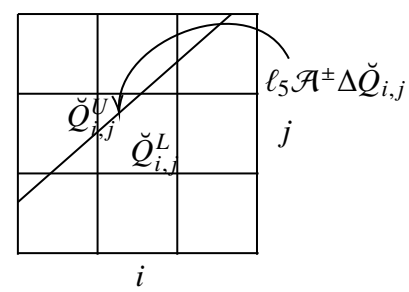
In our model problem the main adjustment we need to make to the original SRD method is handling the presence of the additional state on the other side of the barrier. We denote the upper cut cell by the superscript $Q^U_{i,j}$ and the lower cut cell by the superscript $Q^L_{i,j}$. As shown in Fig. 7.4,

we remark that aside from the barrier edge, the weighted wave propagation method as illustrated in Fig. 6.5a applies to both the upper cut cell and the lower cut cell at each non-barrier edge.

We now need to allow communication between the two sides of the cut through the barrier edge. Consequently, the key differences from the original SRD method are (1) the states used for the Riemann problem at the barrier edge and (2) calculation of fluctuation at the barrier edge, namely wave redistribution.



(a) The weighted waves used to update both cut cells $Q^U_{i,j}$ and $Q^L_{i,j}$ at the non-barrier edges. Weights l_i are lengths of the cut cell edges.



(b) The rotated averages $\breve{Q}^{U/L}_{i,j}$ of both cut cells are used to produce waves at the barrier edge to update the cut cells $Q^{U/L}_{i,j}$. The wave is computed using wave redistribution and weighted by barrier edge length l_5.

Figure 6.7: Weighted waves with edge lengths of a cut cell for both upper and lower cut cells.

Wave redistribution at barrier edge

For the Riemann problem at the barrier edge, we cannot simply use the h-box averages with negated normal momentum as done in the original SRD method due to the presence of state variables on either side. We must define a new Riemann problem to enable overtopping. In fact, we do not need to use the normal h-boxes at all to compute the fluctuation. Instead we can simply use the rotated state variables $\breve{Q}^U_{i,j}, \breve{Q}^L_{i,j}$ on either side of the barrier cut:

$$\breve{Q}^h_{i,j} = R_{i,j} Q^h_{i,j}, \tag{6.7}$$

where $h = L$ or U, and

$$R_{i,j} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \hat{n}_1 & \hat{n}_2 \\ 0 & \hat{t}_1 & \hat{t}_2 \end{bmatrix},$$

with rotation vectors $\hat{n}, \hat{t}$ being simply orthonormal pairs to the barrier. This simple usage of cut cell averages also avoids any geometrical complications at the V-tip, making SRD a better method for more complicated shaped barriers.

Once we have rotated the states, these two states become the left and right states of two ghost problems, as described in Chapter 4. The wave redistribution method then redistributes waves from the two ghost Riemann problems, giving us $\mathcal{A}^{\pm}\Delta\breve{Q}_{i,j}$. Following the algorithm in [27] we rotate them back:

$$\mathcal{A}^{\pm}\Delta Q_{i,j} = R_{i,j}^{T}\mathcal{A}^{\pm}\Delta\breve{Q}_{i,j}. \tag{6.8}$$

These waves are then weighted by the length of the barrier cut edge, as shown in Fig. 6.7b.

These weighted waves are then used to update the cut cells in a conservative but possibly unstable way. In general, this update will look like:

$$\begin{aligned} Q_{i,j}^{L,n+1} = Q_{i,j}^{L,n} - \frac{\Delta t}{\alpha_{i,j}^{L}}(&\ell_{i,j}\mathcal{A}^{+}\Delta Q_{i,j} \\ &+ \ell_{i-1/2,j}^{L}\mathcal{A}^{+}\Delta Q_{i-1/2,j}^{L} + \ell_{i+1/2,j}^{L}\mathcal{A}^{-}\Delta Q_{i+1/2,j}^{L} \\ &+ \ell_{i,j+1/2}^{L}\mathcal{B}^{-}\Delta Q_{i,j+1/2}^{L} + \ell_{i,j-1/2}^{L}\mathcal{B}^{+}\Delta Q_{i,j-1/2}^{L}), \end{aligned} \tag{6.9}$$

$$
\begin{aligned}
Q_{i,j}^{U,n+1} = Q_{i,j}^{U,n} - \frac{\Delta t}{\alpha_{i,j}^{U}} (&\ell_{i,j}\mathcal{A}^{-}\Delta Q_{i,j} \\
&+ \ell_{i-1/2,j}^{U}\mathcal{A}^{+}\Delta Q_{i-1/2,j}^{U} + \ell_{i+1/2,j}^{U}\mathcal{A}^{-}\Delta Q_{i+1/2,j}^{U} \\
&+ \ell_{i,j+1/2}^{U}\mathcal{B}^{-}\Delta Q_{i,j+1/2}^{U} + \ell_{i,j-1/2}^{U}\mathcal{B}^{+}\Delta Q_{i,j-1/2}^{U}),
\end{aligned} \tag{6.10}
$$

where $\alpha_{i,j}^{U/L}$ is the area of cut cell, $\ell_{i,j}$ the length of the barrier edge, and $\ell_{i\pm1/2,j}^{U/L}$ and $\ell_{i,j\pm1/2}^{U/L}$ represent the lengths of the vertical and horizontal edges of the cut cell, respectively. Note that there are five terms as the maximum number of cut cell edges is five, but some of them will drop depending on the shape of the cut cell (e.g. for upper cut cell in Fig. 6.7, we have $\ell_{i-1/2,j}^{U} = \ell_3$, $\ell_{i+1/2,j}^{U} = 0$, $\ell_{i,j+1/2}^{U} = \ell_4$, $\ell_{i,j-1/2}^{U} = 0$).

Neighborhood and overlap count

Once we have updated each cut cell as described above, we then find the neighborhoods for each cut cell. In our model problems, finding the neighborhood for the cut cells is an easier task, as we can use normal neighborhoods for all of them.

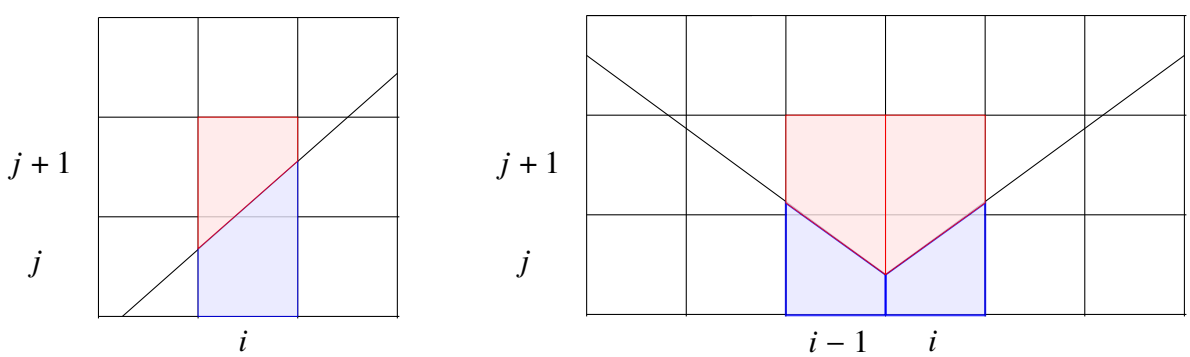


Figure 6.8: Normal neighborhoods sufficient for model problems. On the left, we show upper neighborhood (in red) for cell (i, j) and lower neighborhood (in blue) for cell $(i, j+1)$ and on the right, we show upper neighborhoods (red) for cells $(i-1, j)$ and (i, j) and lower neighborhoods (blue) for cells $(i-1, j+1)$ and $(i, j+1)$.

As shown in Fig. 6.8, for each small cell, we choose the cell directly above or below as this will suffice to produce a neighborhood whose area is greater than $0.5\Delta x\Delta y$. For every cut cell produced by either the linear or V barrier, there will always be a cell above (for upper cut cell) or below

(for lower cut cell) whose $\alpha \geq 0.5$. This ensures using normal neighborhood for both the upper and lower half of the cut to be valid. Note that the obtuse angle of the V barrier enables us to use normal neighborhoods extensively. Finding the neighborhood for the V barrier essentially becomes a double copy of the linear problem. Note also that we must only take one sided neighborhoods to avoid mixing states across barrier, as alluded to in the 1D section.

The overlap count also becomes simplified in both model problems when we use the normal neighborhoods. For small cut cells on either side, we have an overlap count of 1, as no other cut cell uses them for neighborhood formation. For the normal neighboring cells above or below small cut cells, we have an overlap count of 2, as they have $\alpha \geq 0.5$ and are their own neighborhoods, in addition to being a neighbor to the small cut cells. If a cut cell has $\alpha \geq 0.5$, then it will automatically be its own neighborhood and have overlap count of 1. Thus the overlap counts alternate between 1 and 2, depending on whether the cell is a small cut cell, non-small cut cell, or a neighbor of a small cut cell.

These overlap counts and the areas of the neighborhood cells then form neighborhood averages $N_{i,j}$:

$$N_{i,j}^{U/L} = \left(\frac{1}{\alpha_{i,j}^{U/L} + \frac{\alpha_{i,j\pm1}^{U/L}}{2}} \right) \left(\alpha_{i,j}^{U/L} Q_{i,j}^{U/L} + \frac{\alpha_{i,j\pm1}^{U/L}}{2} Q_{i,j\pm1}^{U/L} \right), \tag{6.11}$$

for small cut cells (i, j), where $Q_{i,j}^{U/L}$ represent the unstable but conservative update (the superscript $n+1$ is omitted for clarity and reserved for final update). For all other cells, we have $N_{i,j}^{U/L} = Q_{i,j}^{U/L}$.

All in all, we have for our update formula for all small cut cells (i, j):

$$Q_{i,j}^{n+1,U/L} = \begin{cases} N_{i,j}^{U/L} & \text{if overlap count} = 1 \\ \frac{1}{2}(N_{i,j}^{U/L} + N_{i,j\pm1}^{U/L}) & \text{if overlap count} = 2 \end{cases}. \tag{6.12}$$

6.2 Model problems

All the model problems presented below have steady water height of $h = 1.2$ and a dam jump of $\Delta h = 0.8$ or 1.5, giving the overall height of the dam break to be $h_d = 2.0$ or 2.7. The barrier heights ℓ are chosen to be either $\ell = 1.5$ or $\ell = 5.0$. These are chosen to test both (1) complete reflection of incoming wave by the barrier ($\Delta h = 1.5$, $\ell = 5.0$) and (2) overtopping of wave over the barrier ($\Delta h = 0.8$, $\ell = 1.5$). The boundary conditions for the problems are wall boundaries everywhere except for extrapolation condition on the overtopped side in the overtopping examples.

For comparison, we run two simulations. First is with the same initial condition in GEOCLAW , except with the barriers being single cell wide bathymetric jumps. We note that GEOCLAW results are compared to ensure physical reliability of our results and that only similarity in quality of the numerical solution is sought after. Our problem is the numerical limit that the barrier thickness approaches zero, but it is difficult to attain reliable numerical solution using GEOCLAW by thinning out the bathymetric jump. Therefore, we compare our simulations against the second, true numerical solutions performed in the mapped grid examples, where we use grid transformations and the wave redistribution method on a computational grid edge mapped to the barrier on the physical grid. We compare the 2D contour plots of water height at specific times from both the SRD cut cell method and GEOCLAW /mapped examples and the gauge data, which are height measurements at a specified location in the domain (asterisked), in order to perform convergence analysis.

6.2.1 The 20° angled barrier

Here we present the first numerical example with a positively sloped 20° barrier, with a 150× 150 grid. We first show a sufficiently high barrier example that prohibits overtopping of an incoming wave and then a lower barrier that permits overtopping (Fig. 6.9). In the reflection example, we place our gauge at $(0.5, 0.39)$ just off the barrier, to capture the reflection close to the wall. In the overtopping example, we place two gauges, one at $(0.5, 0.39)$ as before and another at $(0.5, 0.8)$ to both capture the reflection and also the overtopped wave further away from the barrier.

Case of Reflection Only

We test a dam break problem to simulate the incidence of oblique waves upon the barrier.

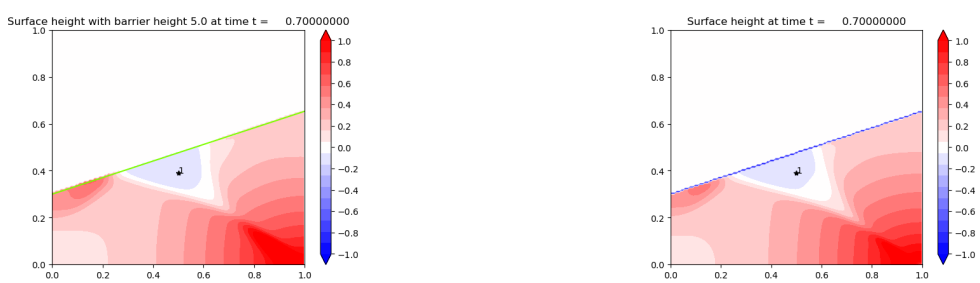
(a) Reflection only case: $\ell = 5.0$, $h_d = 2.7$. (b) Overtopping case: $\ell = 1.5$, $h_d = 2.0$.

Figure 6.9: Initial condition for two problems. Both on 150×150 grid.

In order to highlight the wave reflections we amplify the dam break size as well to 1.5. We observe very good comparison at both time snapshots (Fig. 6.10 and Fig. 6.11).

(a) SRD (b) GEOCLAW

Figure 6.10: Comparison of SRD with GEOCLAW at time $t = 0.3$



(a) SRD (b) GEOCLAW

Figure 6.11: Comparison of SRD with GEOCLAW at time $t = 0.7$.

At time $t = 0.3$, the incoming wave is gliding up along the barrier while being reflected back in direction normal to the barrier. One can observe how the wave front on the right has not yet reached

the barrier. By time $t = 0.7$, the reflected wave has bounced off the wall boundary condition on the bottom and reflected back onto the barrier, repeating the gliding motion as seen on the lower left region of the barrier. We can see that the gauge heights for all time are also in good agreement between the two methods (Fig. 6.12).

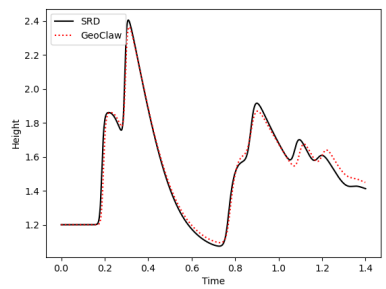


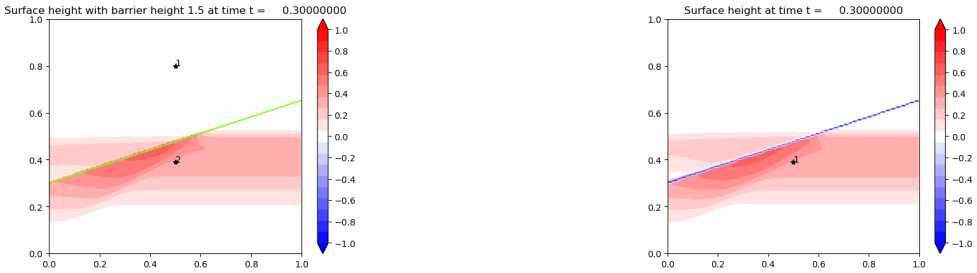
Figure 6.12: Time profile at Gauge 1 $(0.5, 0.39)$ for complete blockage example with slanted barrier. A slight phase error is shown towards the latter time period, which may be due to slight difference in the placement of barriers in the two examples.

Case of Overtopping

Next we test overtopping using the same slanted barrier. This time we take $\Delta h = 0.8$ and allow the the top edge of the domain to have extrapolation boundary where we let waves exit the domain.

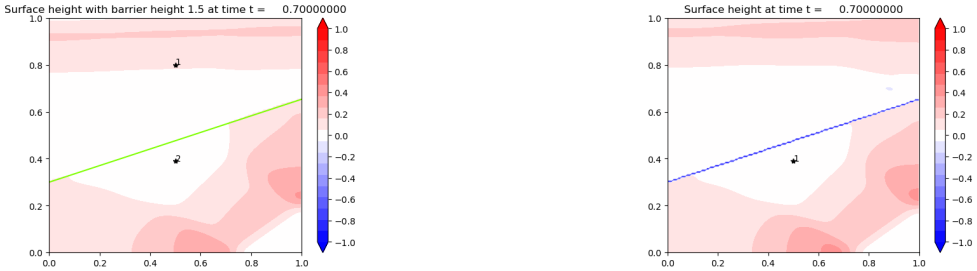
We observe from both SRD results and GEOCLAW results that the wave is abated from the barrier and proceeds in the same direction after overtopping (Fig. 7.7). The reflection shown at time $t = 0.7$ on the lower side of the barrier is similar to the reflective behavior observed already in the complete blockage example (Fig. 7.8). An interesting observation to be made is at $t = 1.4$, the gliding waves 'pinch up' to achieve overtopping momentum on the right, whereas it does not overtop on the left part of the barrier.

The similarity in behavior (Fig. 7.7-Fig. 7.9) and also magnitude of waves as seen in Fig. 7.10 assure us of the physicality of the SRD results. We note that there is characteristic difference in the overtopped wave profile Fig. 6.16a. The GEOCLAW example has a lower dip at the end of the first wave than the SRD. We attribute this to the presence of the bathymetric cell jump in the GEOCLAW example causing a rarefaction at the "behind" interface of the jump (i.e. on the side



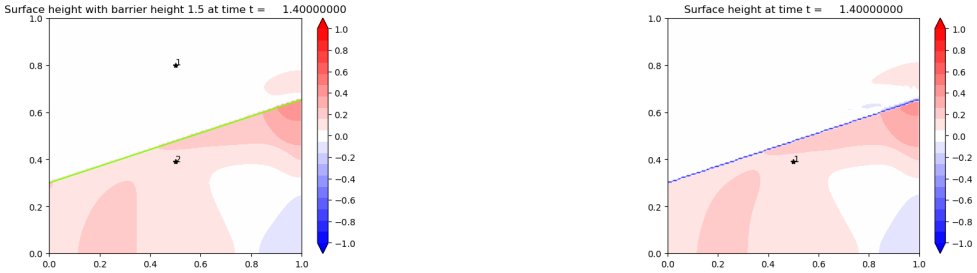
(a) SRD (b) GEOCLAW

Figure 6.13: Comparison of SRD with GEOCLAW at $t = 0.3$.



(a) SRD (b) GEOCLAW

Figure 6.14: Comparison of SRD with GEOCLAW at $t = 0.7$.



(a) SRD (b) GEOCLAW

Figure 6.15: Comparison of SRD with GEOCLAW at $t = 1.4$.

of the overtopping) due to the large disparity in the heights [33]. Also the peaks are slightly off-centered due to the nonzero thickness of barrier causing earlier overtopping for GEOCLAW . The reflected wave shown in Fig. 6.16b, however, is virtually identical.

(a) Time profile of Gauge 1 (0.5,0.8).

(b) Time profile of Gauge 2 (0.5,0.39).

Figure 6.16: Gauge profiles compared with GEOCLAW results. Note that there is a discrepancy in the overtopped gauge (left), as water feels the big bathymetric variation in the GEOCLAW case.

Comparison to mapped grid

We validate the SRD results against a mapped grid example. We transform the computational uniform grid (x, y) into a skewed grid $f_L(x, y)$ (Fig. 6.17):

$$f_L(x, y) = (x, \mu_L(y)), \tag{6.13}$$

$$\mu_L(y) = \begin{cases} \frac{L(x)}{y^*} y & \text{if } y \in [0, y^*] \\ \frac{1-L(x)}{1-y^*}(y-1) + 1 & \text{if } y \in [y^*, 1] \end{cases}, \tag{6.14}$$

where $L(x)$ is the barrier line equation and y^* is the computational y-edge mapped to the barrier edge (lime green in Fig. 6.17), with $y^* = \frac{y_1+y_2}{2}$ with the subscripts denoting vertex numbers as denoted in Fig. 2.2a. At $y = y^*$ wave redistribution is applied.

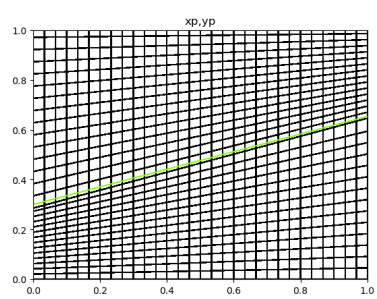
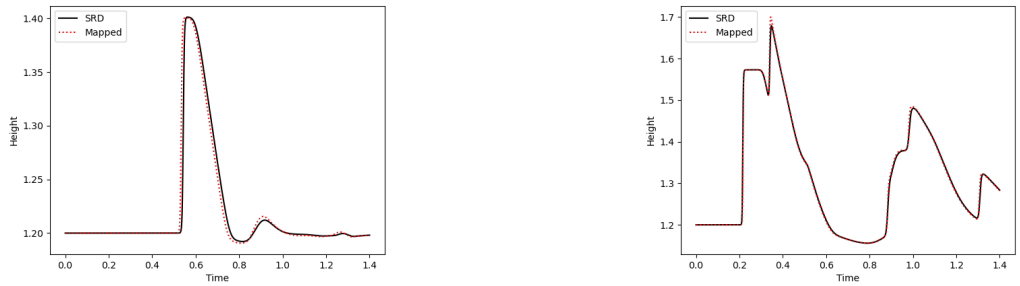


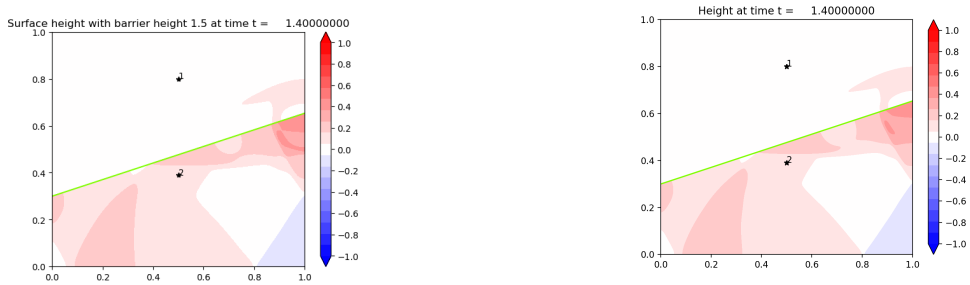
Figure 6.17: Mapped grid for the 20° barrier. Coarsened to 30×30 to highlight mapping.

In Figs. 6.18 and 6.19, we observe the gauge results and contours plot at $t = 1.4$ between



(a) Time profile of Gauge 1 $(0.5, 0.8)$. (b) Time profile of Gauge 2 $(0.5, 0.39)$.

Figure 6.18: Gauge comparisons between SRD and mapped grid. Both on 900×900 grid.



(a) SRD (b) Mapped grid

Figure 6.19: Comparison between SRD (900×900) and mapped grid (900×900) at $t = 1.4$.

the SRD and the mapped results both on 900×900 grid and see the disappearance of the earlier differences in the wave shape.

Convergence

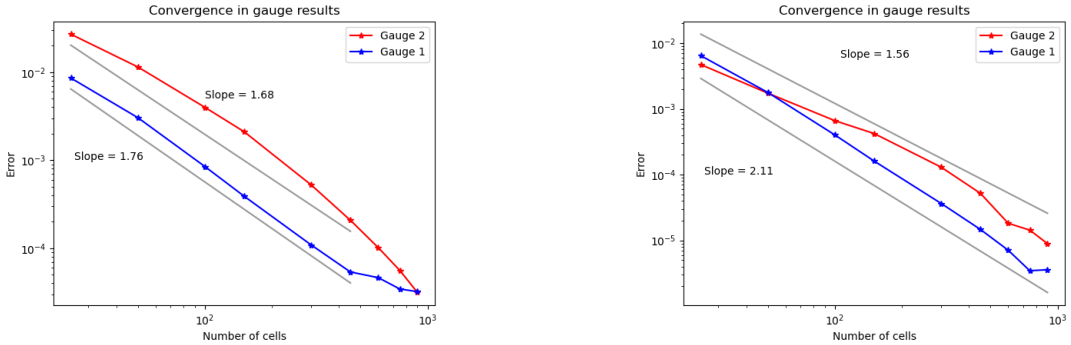
For convergence studies we observe the results at gauge point against the mapped grid (900×900) result, as would be of interest in storm simulations. Also we only study the convergence of the overtopped examples, for similar reasons as above mentioned, namely, that in realistic scenarios the barriers will be overtopped by incoming waves.

The convergence is shown in both Table 6.1 and Fig. 6.20. We observe a convergence order of approximately 1.7 for both gauge points in the L_1 norm and somewhere in between 1.5 and 2.1 in the L_∞ norm. We note that gauge 2 is closer to the barrier and yet the order of convergence is very similar to that for gauge 1, which is further away from the barrier. This order of convergence is

Δx	N_x, N_y	L_1 Error		L_∞ Error	
		Gauge 1	Gauge 2	Gauge 1	Gauge 2
4.e-2	25	8.58e-3	2.70e-2	6.46e-03	4.72e-03
2.e-2	50	3.04e-3	1.14e-2	1.77e-03	1.74e-03
1.e-2	100	8.46e-4	3.98e-3	4.02e-04	6.62e-04
0.666e-2	150	3.89e-4	2.10e-3	1.61e-04	4.22e-04
0.333e-2	300	1.08e-4	5.23e-4	3.65e-05	1.30e-04
0.222e-2	450	5.37e-5	2.08e-4	1.47e-05	5.20e-05
0.1666-2	600	4.63e-5	1.01e-4	7.08e-06	1.82e-05
0.1333-2	750	3.44e-5	5.59e-5	3.45e-06	1.43e-05
0.1111-2	900	3.21e-5	3.14e-5	3.58e-06	8.92e-06

Table 6.1: L_1 and L_∞ errors computed against mapped grid results at gauge point 1 (0.5,0.8) and 2 (0.5, 0.39).

surprising, given that we are not doing any gradient reconstruction at the cut cells but only using piecewise constant values [43]. This is most likely due to use of transverse solvers away from the barrier [10]. And a surprising note to be made is that in the L_∞ norm there is almost second order convergence in gauge 1, which looks promising for SRD method in its capability to model overtopping waves.



(a) L_1 error order around 1.7.

(b) L_∞ error order around 1.5 to 2.1.

Figure 6.20: Convergence plots with mapped grid 900×900 for 20° problem.

We also do convergence studies with the GEOCLAW example by comparing our SRD results against GEOCLAW example on 1200×1200 grid as our reference solution and the solution converges with greater than one order of convergence around 1.3 as seen in Fig. 6.21.

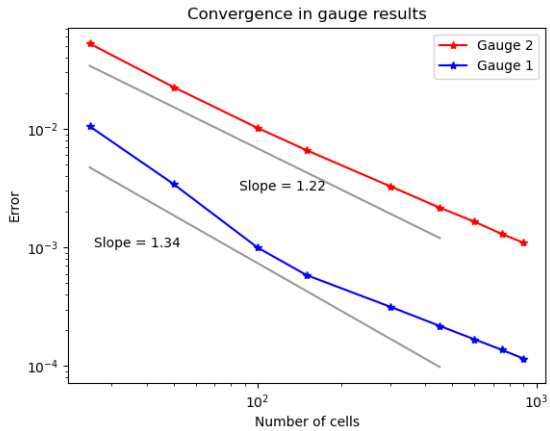


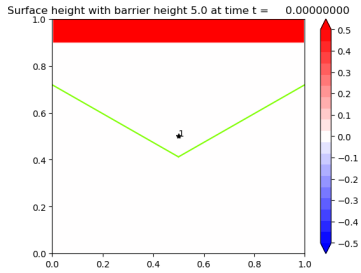
Figure 6.21: L_1 convergence of gauge profiles compared to GEOCLAW results on 1200×1200 grid (right) Order of 1.34 is observed at gauge 1 (further from barrier) and 1.22 observed at gauge 2 (closer to barrier).

6.2.2 117° angled V barrier

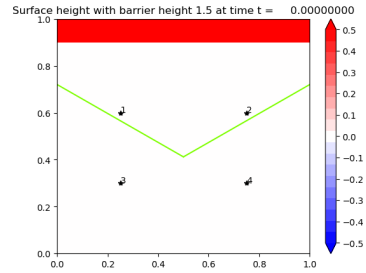
The inspiration for the V barrier problem comes from the Maeslant Barrier in the Netherlands (Fig. 6.22), which closes to form a curved V-shape.

Figure 6.22: The Maeslant Barrier in the Netherlands. The barrier can approximately be represented as a V-shape (highlighted in red).

We also note that the concave part of the Maeslant barrier is where the incoming wave is expected to hit (i.e. V faces towards the sea). Thus we model our problem such that a dam break initiates a wave from the top side of the V barrier (Fig. 6.23). Also, in [7], a similar example is solved for the Euler equations, but with no flux allowed at the V boundary and with computations only for shock wave reflection.



(a) Reflection only case: $\ell = 5.0$, $h_d = 2.7$.

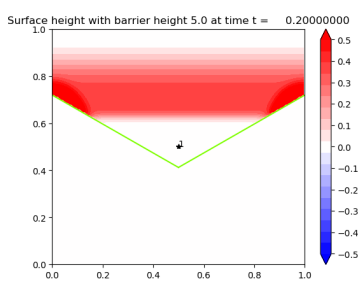


(b) Overtopping case: $\ell = 1.5$, $h_d = 2.0$.

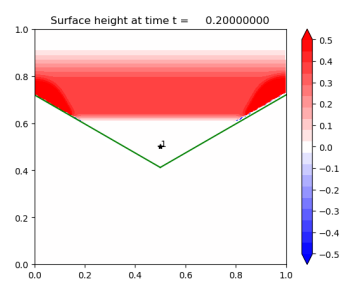
Figure 6.23: Initial conditions for two problems. Reflection: 150×150, overtopping: 300×300 grid.

Case of Reflection Only

Fig. 6.24 to Fig. 6.25 show an example with 150×150 grid in a domain of $[0, 1] \times [0, 1]$ for both SRD and GEOCLAW and show a nice containment of the water behind the barrier. At Fig. 6.24, we can see the incoming wave gliding across each side of the V barrier towards the center. At Fig. 6.25, the gliding waves have crossed each other and are spreading radially outward away from the center of the barrier, leaving a dip (in blue). In Fig. 6.26, we show the wave profile at gauge point 1. We observe two waves: first initial wave that is reflected at the center and second wave reflected from the side boundary conditions. The second wave is lower in amplitude as it has lost some of its momentum.

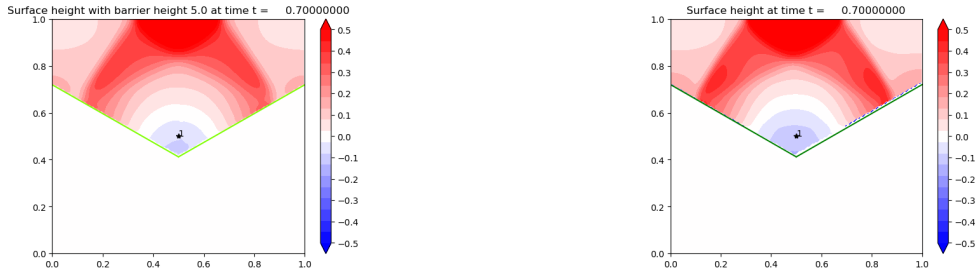


(a) SRD



(b) GEOCLAW

Figure 6.24: Solution at $t = 0.2$. Because the barrier does not allow overtopping, the two problems are numerical the same.



(a) SRD (b) GEOCLAW

Figure 6.25: Solution at $t = 0.7$. The gliding reflected waves have crossed each other in opposite directions.

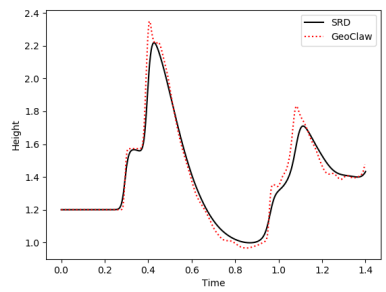
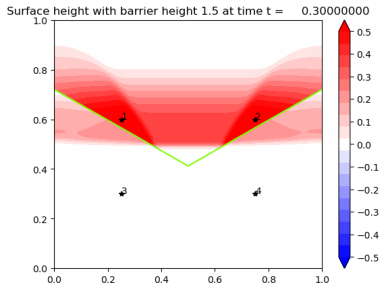


Figure 6.26: Time profile at Gauge 1 (0.5,0.5). Slight difference in the peaks are likely due to the diffusive nature of SRD, introduced by the neighborhood averaging.

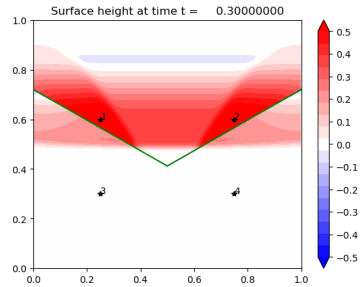
Case of Overtopping

We increase the resolution to 300×300 for clearer results and first do a comparison against GEOCLAW results to show the physicality of our results. From Fig. 6.27 to Fig. 6.29, we can see the qualitative similarities of the two 2D plots. At time $t = 0.3$, we see the reflecting wave's moving into the center (where the amplitude reaches peak in gauge 1/2 as in Fig. 6.30a) and the overtopping wave's "wing"-like structure just below the V barrier (Fig. 6.27). In Fig. 6.28, we see the overtopping wave moving radially outward from the center, more captured in the SRD results than GEOCLAW. Finally in Fig. 6.29 we see the small island of wave amplitude at the bottom center of the plot in both results.

We place our gauges at either side of the V barrier $(0.25, 0.3)$, $(0.75, 0.3)$, $(0.25, 0.6)$, $(0.75, 0.6)$ in order to test for symmetry in the results Fig. 6.30. We observe the major peaks lining up between the two results of SRD and GEOCLAW. Again we see slightly earlier peak and deeper dip in the

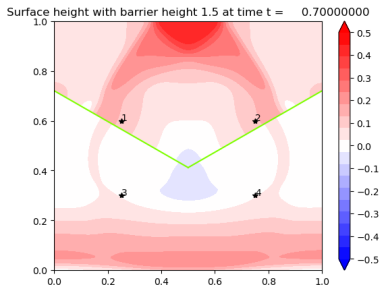


(a) SRD

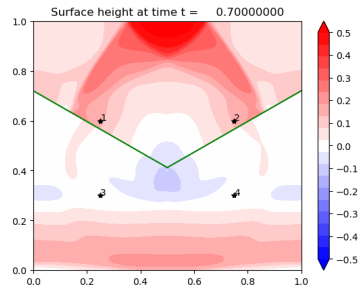


(b) GEOCLAW

Figure 6.27: Solution at $t = 0.3$. Both 300×300 grid. Note the structure of the just overtopped wave.

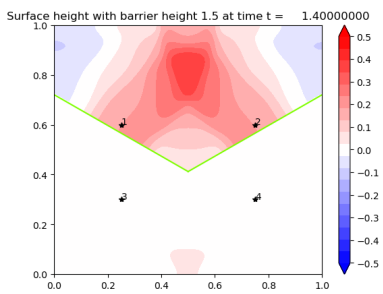


(a) SRD

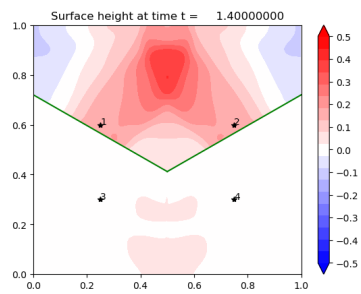


(b) GEOCLAW

Figure 6.28: Solution at $t = 0.7$. Note the radially outward moving wave from the center of the V barrier.



(a) SRD



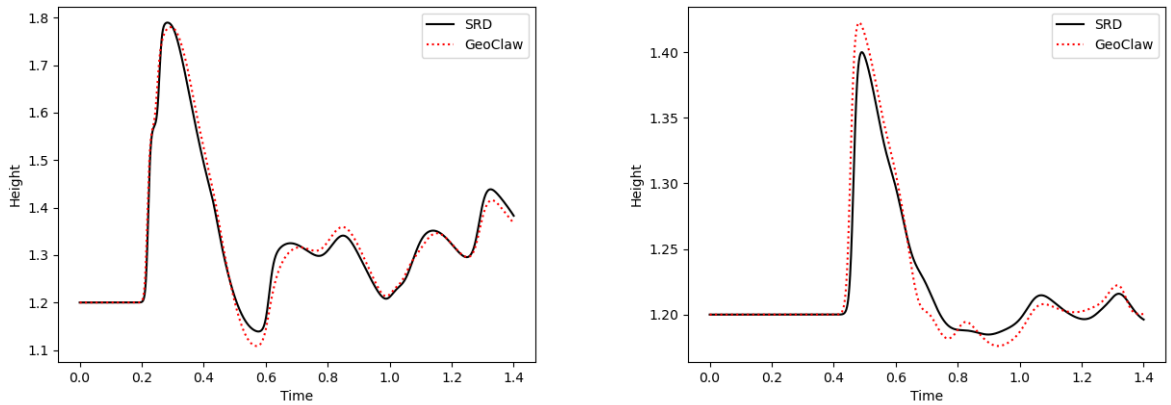
(b) GEOCLAW

Figure 6.29: Solution at $t = 1.4$. Note the "island" of peak at the bottom center.

overtopped waves, due to the presence of physical barrier.

Computational Superiority to Refinement using GEOCLAW

To highlight the superiority of using the SRD method on the zero width barrier over using barrier refinement, we compare the computational costs of a V barrier simulation on GEOCLAW with



(a) Time profile of Gauge 1,2 $(0.25(0.75), 0.6)$.

(b) Time profile of Gauge 3,4 $(0.25(0.75), 0.3)$.

Figure 6.30: Gauge profiles compared with GEOCLAW results: 300×300 grid.

adaptive (double) refinement at the barrier and SRD with same resolution. In reality the refinement level required at barriers will be greater than 2 as barriers are much skinnier than surrounding bathymetric surfaces. (The barrier in these GEOCLAW runs is two cells wide, to get down to single cell width in the adaptive mesh refinement.)

We show results from using two resolutions $\Delta x = 1/300$, $1/450$. These already show the computational benefit we derive from our proposed method. We compare the Δt's and number of steps taken from both SRD and GEOCLAW simulations. For $\Delta x = 1/300$ we observe that we get tenfold increase in the minimum Δt (from 8.6e-06 to 9.6e-05) and 320 % increase in the average Δt (from 0.00028 to 0.0012) and about fivefold decrease in the number of steps taken (9958 steps to 1850 steps). For $\Delta x = 1/450$ we observe about 70-fold increase in the minimum Δt (from 2.9e-06 to 2.2e-04) and 340 % increase in average Δt (from 1.7e-4 to 7.7e-4) and 5.5 times reduction in number of steps taken (15861 to 2843 steps).

6.2.3 Comparison to mapped grid

The mapped grid is shown in Fig. 6.31. Here we transform the computational uniform grid into

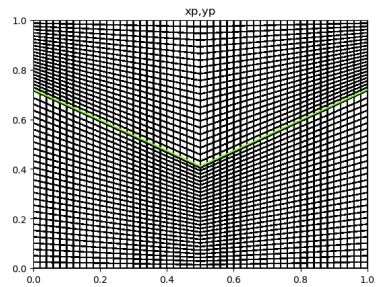


Figure 6.31: Mapped grid for the V barrier. Coarsened to 50×50 to highlight mapping.

a physical V shaped grid akin to what is done in [7], with the following mapping f:

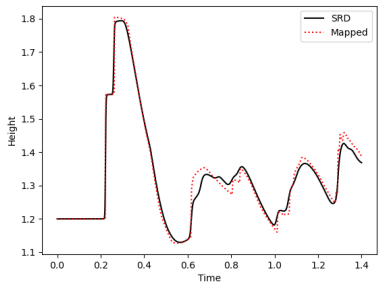
$$f_V(x, y) = (x, \mu_L(y)), \tag{6.15}$$

$$\mu_V(y) = \begin{cases} \frac{L_1(x)}{y^*} y, & (x, y) \in [0, 0.5] \times [0, y^*] \\ \frac{1-L_1(x)}{1-y^*}(y-1)+1, & [0, 0.5] \times [y^*, 1] \\ \frac{L_2(x)}{y^*} y, & [0.5, 1] \times [0, y^*] \\ \frac{1-L_2(x)}{1-y^*}(y-1)+1, & [0.5, 1] \times [y^*, 1] \end{cases}, \tag{6.16}$$

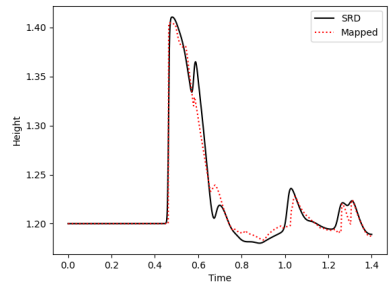
where y^* is the computational barrier edge taken to be $y^* = 0.5(y_1 + y_2)$ (Fig. 2.2b), and $L_1(x)$ is the barrier line from coordinate 1 to 2 and $L_2(x)$ that from coordinate 2 to 3. We apply wave redistribution (Section 6.1.5) at y^*.

Convergence

In Fig. 6.32 we plot the gauge results of 1250×1250 mapped grid V barrier example and 1050×1050 SRD example. We do see that the SRD results contain more fine movements of the wave and that the mapped grid example produces more smooth wave patterns. Overall, however, we see convergence as shown in Table 6.2 and Fig. 6.34 (for gauge points 1 through 4). The order of convergence are somewhere around 1.6 for both gauge points (Fig. 6.34). Again we observe greater-than-one order of convergence despite using piecewise constant approximations. However, unlike in our linear barrier example, the L_∞ norm convergence is comparable to L_1 convergence. Nonetheless, both gauges show first order convergence.



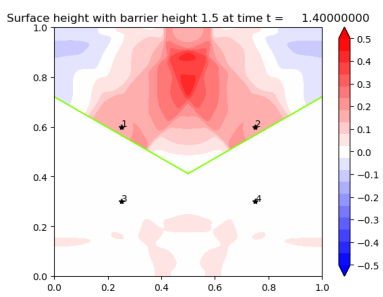
(a) Time profile of Gauge 1,2 $(0.25(0.75), 0.6)$.



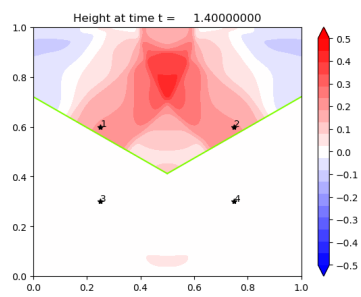
(b) Time profile of Gauge 3,4 $(0.25(0.75), 0.3)$.

Figure 6.32: Gauge profiles compared with mapped grid results: 1050×1050 for SRD and 1250×1250 for mapped grid.

Finally, in Fig. 6.33 we show two 2D plots comparing the SRD results on 1050×1050 grid and mapped grid results on 1250×1250 grid at $t = 1.4$. We see that SRD is less diffusive even on a slightly lower resolution that the mapped grid example. This can be seen in the finer details in both the reflected side (where the reflected waves cross in the center) and also the overtopped side, in the appearance of more residual overtopped waves.



(a) SRD results at $t = 1.4$.



(b) Mapped grid results at $t = 1.4$.

Figure 6.33: Comparison between SRD (1050×1050) and mapped grid (1250×1250) results.

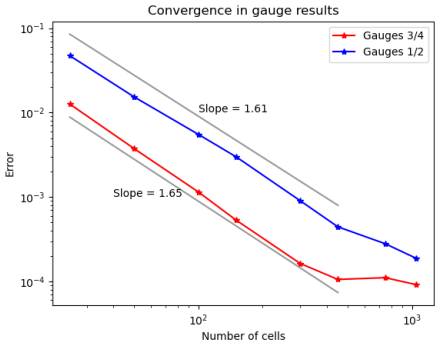
We also compare our V barrier SRD results with the finite width thin barrier simulation on GEOCLAW, refined to 1200×1200 grid, i.e. $\Delta x \approx 0.0009$ thick barrier and observe a convergence order of 1.3 as shown in Fig. 6.35.

6.2.4 Conservation

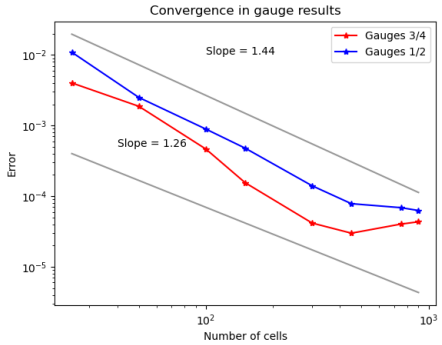
Finally, we show numerical conservation of our method by doing a V barrier example that contains water on one side of the barrier as in Fig. 6.24. This time we measure the conservation of

Δx	N_x, N_y	L_1 Error		L_∞ Error	
		Gauge 1,2	Gauge 3,4	Gauge 1,2	Gauge 3,4
4.e-2	25	4.72e-2	1.26e-2	1.09e-2	4.0e-3
2.e-2	50	1.53e-2	3.75e-3	2.50e-3	1.87e-3
1.e-2	100	5.52e-3	1.14e-3	8.90e-4	4.64e-4
.666e-2	150	3.00e-3	5.34e-4	4.78e-4	1.55e-4
.333e-2	300	9.04e-4	1.63e-4	1.40e-4	4.17e-5
.222e-2	450	4.44e-4	1.05e-4	7.84e-5	3.00e-5
.1333-2	750	2.80e-4	1.10e-4	6.89e-5	4.03e-5
.1111-2	1050	1.87e-4	9.16e-5	6.28e-5	4.34e-5

Table 6.2: L_1 and L_∞ errors computed as difference between mapped and SRD at gauge points 1,2 located at (0.25,0.6) and (0.75,0.6) and points 3,4 located at (0.25,0.3) and (0.75,0.3).



(a) L_1 error order around 1.6.



(b) L_∞ error order around 1.2 to 1.4.

Figure 6.34: Convergence plots of SRD against mapped grid (1250×1250). The kink at the end is due to the similarity in resolution between the reference and the cut cell solution.

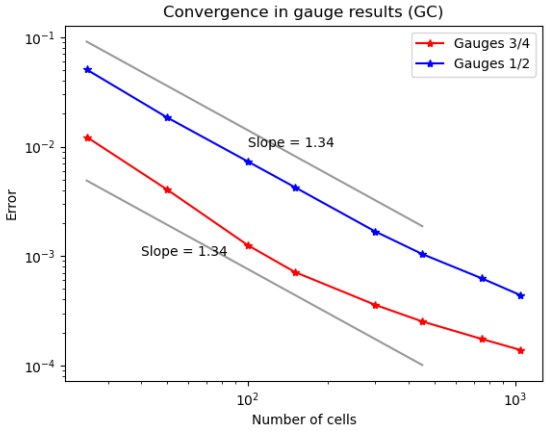


Figure 6.35: L_1 convergence of gauge profiles compared to GEOCLAW results on 1200×1200 grid (right) Order of around 1.34 is observed.

momentum:

$$\mu_T = \sum_{i,j} \alpha_{i,j} Q_x(i,j),$$

where $Q_x(i,j)$ denotes the x-momentum at cell $Q_{i,j}$ and $\alpha_{i,j}$ denote the areas of cell $Q_{i,j}$.

Note that there is a data structural aspect to calculating total momentum mass in our 2D problems. Because we keep two arrays of solutions, one array containing $Q^L_{i,j}$ and another containing $Q^R_{i,j}$, we need to bookeep which array to use to calculate the total mass. This is in addition to the fact that the array of $\alpha_{i,j}$ for the cut cells is not ordered in the same way as the Q arrays. Furthermore, there is a computational geometric aspect to the conservation calculation as well. Because our algorithm for cut cells' geometric calculation is accurate to 1e-6, our conservation accuracy will also bottom out at 1e-6 (comparable to magnitudes reported in literature [44]). To work around this, we solve the dam break reflection only problem as Fig. 6.24 on smaller scale problem with 6×6 grid, where we exactly know the cut cell areas. Also, since our problem starts with $\mu_T = 0$, we get the *relative* total difference by dividing by the average of the numerical total momentum $\bar{\mu}_T$. We see that the momentum difference fluctuates around -4e-15 (Fig. 6.36).

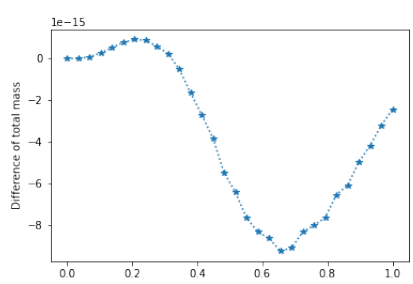


Figure 6.36: Relative momentum difference across time

Chapter 7: Cell Merging

Cell merging is the simplest out of all the methods discussed to deal with small cut cells. It simply combines the cut cell with a nearby larger cell, to result in a cell that satisfies $V_i > 0.5\Delta x\Delta y$. Then the method treats the combined cell as a single unit, and updates the cell, using this final update average to update both the small component cell and larger component cell.

7.1 Numerical method on cut cells: cell merging

Again, as seen in Fig. 6.2 we consider the same types of cut cells, where the cut happens only once in a grid cell. In this section, first we discuss how the cell merging method works. Then we describe how the fluctuation at the barrier cut edge is computed in both first and second order. Then we discuss how the rest of the cut cell edges are computed in both first and second order.

7.1.1 Cell merging

Cell merging works by absorbing a small cell into a larger neighboring cell, and considering the larger combined cell as its own cell. This way, the CFL limitation on the small cell is avoided, as the combined cell will be large enough to take a full time step with the numerical update.

The first step in the cell merging method is to identify which cells need merging. We set the area threshold to be at $0.5\Delta x\Delta y$, such that any cell whose area is less than this threshold is a cell that needs to be merged. Second, we need to identify the neighboring cell with which to merge the small cell. In all cases, the neighboring cell will be *the normal* neighboring cell. For our problems, this means the cell directly above the small cell (for small cells above barrier) or directly below the small cell (for small cells below the barrier) as shown in Fig. 7.1. The merged cell will then have a volume weighted average $\bar{Q}$ of the comprising cells Q_1 and Q_2, or $\bar{Q} = V_1Q_1 + V_2Q_2$.

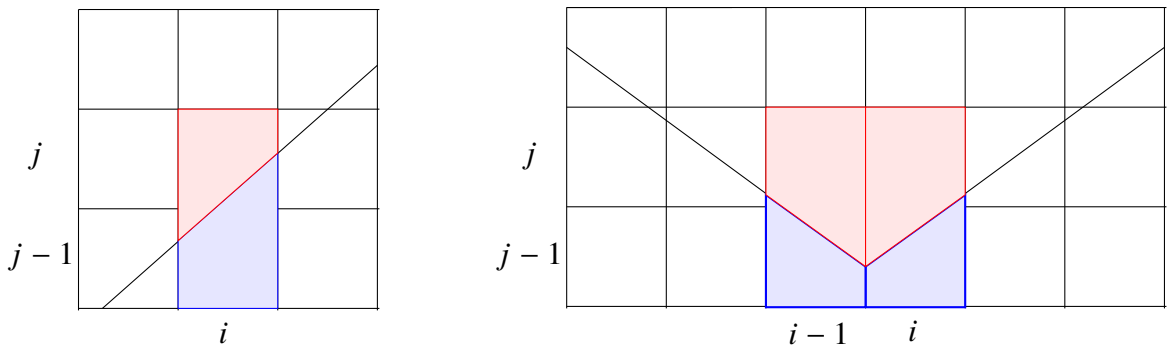


Figure 7.1: On the left, we show upper merged cell for cell $(i, j-1)$ and lower merged cell for cell (i, j) and on the right, we show the same for V barrier example for cells $(i-1, j-1)$ and $(i, j-1)$ and for cells $(i-1, j)$ and (i, j). Note how the shared edge does not show to highlight merging.

The third step is to update the merged cell. We adopt the style of using every outer edge as done in [45] to update our merged cells. This not only gives a conservative method but gives greater accuracy as we do not average the fluctuation over a larger combined edge of a merged cell. The only edge that does not play a role in updating the merged cell is the edge between the merging cells, as the merging essentially makes the two into one.

We describe how to compute the fluctuation at each edge of the cut cells in the next subsections. Note that if we use a first order fluctuation at each edge, we get a first order cell merging method. If we use a second order fluctuation at each edge, we get a second order cell merging method. That is, cell merging is simply a way to circumvent the CFL restriction and relies on the underlying computational method for fluctuation calculations to become either a low resolution method or a high one.

Final step is to use the updated merged cell to update both the small cell and the merged neighboring cell. If Q_M^{n+1} is the updated merged cell average resulting from merging cells (i, j) and $(i, j+1)$, then we have $Q_{i,j}^{n+1} = Q_M^{n+1} = Q_{i,j+1}^{n+1}$. This is also because the cell merging method considers the merged cell as one big cell.

7.1.2 At barrier edges

Unlike SRD, cell merging is a co-processing method that requires calculations during the numerical update. In particular, the fluctuations at each edge of the merged cell need to be computed

in order to update the merged cell. First we discuss how the fluctuation is computed at the barrier edge.

First order

To compute the fluctuation at the barrier edges, we use cell averages on either side of the cut $Q_{i,j}^{L/U}$ and apply the rotated WR.

These fluctuation waves are then weighted by the length of the barrier cut edge, as shown in Fig. 7.2. Note that as shown in Fig. 7.2, we may need to compute fluctuations at two barrier edges to update a single merged cell because we use every piecewise edge of merged cells.

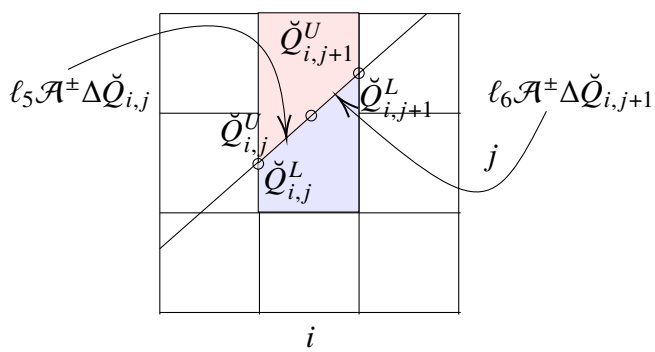


Figure 7.2: The rotated averages $\breve{Q}_{i,j}^{U/L}$ and $\breve{Q}_{i,j+1}^{U/L}$ are used to produce waves at the barrier edges of cut cell (i, j) and cell $(i, j+1)$ (with lengths ℓ_5, ℓ_6) to update the merged cells in blue and red. The red merging is for upper cell (i, j) and blue for lower cell $(i, j+1)$.

Second order

To perform second order methods, we also need to compute gradients $\nabla Q_{i,j}$ to do linear approximations. We use the gradients to approximate the solution values at cell edge midpoints at the barrier. In order to compute $\nabla Q_{i,j}$ we use the least square gradient reconstruction method (LSQ). To use LSQ, we compute (1) each midpoint of the cut cell grid edge (x_e, y_e), (2) centroid of the cut cell (x_i, y_i), and (3) displacement matrix $\Delta\mathbf{r}_{\{S_{i,j}\}}$ to their neighboring cell centroids (four/five-point stencil $S_{i,j}$ in Fig. 7.3). We pre-compute these information for each grid, and use them to do least

square fitting on the following equation:

$$Q_{\{S_{i,j}\}} - Q_{i,j} = \Delta\mathbf{r}_{\{S_{i,j}\}} \nabla Q_{i,j}, \tag{7.1}$$

where $\{S_{i,j}\}$ makes up the stencil for (i, j) with $|\{S_{i,j} \ : S_1, ..., S_n\}| = n$,

$$Q_{\{S_{i,j}\}} - Q_{i,j} = \begin{bmatrix} Q_{S_i} - Q_{i,j} \\ \vdots \\ Q_{S_n} - Q_{i,j} \end{bmatrix}$$

and dimension thus being $n \times 3$, and

$$\Delta\mathbf{r}_{\{S_{i,j}\}} = \begin{bmatrix} (x_{S_i} - x_i), (y_{S_i} - y_j) \\ \vdots \\ (x_{S_n} - x_i), (y_{S_n} - y_j) \end{bmatrix}$$

with dimension $n \times 2$, the entries (x_i, y_j) being the centroid of cell (i, j) and (x_{S_i}, y_{S_i}) being the centroid of cell neighboring S_i, and finally

$$\nabla Q_{i,j} = \begin{bmatrix} h_x, (hu)_x, (hv)_x \\ h_y, (hu)_y, (hv)_y \end{bmatrix}_{(i,j)},$$

being the 2×3 matrix form of the approximation of the gradient at (i, j).

With the gradient $\nabla Q_{i,j}$ computed we can 'walk' from the centroid (x_i, y_j) to the cell grid edges (x_e, y_e) to get $Q^e_{i,j} = Q_{i,j} + \nabla Q^T_{i,j}(x_e - x_i, y_e - y_i)$. We use this gradient calculation only at the barrier edge and elsewhere we use second order wave propagation method. This is to reduce expensive gradient computation and sources of possible numerical instability [10]. Also, we resort to using gradient approximations on the barrier edge, since it is not clear what $I = i + 1$ or $i - 1$ in Eq. (3.25) should indicate, while using gradients makes it clear that we should linearly approximate the solution at the midpoint of the barrier edge and compute the Riemann problem there.

To approximate the gradient, we use the following stencils depending on which cut cell we are approximating the gradient for. There are three types of stencils we use, as shown in Fig. 7.3. In the diagram we focus on the upper cut cells at the center. However, the same principle applies to the lower cut cells and also the cut cells with barrier of slope $m > 1$. We use the regular five point stencil with the exception of cells that the barrier blocks the central cell from accessing.

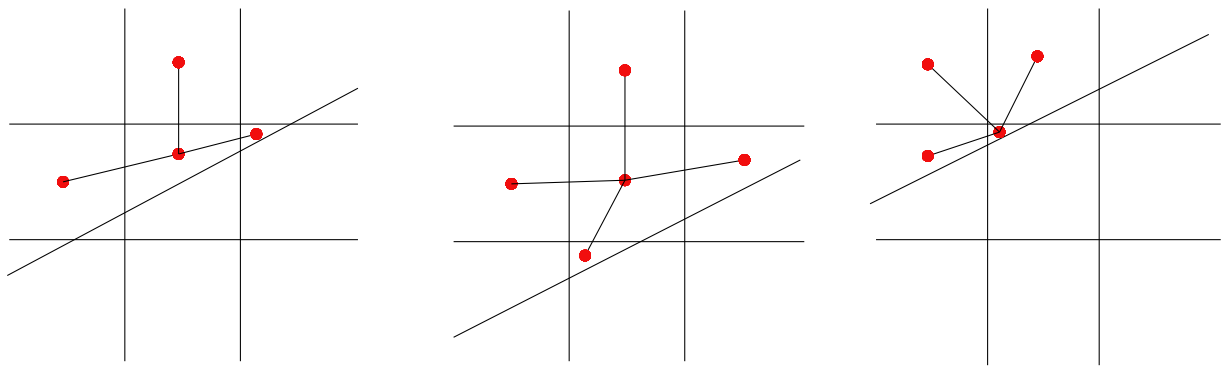


Figure 7.3: Stencils used for approximating gradients on different types of cut cells. The cut cell of interest is the upper cut cells at the center.

To limit the gradients we use the Barth-Jespersen limiter. This limiter has the advantage of keeping the gradients low such that the reconstructed value at the barrier edge will always be in between the maximum and minimum values over the stencil. This assures nice properties such as positivity for the height variable, given everywhere else height is positive. Also, it avoids oscillations by not introducing new maxima.

The Barth-Jespersen limiter α is computed by taking the minimum of α_{S_k}, where S_k is a member of stencil $S_{i,j} = \{S_k\}$:

$$\alpha = \min\{\alpha_{S_k}\}, \tag{7.2}$$

where

$$\alpha_{S_k} = \begin{cases} \min(1, (M_{i,j} - Q^n_{i,j})/(Q^n_{S_k} - Q^n_{i,j})) & \text{if } Q^n_{S_k} - Q^n_{i,j} > 0 \\ \min(1, (m_{i,j} - Q^n_{i,j})/(Q^n_{S_k} - Q^n_{i,j})) & \text{if } Q^n_{S_k} - Q^n_{i,j} < 0 \\ 1 & \text{if } Q^n_{S_k} - Q^n_{i,j} = 0, \end{cases} \tag{7.3}$$

where $M_{i,j}$ denotes the maximum solution value over stencil and $m_{i,j}$ the minimum. We simply multiply the limiter to $\nabla Q^n_{i,j}$ to give $\nabla \tilde{Q}^n_{i,j} = \alpha \nabla Q^n_{i,j}$. Note that since our $Q \in \mathbb{R}^3$, we apply the limiter to each variable in Q (i.e. h, hu, hv).

With the limited gradients calculated on either side of the cut, we compute the linearly reconstructed value Q at the barrier edge from both sides and apply rotated wave redistribution on those values as described in Chapter 4.

7.1.3 At non-barrier edges

At every other edge, which are the vertical and horizontal edges of the cut cell, we employ the base method as described in Section 3.3.2 with two caveats: weighting of fluctuation waves as done in SRD and choice of waves for limiting at the cut cell non-barrier edges.

First order

In the first order, the $\mathcal{A}^{\pm}\Delta Q_{i-1/2,j}$ and $\mathcal{B}^{\pm}\Delta Q_{i,j-1/2}$ are computed exactly in the same way as described in Chapter 3. Because of the shorter length of the cut cell edges, however, we need to *weight* the fluctuations that arise by the length of the cut cell edge to Δx [46], just as was done with the fluctuation at the barrier edges. This is the first caveat and shown diagrammatically in Fig. 7.4.

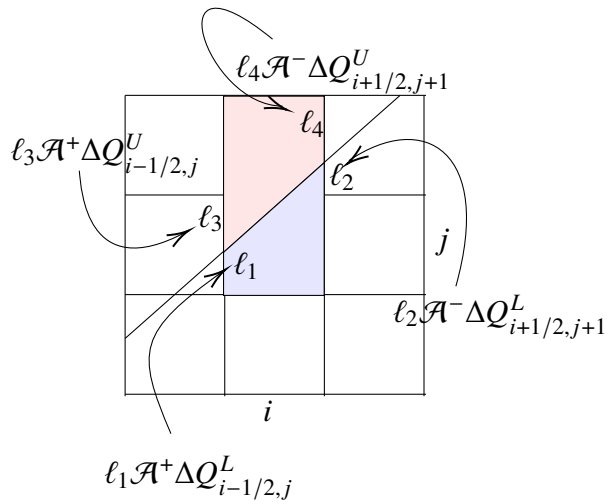


Figure 7.4: The length (l_i) weighted waves on cut edges used to update merged cell for upper cut cell (i, j) in red and for lower cut cell $(i, j+1)$ in blue. The fluctuations are also computed as usual at the uncut edges (e.g. bottom edge and right edge of cell (i, j) for the blue merged cell).

All in all, the general update formula for a lower cut cell in first order will look as follows:

$$
\begin{aligned}
Q^{L,n+1}_{i,j} = & \frac{(\alpha^L_{i,j}Q^{L,n}_{i,j} + \alpha^L_{i,j+1}Q^{L,n}_{i,j+1})}{\alpha^L_{i,j} + \alpha^L_{i,j+1}} - \frac{\Delta t}{\alpha^L_{i,j} + \alpha^L_{i,j+1}}(\ell_{i,j}\mathcal{A}^+\Delta Q_{i,j} + \ell_{i,j+1}\mathcal{A}^+\Delta Q_{i,j+1} \\
& + \ell^L_{i-1/2,j}\mathcal{A}^+\Delta Q^L_{i-1/2,j} + \ell^L_{i+1/2,j}\mathcal{A}^-\Delta Q^L_{i+1/2,j} \\
& + \ell^L_{i+1/2,j+1}\mathcal{A}^-\Delta Q^L_{i+1/2,j+1} + \ell^L_{i,j-1/2}\mathcal{B}^+\Delta Q^L_{i,j-1/2}),
\end{aligned} \tag{7.4}
$$

where $\alpha^{U/L}_{i,j}$, $\ell_{i,j}$ denote the area of cut cell and the length of the barrier edge, and $\ell^{U/L}_{i\pm1/2,j}$ and $\ell^{U/L}_{i,j\pm1/2}$ represent the lengths of the vertical and horizontal edges of the cut cell, respectively, as done in Fig. 6.7. The first term in this update formula is the merged cell average and the rest are net fluctuation. Note that now since $\alpha^L_{i,j} + \alpha^L_{i,j+1} > 0.5\Delta x\Delta y$, we do not run into CFL restriction. Upper cut cells are updated in a similar manner.

Second order

In second order, waves and limited wave corrections are also computed as described in Section 3.3.2 and weighted by ℓ_i. However, the second caveat is in computing second order correction

in the non-barrier cut cell edges. This is because some cut cells do not have a right or left edge, due to the barrier completely blocking it, when information from the right or left edge is necessary to perform limiting (the index I in Section 3.3.3). An example of such scenario is depicted in Fig. 7.5. The upper cut cell only has two non-barrier edges, marked by two red $\times$'s. Note how they do not have naturally adjacent edges both forward and backward to perform wave limiting. In such cases, we use the wave and speed that arise from the barrier (denoted by the overlapping vertical and horizontal double-ended arrows in the right figure) to correct the fluctuations at the left and top edge. The wave and speed from the barrier is first computed normal to the barrier then rotated back to x and y direction.

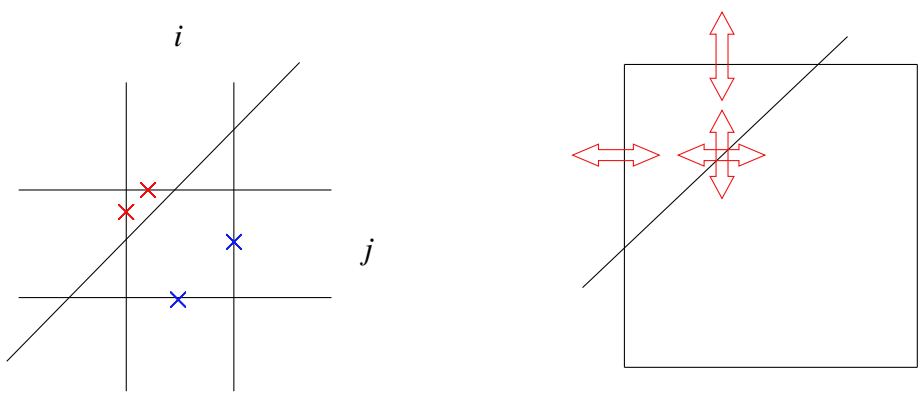


Figure 7.5: Barrier is blocking access from the upper cut cell (i, j) to its right and bottom neighboring cell, when speed and wave information from Riemann problems there (crossed in blue) are required to perform limiting.

7.2 Model problems

Here we only consider the overtopping examples, as they are our main interest in barrier modeling. All the model problems presented below have the same conditions as those shown in SRD, with steady water height of $h = 1.2$ and a dam jump of $\Delta h = 0.8$, giving the overall height of the dam break to be $h = 2.0$. The barrier height ℓ is chosen to be $\ell = 1.5$. For the boundary conditions, we have one extrapolation boundary condition on the overtopping side, such that the wave can exit the domain after overtopping the barrier. Furthermore, these computations are done using the second order method, with the Barth Jespersen limiter for gradient at the cut edge and

minmod limiter at Cartesian edges. The difference between these results and SRD results is that these examples are all run using the second order CM method, and have much higher resolution with 900×900 grid.

As before, we run simulations with the same initial condition with a mapped grid suited for each barrier and observe the 2D color contour plots of water height at specific times from both the CM cut cell method and mapped example runs. Also we observe gauge data, marked by an asterisk with a number in the figures below. We compare our results only against the numerical solutions computed on mapped grids, as we have already seen the behavior of GEOCLAW simulations earlier.

7.2.1 The 20° angled barrier

The results show very close match between CM results and mapped results. We place two gauges, one at $(0.5, 0.39)$ as before and another at $(0.5, 0.8)$ to both capture the reflection and also the overtopped wave further away from the barrier.

Case of Overtopping

We show the initial condition in Fig. 7.6.

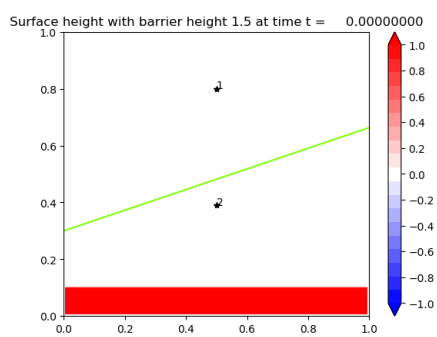


Figure 7.6: Initial condition for overtopping case: $\ell = 1.5$. Dam height is 2.0. Grid is 900×900.

We observe from both CM results and mapped grid results that the wave is abated from the barrier and proceeds in the same direction after overtopping (Fig. 7.7). The reflection wave on the lower side of the barrier is gliding up the linear barrier, while reflecting back in normal direction to the barrier at the same time.

(a) CM (b) Mapped grid

Figure 7.7: Linear barrier example at $t = 0.3$: CM on left and mapped grid on right.

At $t = 0.7$, the overtopped wave almost exits the domain at the upper boundary. On the lower side of the barrier, the reflected waves are bouncing around the wall boundary conditions, having reflected from the bottom boundary (the lower left radial wave upward) and from the barrier corner (the upper right radial wave downward).

(a) CM (b) Mapped grid

Figure 7.8: Solution at $t = 0.7$.

An interesting observation to be made is at $t = 1.4$, the doubly-reflected wave (from barrier and bottom edge) overtops again at the upper right, and its "double-tongued" shape is nicely captured in both CM and mapped grid results.

Comparison to mapped grid

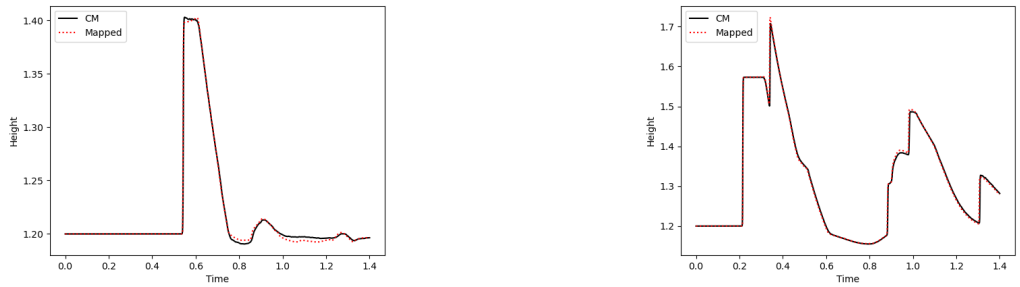
The way to validate our simulations is again by comparing them against a mapped grid example, as shown on the right column in (Fig. 7.7-Fig. 7.9). The mapped grid is shown in Fig. 6.17. We transform the computational uniform grid (x, y) into a skewed grid $f_L(x, y)$, with the following mapping f_L, as done previously in SRD examples.

(a) CM (b) Mapped grid

Figure 7.9: Solution at $t = 1.4$

A remark about this mapping is that it is a linear and affine transformation that stretches and squeezes the upper and lower halves of the computational domain into the physical grid as shown in Fig. 6.17. We implement wave redistribution at the computational barrier edge ($y = y^*$). The finite volume method for mapped grids is explained in [10] and is very much similar to the rotational part of the wave redistribution method explained in Section 6.1.5.

We observe the gauge results between the CM and the mapped examples in Fig. 7.10 and see a very good comparison. The gauge results of the mapped example are from a run on 900×900 grid, and the results of the CM example are also from a 900×900 grid.



(a) Time profile of Gauge 1 $(0.5, 0.8)$. (b) Time profile of Gauge 2 $(0.5, 0.39)$.

Figure 7.10: Gauge comparisons between CM and mapped grid. Results from 900×900 for CM and from 900×900 for mapped grid.

Convergence

For convergence studies we observe the convergence of the wave height profile at each gauge point. A note on the usage of gauge results for convergence is that as our studies are for developing

models for protective strategies against storms, we focus not on the numerical results at a slice of the spatial domain at a fixed time, but rather on a slice of the temporal domain at a fixed spatial point in the grid. This is often done in storm simulations, to test the storm models' accuracy against real results collected at a specified gauge point off a coast (e.g. surge measurement at Battery Park, NYC).

		L_1 Error (1st)		L_1 Error (2nd)	
Δx	N_x, N_y	Gauge 1	Gauge 2	Gauge 1	Gauge 2
4.e-2	25	1.03e-2	2.45e-2	8.95e-3	1.37e-2
2.e-2	50	3.21e-3 (3.18)	1.13e-2 (2.16)	2.57e-3 (3.20)	4.28e-3 (3.48)
1.e-2	100	9.20e-4 (3.49)	4.02e-3 (2.81)	7.27e-4 (4.00)	1.07e-3 (3.54)
0.666e-2	150	4.13e-4 (2.23)	2.22e-3 (1.80)	3.36e-4 (1.49)	7.17e-4 (2.16)
0.5e-2	200	2.56e-4 (1.61)	1.25e-3 (1.77)	1.62e-4 (1.84)	3.88e-4 (2.08)

Table 7.1: L_1 errors at Gauge 1 (0.5,0.8) and Gauge 2 (0.5,0.39).

Also we only study the convergence of the overtopped examples, for similar reasons as above-mentioned, namely, that in realistic scenarios the barriers will be overtopped by incoming waves. For our standard solution to compare against, we use the mapped grid example, on a 900×900 grid. Then we take the heights at time intervals $\{0.0,\ 0.1,\ \ldots\ 1.4\}$ and compare the CM and mapped results.

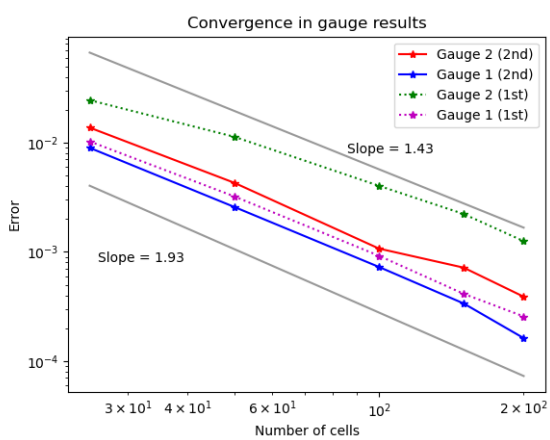


Figure 7.11: Convergence of gauge profiles using L_1 norm.

Shown in Table 7.1 are the L_1 normed errors for the overtopped wave profile. We also have the convergence plot in Fig. 7.11. We observe that the gauge 2 seems to converge little more slowly,

especially in the first order method. We attribute this to the fact that it is nearer to the barrier, where the cut cells are. However, in the second order method, gauge 2 performs just as well as gauge 1, because of the gradient reconstruction at the edges and limiting. We also show the L_∞ error norm and observe similar convergence (see Table 7.2 and Fig. 7.12):

		L_∞ Error (1st)		L_∞ Error (2nd)	
Δx	N_x, N_y	Gauge 1	Gauge 2	Gauge 1	Gauge 2
4.e-2	25	7.00e-3	4.35e-2	5.57e-3	2.87e-3
2.e-2	50	2.04e-3 (3.41)	1.93e-3(2.25)	8.86e-4 (6.29)	8.35e-4 (3.44)
1.e-2	100	5.03e-4 (4.06)	8.13e-4 (2.38)	2.56e-4 (3.45)	3.03e-4 (2.75)
0.666e-2	150	2.18e-4 (2.31)	4.93e-4 (1.65)	1.37e-4 (1.87)	1.40e-4 (2.16)
0.5e-2	200	1.30e-4 (1.67)	2.99e-4(1.64)	5.51e-5 (2.48)	1.04e-4 (1.34)

Table 7.2: L_∞ errors at Gauge 1 (0.5,0.8) and Gauge 2 (0.5,0.39).

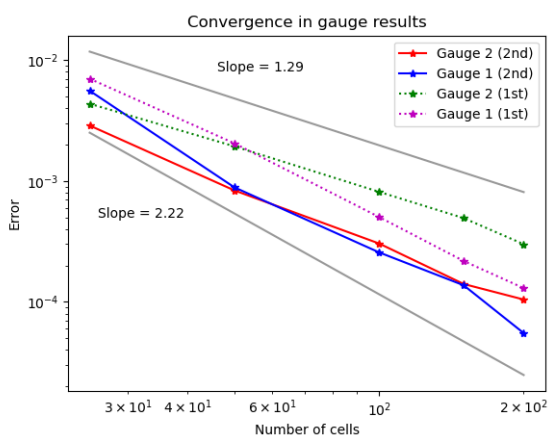
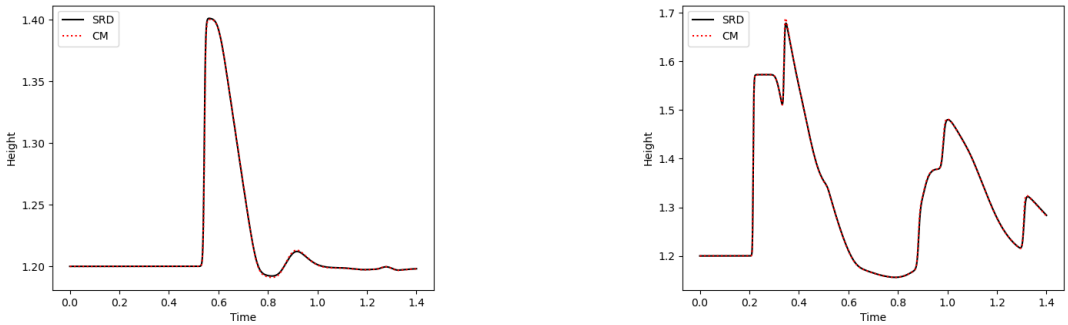


Figure 7.12: Convergence of gauge profiles using L_∞ norm.

Comparison between SRD and CM

We can now compare the results of CM with those of SRD at the same resolution we use in this chapter $\Delta x = 1/900$. We compare the gauge results and observe that they are almost identical as can be seen in Fig. 7.13 and Fig. 7.14. These comparisons show that both give comparable accuracy with the same resolution, meaning that all things being equal (order of method, grid setup, etc.) either method gives a viable solution to our problem. The 2D contour plot of the two

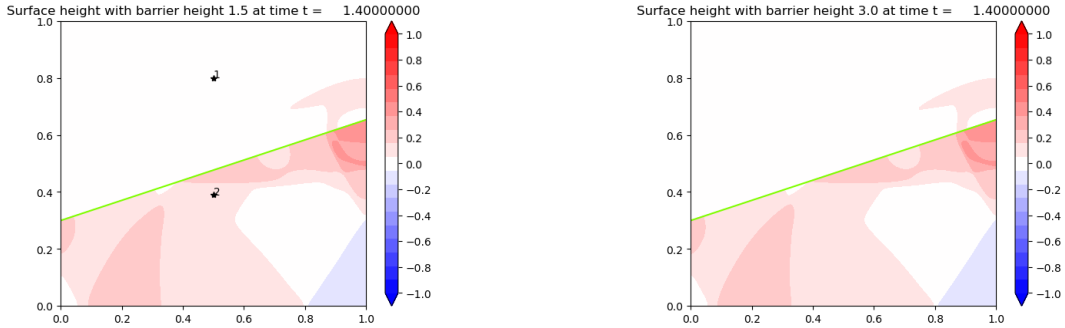


(a) Gauges 1: overtopped wave

(b) Gauges 2: reflected wave

Figure 7.13: The gauge results of the overtopping linear barrier problem between SRD and CM.

methods also show approximately same behavior. Here we plot the final time of the linear barrier example at $\Delta x = 1/900$.



(a) SRD

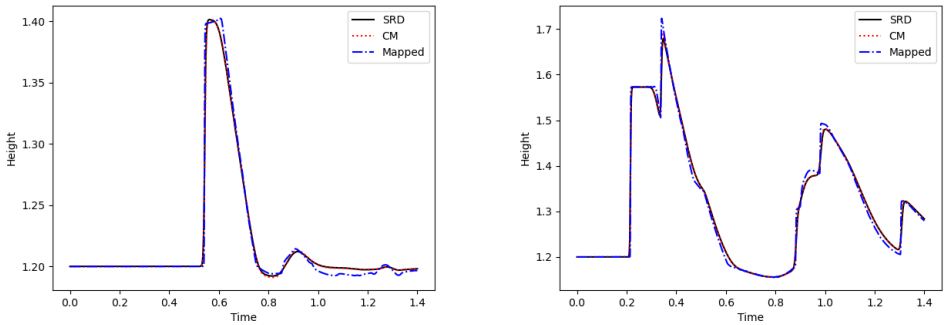
(b) CM

Figure 7.14: Comparison of SRD with CM in first order at $\Delta x = 1/900$.

We show the gauge results of all three methods, SRD, CM and mapped grid, in Fig. 7.15 and observe their good agreement.

7.2.2 The V Barrier

Again we run our algorithm on the V barrier model problem, with gauge points on either side of the barrier, to test reflection and overtopping, and on either side of the domain, to test symmetry. We observe good comparison and convergence between CM and mapped grid results.



(a) Overtopping gauge (b) Reflecting gauge

Figure 7.15: Gauge results for linear barrier problem on 900×900

Case of Overtopping

Again we have water height as 1.2 and the dam jump to be 0.8 to achieve total dam height of 2.0. As we shall see, this is enough height to overtop the barrier of height 1.5. We do a comparison against mapped grid results to show accuracy of our CM results. From Fig. 7.17 to Fig. 7.19, we can see the similarities of the two 2D plots. At time $t = 0.3$, we see the overtopping wave's "wing"-like structure just below the V-barrier where the amplitude is highest. In Fig. 7.18, we see the overtopping wave moving radially outward from the center, shown both in the CM and mapped grid results. Finally in Fig. 7.19 we see the small islands of wave amplitude at the bottom center of the plot in both results.

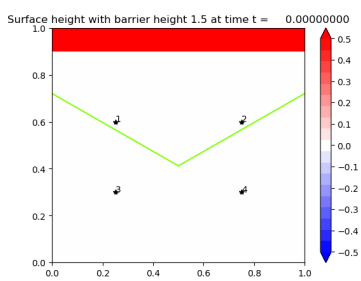
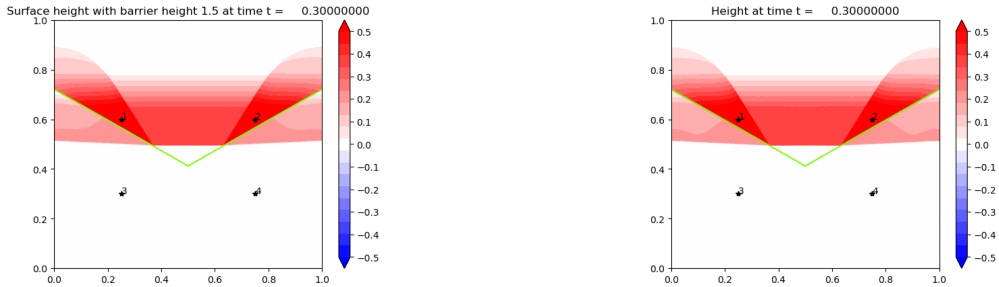


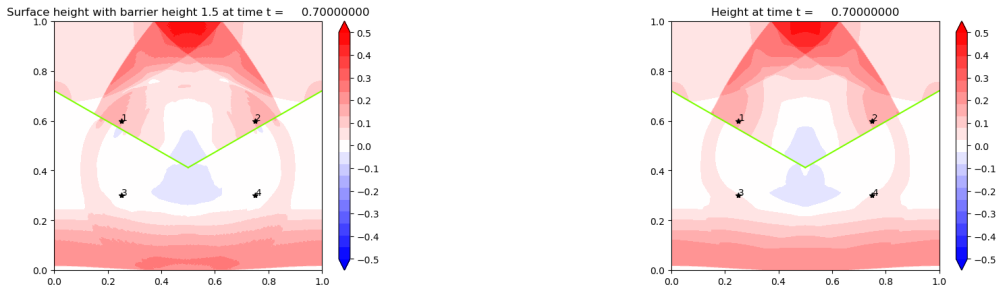
Figure 7.16: Initial condition for overtopping case: $\ell = 1.5$. Dam height is 2.0. Grid is 900×900 for CM and 1000×1000 for mapped grid.

We place our gauges at either side of the V-barrier $(0.25, 0.3)$, $(0.75, 0.3)$, $(0.25, 0.6)$, $(0.75, 0.6)$ in order to test for symmetry in the results. Indeed we do find symmetry as can be seen in the iden-



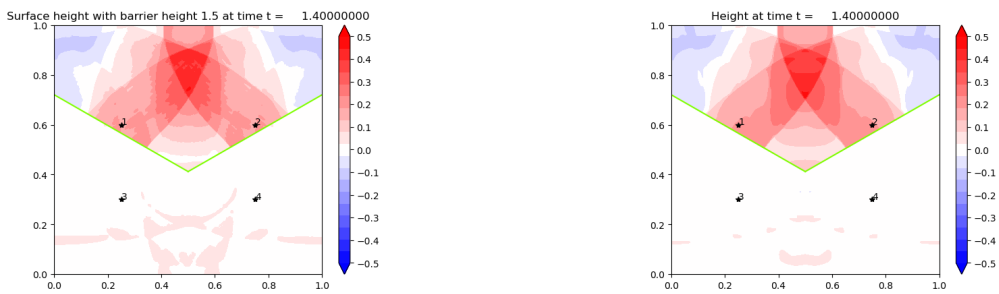
(a) CM (b) Mapped grid

Figure 7.17: CM Comparison with mapped grid at $t = 0.3$. Note the structure of the just overtopped wave.



(a) CM (b) Mapped grid

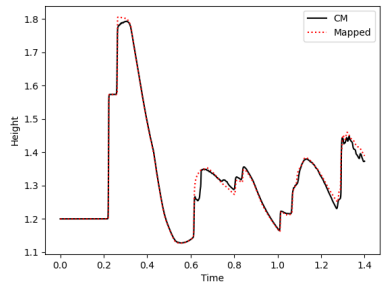
Figure 7.18: CM Comparison with mapped grid at $t = 0.7$. Note the radially outward moving wave from the center of the V-barrier.



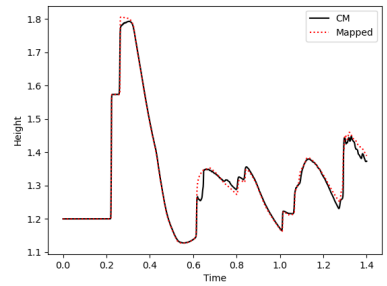
(a) CM (b) Mapped grid

Figure 7.19: CM Comparison with mapped grid at $t = 1.4$. Note the “island" of peak at the bottom center.

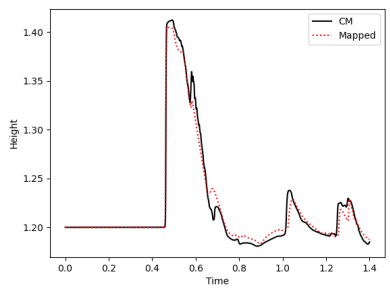
tial plots of the gauge results in Fig. 7.20.



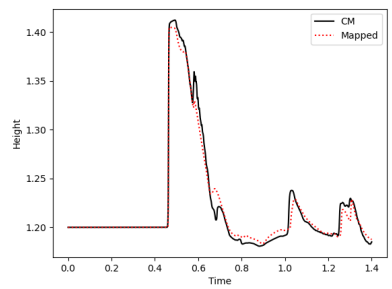
(a) Time profile of Gauge 1 $(0.25, 0.6)$.



(b) Time profile of Gauge 2 $(0.75, 0.6)$.



(c) Time profile of Gauge 3 $(0.25, 0.3)$.



(d) Time profile of Gauge 4 $(0.75, 0.3)$.

Figure 7.20: Gauge profiles compared with mapped grid results: 900×900 for CM and 1000×1000 for mapped grid.

7.2.3 Comparison to mapped grid

The mapped grid for the V barrier is shown in Fig. 6.31. Again we transform the computational uniform grid into a chevron grid akin to what is done in [7], with mapping f_V as used before.

The mesh size is chosen such that the barrier in the computational domain lies exactly on $y = y^*$. Transforming the grid with this mapping, using the mapped grid finite volume method explained in Section 6.1.5 and [10, 27], and applying wave redistribution at the computational barrier edge will give us the numerical solution to the exact same problem that we solve in the Cartesian coordinates using CM cut cell method.

Convergence

In Fig. 7.20 we plot the gauge results of 1000×1000 mapped grid V-barrier example and 900×900 CM example. We do see that the CM results contain more fine movements of the wave

and that the mapped grid example produces more smooth wave patterns, implying slight difference in the rate of convergence between the CM method and mapped grid method. Overall, nonetheless, we see convergence as shown in Table 7.3. The order of convergence are around 2 for the second order method and 1.6 for the first order method for both gauge points (Fig. 7.21).

We note that CM resolves more detail even on a slightly lower resolution than the mapped grid example. This can be seen in the finer details in both the reflected side (where the reflected waves cross in the center) and also the overtopped side, in the appearance of more residual overtopped waves (Fig. 7.19).

		L_1 Error (1st)		L_1 Error (2nd)	
Δx	N_x, N_y	Gauge 1,2	Gauge 3,4	Gauge 1,2	Gauge 3,4
4.e-2	25	4.83e-2	9.86e-3	4.16e-2	8.07e-3
2.e-2	50	1.58e-2 (3.06)	3.32e-3(2.97)	8.55e-3(4.87)	2.27e-3 (3.56)
1.e-2	100	5.70e-3 (2.77)	1.01e-3 (3.30)	2.28e-3 (3.74)	5.12e-4 (4.42)
0.666e-2	150	3.04e-3 (1.88)	4.51e-4 (2.23)	1.05e-3 (2.17)	2.33e-4 (2.20)
0.5e-2	200	1.83e-3 (1.66)	2.53e-4(1.78)	5.14e-4 (2.05)	1.54e-4 (1.52)

Table 7.3: L_1 errors at Gauges 1,2 (0.25,0.6), (0.75,0.6) and Gauges 3,4 (0.25,0.3), (0.75,0.3).

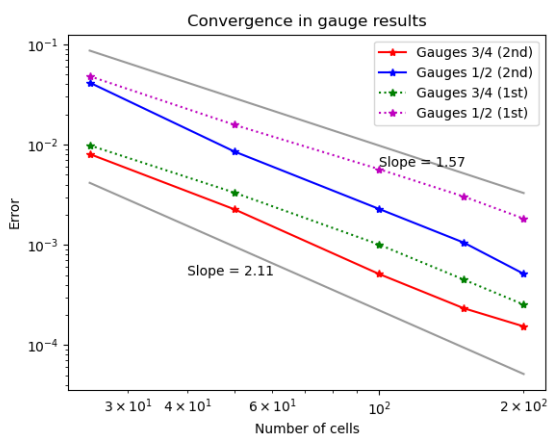


Figure 7.21: Convergence of gauge profiles in L_1 norm.

In the V barrier example we also compute the L_∞ error and observe similar results, as seen in the linear example (Table 7.4):

We can see from Fig. 7.22 similar behavior in the orders of convergence in L_∞:

Δx	N_x, N_y	L_∞ Error (1st)		L_∞ Error (2nd)	
		Gauge 1,2	Gauge 3,4	Gauge 1,2	Gauge 3,4
4.e-2	25	1.01e-2	3.69e-3	1.06e-2	2.95e-3
2.e-2	50	2.66e-3 (3.79)	1.86e-3 (1.98)	2.02e-3 (5.23)	8.49e-4 (3.47)
1.e-2	100	1.02e-3 (2.60)	4.44e-4 (4.18)	4.57e-4 (4.42)	1.26e-4 (6.74)
0.666e-2	150	5.54e-4 (1.84)	1.35e-4 (3.29)	1.93e-4 (2.36)	6.90e-5 (1.82)
0.5e-2	200	2.82e-4 (1.96)	4.65e-5 (2.90)	1.19e-4 (1.62)	4.53e-5 (1.52)

Table 7.4: L_∞ errors at Gauges (0.25,0.6), (0.75,0.6) and Gauges (0.25,0.3), (0.75,0.3).

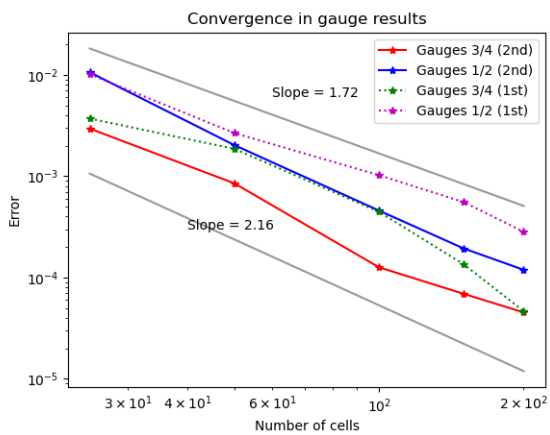


Figure 7.22: Convergence of gauge profiles using L_∞ norm.

7.2.4 Conservation

In Fig. 7.23, we again check numerical conservation of our CM method by doing a V barrier example that contains water on one side of the barrier as in Fig. 6.24, measuring total momentum μ_T as defined in Eq. (7.1). The total momentum relative difference is centered around zero and fluctuates with variance of around 1e-14. Although the error is of order 10^{-14} compared to 10^{-15} in the SRD method, the CM method keeps total momentum more centered around zero than the SRD method, which had relative difference centered around -4e-15.

Comparison between SRD and CM

We can again further compare the results using SRD and CM. We compare them in the same resolutions with grid of 900×900. Results show that they are almost identical, as can be seen in the gauge results (Fig. 7.24).

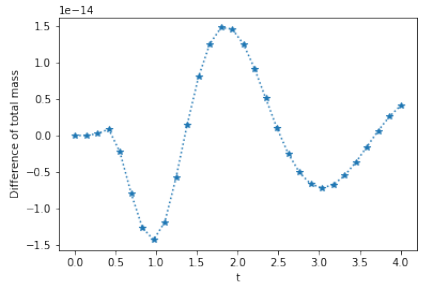
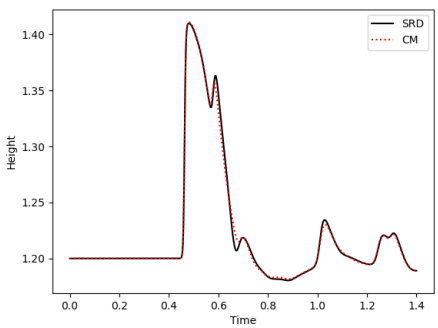
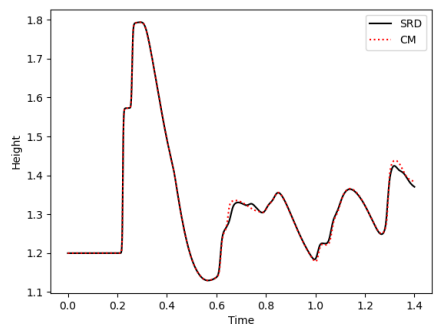


Figure 7.23: Relative momentum difference across time.



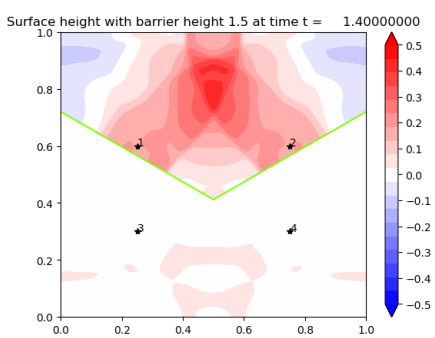
(a) Gauges 3,4: overtopped wave



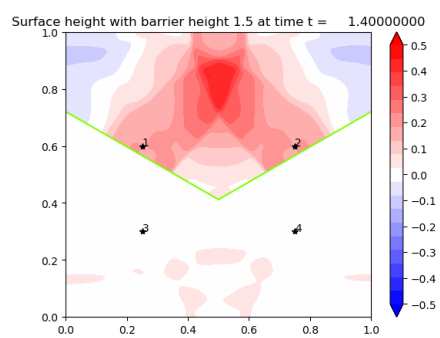
(b) Gauges 1,2: reflected wave

Figure 7.24: The gauge results of the overtopping V problem between SRD and CM.

Furthermore, the 2D contour plot at the final time step ($t = 1.4$) is close to identical, with SRD showing a little more symmetry than CM (Fig. 7.25): The overtopped wave at the bottom appears



(a) SRD

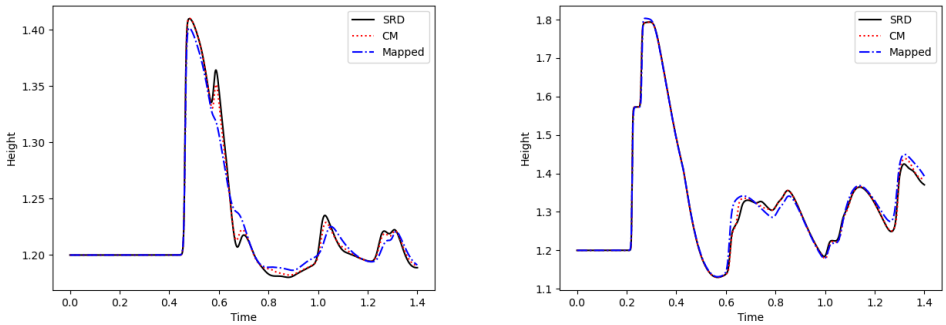


(b) CM

Figure 7.25: Contour plots of overtopping V problem between SRD and CM at $\Delta x = 1/900$.

more clearly in both cut cell methods than the mapped grid results. We also show the comparison between SRD, CM and mapped grid methods for the gauge results in Fig. 7.26 and observe good

agreement.



(a) Overtopping gauge (b) Reflecting gauge

Figure 7.26: Gauge results for V barrier problem on 900×900 for cut cell methods and 1000×1000 for mapped grid.

7.3 Computational Superiority to Refinement using GEOCLAW

To highlight the computational superiority of using the CM method on the zero width barrier, we again do a V-barrier simulation on GEOCLAW using same resolution as CM but adaptive (double) refinement at the barrier. This means that GEOCLAW has more resolution (2X) at the barrier than the CM example.

We show results from using resolution $\Delta x = 1/300,\ 1/450$. The relaxed timesteps and reduced number of timesteps show the computational benefit we derive from our proposed method. For $\Delta x = 1/300$ we observe that we get 184% increase in the minimum Δt (from 8.6e-06 to 2.4e-05) and 162 % increase in the average Δt (from 0.00028 to 0.00074) and about fivefold decrease in the number of steps taken (9958 steps to 2037 steps). For $\Delta x = 1/450$, we observe that we get fivefold increase in the minimum Δt (from 2.9e-06 to 1.9e-05) and 175 % increase in the average Δt (from 0.00018 to 0.00049) and about fivefold decrease in the number of steps taken (15861 steps to 3082 steps). We do note, however, that there may be different reasons for using GEOCLAW 's adaptive mesh refinement versus using a zero-width approximation, e.g. to get much finer detail near the barrier region.

Chapter 8: Realistic Numerical Examples

Finally we present realistic barrier examples including bathymetric variation. On all our examples here, we resort to CM method to emphasize the experimental value and simulation results more than the method used. SRD methods also work on the same examples. First we will show a Gaussian shaped island being protected by the two types of barriers. Then we will show a normalized bathymetry of South Carolina (SC) being protected by a linear barrier and actual bathymetry of NYC being protected by a linear barrier. The sea surface level is set at -0.8 for all problems. For the Gaussian island and SC problems, we set $g = 1$ for illustrative purpose. (In a sense, one can think of even $g = 1$ case as being 'physical' since the spatial dimension is being scaled by 9.8m, meaning unit of 1 in these examples correspond to 9.8m). For NYC, we set $g = 9.8m/s^2$ for a realistic simulation. The only additional algorithmic aspect that needs further discussion in these examples is wetting and drying methods.

8.1 Drying and Wetting algorithm

One of the main issue with drying and wetting is that on a dry region, there is no water, which means we can no longer use the usual eigenvalues for our wave speeds. In fact, the water and dry interface moves at a much faster speed than a wave in an all wet region. The speed of a wave inundating a dry region is given by $s = \hat{u} \pm 2\sqrt{g\bar{h}}$ (with Roe or Einfeldt average $\hat{u}$ and arithmetic average $\bar{h}$) as opposed to $s = \hat{u} \pm \sqrt{g\bar{h}}$ for a wave in a wet region. This is calculated using Riemann invariants [30]. Thus, higher speed means greater CFL restriction, as the CFL number will be high. To compensate for this, we need to take shorter Δt (recall CFL = $s\Delta t/\Delta x$).

Although we use a Riemann problem solver that handle wetting and drying situations [30], practice shows that the solver is not as robust as desired when there is a lot of inundation and

recession of waves on dry bathymetry all happening simultaneously. These are cases where the rarefaction waves have both positive and negative speeds and are called transonic rarefactions [10], which are another source of numerical challenge. Our solver can give negative updates in such cases. There are, however, many methods that deal with this issue [47]. We employ the relatively simple method described in [48] where negatively updated state cells (due to the Riemann problem at water-ground interface) are zeroed out. Adding this fix stabilizes the solver.

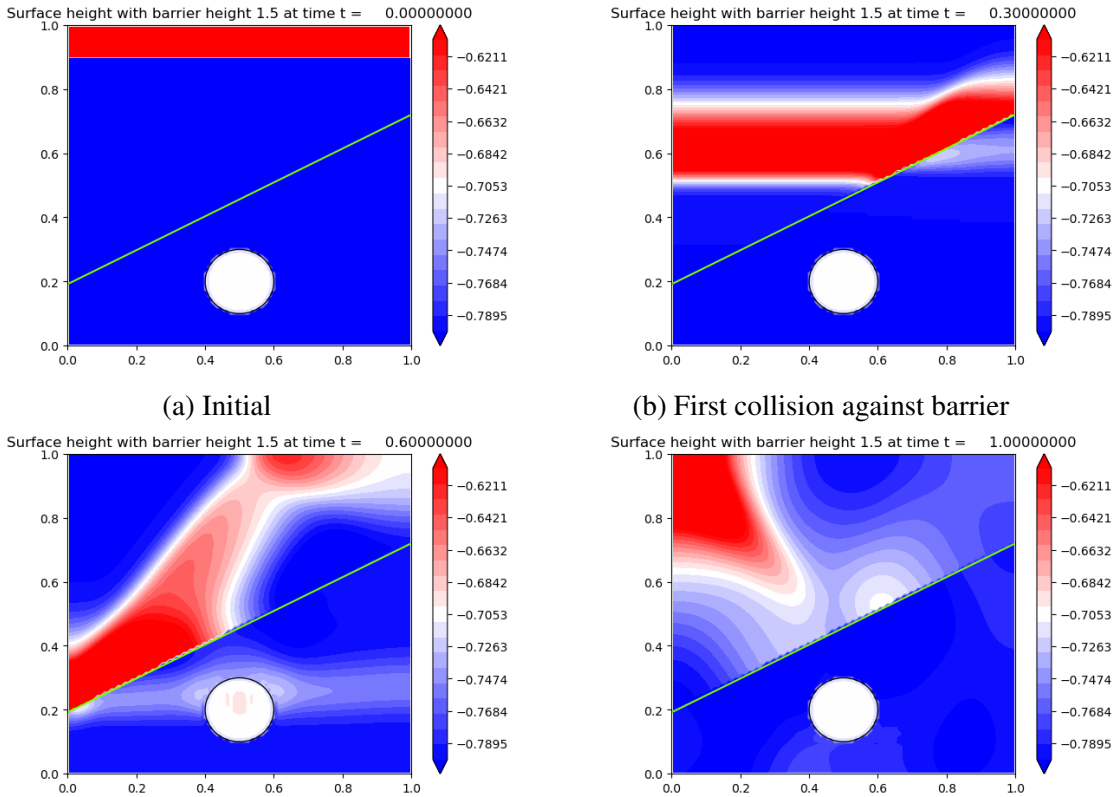
8.2 Gaussian island

Here we do two experiments that will potentially comment on design aspects of a storm barrier in addition to the effectiveness of barriers in general. To compare the effectiveness of the linear barrier versus the V barrier, we add an island on the other side of the barrier and observe flooding at the island center. We will compare the gauge results without a barrier and with a barrier and observe how much protection each barrier type provides.

The Gaussian island is defined as follows:

$$b(x, y) = \begin{cases} h_{\max} \exp(-(x - x_0)^2 - (y - y_0)^2), & \text{for } d((x, y), (x_0, y_0)) < r^2 \\ -2 & \text{otherwise.} \end{cases} \tag{8.1}$$

The boundary conditions are such that there is a wall boundary condition on the side of the wave and extrapolation condition on the side of the island, to allow for wave to exit after hitting the island. Furthermore, for sake of comparison, the V barrier designed to be mirror images of the upper half of the linear barrier. The initial conditions are $\ell = 1.5$, dam height h_d of 1.7 with base height $h = 1.2$, and island peak height and radius of $h_{\max} = 1.3$ and $r = 0.1$. We provide 2D contour plots of the surface height, which will show the island being inundated. The contour of the island has a grey hue with the shape of a smaller circle inside. This indicates a dry state. When this grey circle changes color to become pinkish, inundation has occurred.



(a) Initial

(b) First collision against barrier

(c) First inundation of island: observe the center of the island.

(d) Receding and wave leaving domain behind island

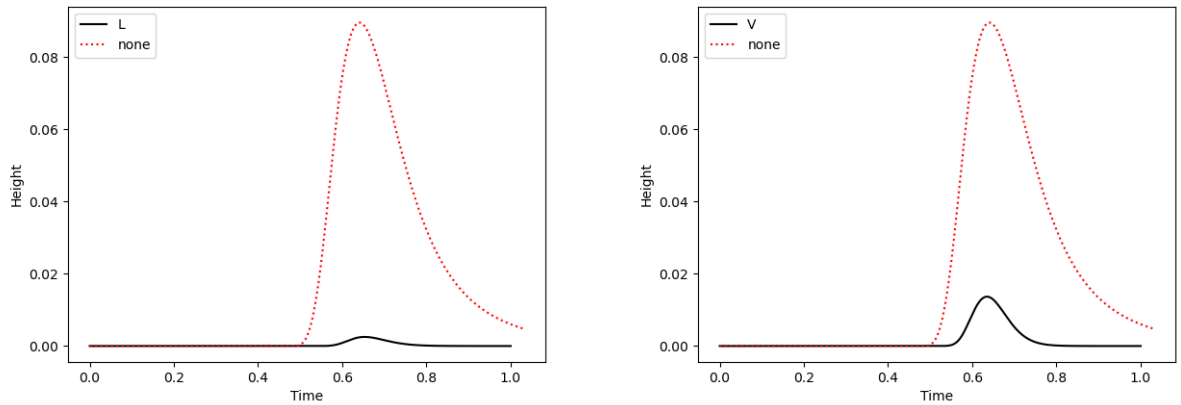
Figure 8.1: Four time snapshots of barrier action protecting an island. Grid 100×100.

8.2.1 Linear barrier

For the linear barrier, we can see from the snapshots in Fig. 8.1 that the waves glide along the barrier and bounce around in the enclosed side of the barrier, while transmitting some of the wave in the direction of the dam break. Inundation starts around $t = 0.6$. The reflected wave is redirected to a direction parallel to the barrier. We observe that for the linear barrier, inundation is about 0.003. Without the barrier, the inundation becomes around 0.09, implying that the barrier protects the island from about 96% of the peak inundation (Fig. 8.2a).

8.2.2 V barrier

We do the same experiment using the V barrier (Fig. 8.3). When it comes to inundation at the island, the V barrier actually does worse. This is because as the water gathers towards the center



(a) Linear barrier vs no barrier: 96% protection.

(b) V barrier vs no barrier: 84% protection.

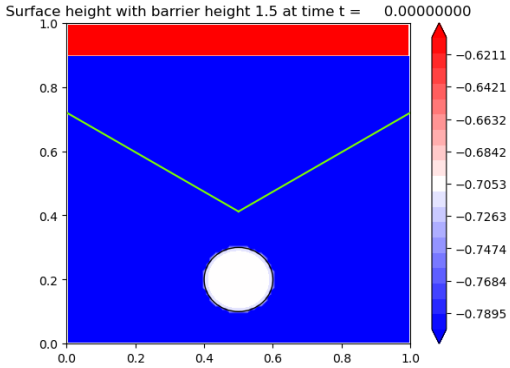
Figure 8.2: Effectiveness of each barrier measured by gauge results at peak of island. V barrier collects water to the center to cause a greater overtopping effect.

from either end, it gains momentum and 'dumps' all the water in the direction of the island. The mitigated inundation is 0.014, or about 84% protection (Fig. 8.2b). We conclude that the linear barrier does better in protecting against inundation (Fig. 8.4). However, the V barrier plots show perhaps more controlled behavior when acting against the wave due to symmetry.

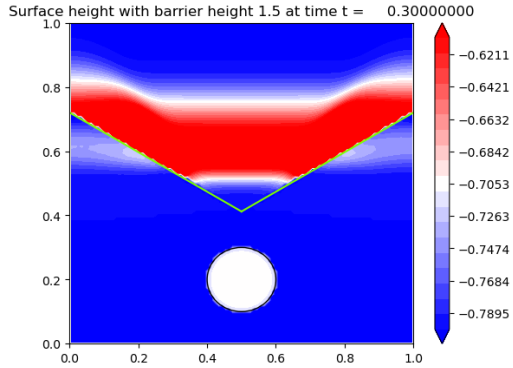
The controlled behavior of the waves in the V barrier could be a merit to consider in barrier design, however, since it directs the incoming wave toward each other toward the center, instead of directing it all in one direction as the linear barrier does. If, for example, there were another island at one end of the linear barrier, the waves could reach there. The V barrier avoids this by aggregating the wave toward the center and evenly distributing out the reflected waves.

8.3 Myrtle Beach, SC

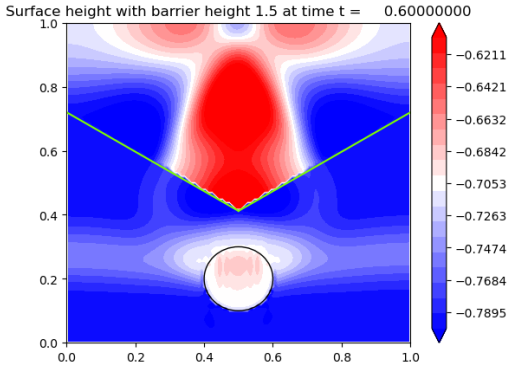
The following two experiments are akin to work done in [49], where storm protective measures are analyzed on multiple coasts. To run a more realistic example, we test case our model on real bathymetric data and choose the coast off South Carolina as a simple example. The geography of this region suits our purposes as we can naturally place a linear barrier just off the coast (Fig. 8.5). There is a nice curvature of bathymetry that can be protected by a slanted barrier.



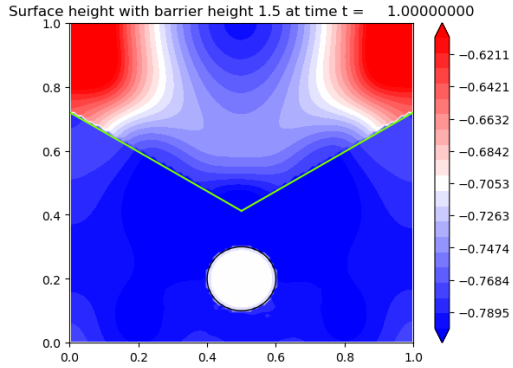
(a) Initial



(b) First collision against barrier



(c) First inundation of island indicated by pinkish patch



(d) Receding waves evenly being spread out

Figure 8.3: Four time snapshots of barrier action protecting an island. Grid 100×100.

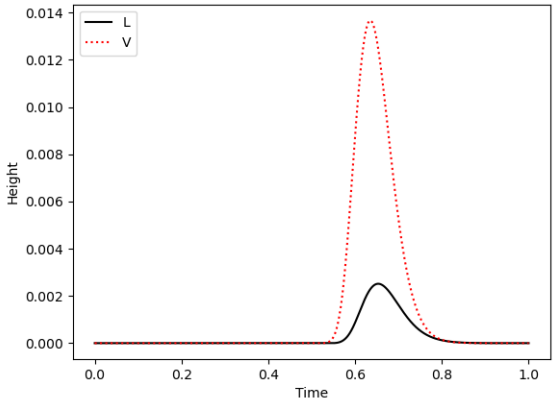
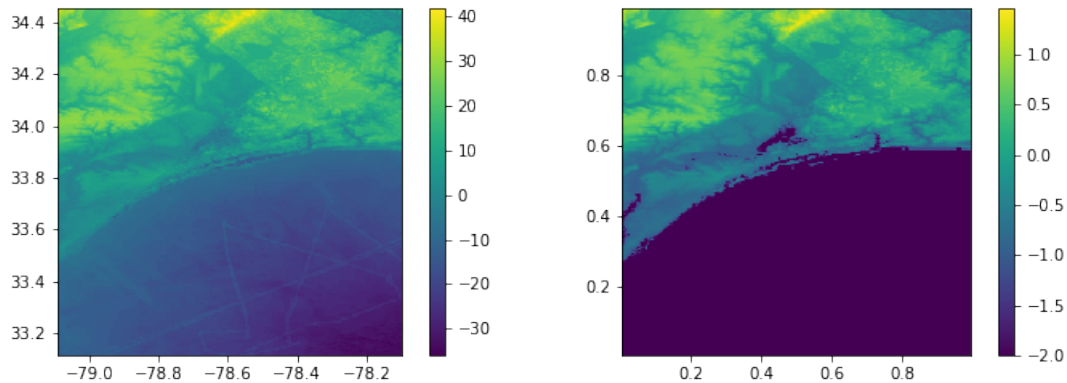


Figure 8.4: Linear barrier vs V barrier: about 78% more protection in linear barrier. V barrier actually gathers the water towards the center, causing a greater overtopping effect than linear barrier.

We take a NetCDF bathymetric data from GEBCO [50], a publicly available bathymetric data source, and normalize the depths and heights as a way of preprocessing the data for our model.



(a) Raw bathymetric data

(b) Processed bathymetric data

Figure 8.5: GEBCO bathymetric data from -79° to −78° and 33° to 34°. Normalization using mean and standard deviation is used to scale the raw physical heights and depths.

We also flatten the ocean floor. This processing eases analysis of the protective effectiveness of the barrier, as widely varying bathymetry levels sometimes make it difficult to know where to exactly look for inundation.

Once we normalize the bathymetry, we flatten the ocean bed where the barrier will lie, in order to have a uniform barrier height above water surface. The bathymetry at the flat bottom is set as two standard deviations below the mean $b = -2$. The barrier is placed as seen in Fig. 8.6. Finally we interpolate using linear interpolation to fit the bathymetric data to our own grid.

To highlight the effectiveness of the barrier, we run a dam break scenario with jump height of 0.5 from $y = 0$ to $y = 0.1$ as seen in Fig. 8.7 and with base height of 1.2.

We then plot a zoomed in region and show the height of water in this region, as it includes an initially dry area that resembles the shape of Florida (see Fig. 8.8). Shown in dark purple is the dry region, where the underlying elevated bathymetry is what is keeping the region dry.

At time $t = 0.5$, we can see clearly the greatest difference between having a barrier and not having a barrier. Along the inland delta, we can see significant flooding in the case of no barrier shown in Fig. 8.9a, compared to the case of barrier shown in Fig. 8.9b.

Also, we can compare the overall domain at initial time and final time to see the effect of the barrier as shown in Figs. 8.10a and 8.10b. We can see that with the barrier, the region behind

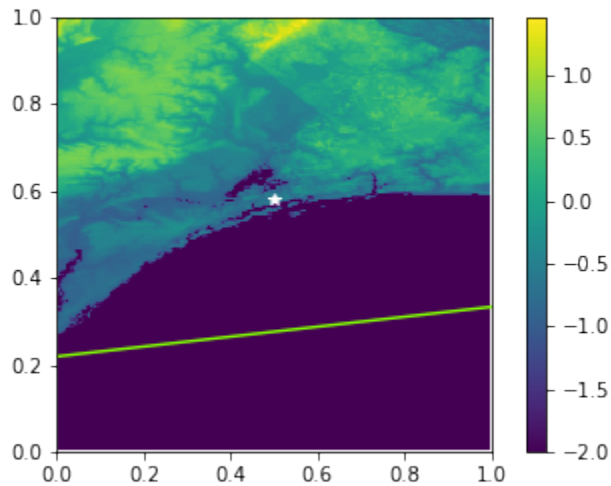


Figure 8.6: Bathymetry with zero width slanted barrier off the bay: $\ell = 1.5$. White star is the gauge point.

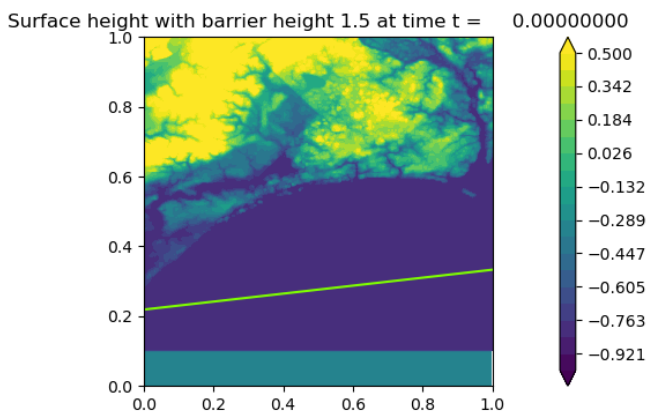


Figure 8.7: Dam break on South Carolina Bay: $\Delta h = 0.5$, $\Delta x = 1/200$. Surface height is being plotted, $\eta = h + b$, with $\eta_{sea} = -0.8$. Barrier height is then $\ell = \eta_{sea} - b_{min} + 0.3$.

the barrier island (inner bay located in region $[0.1, 0.4] \times [0.5, 0.6]$) does not see surge, whereas without the barrier it gets flooded.

Finally, we can compare gauge results at a point $(0.5, 0.58)$ asterisked in Fig. 8.6 and shown in Fig. 8.11. We see that by introducing the barrier, we reduce not only the height of the wave, but also its width, or the time duration of the wave. The barrier also introduces new smaller ripples due to its reflection of waves, as seen near $t = 1.0$.

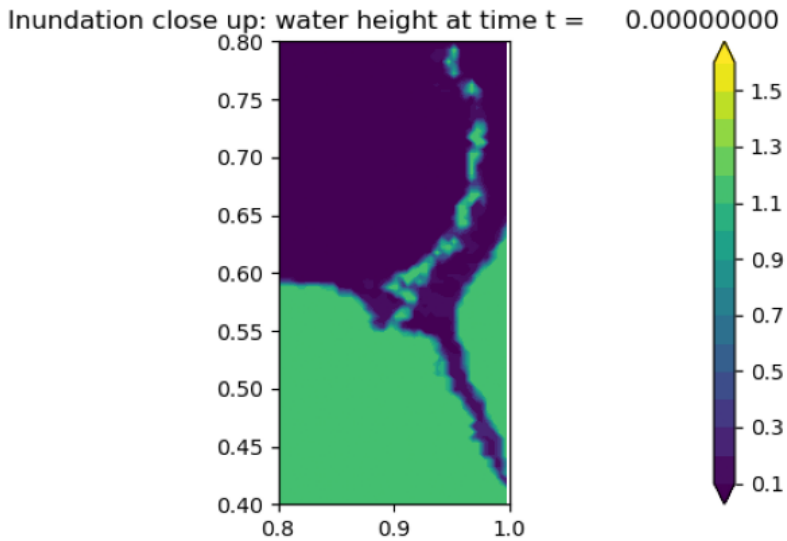
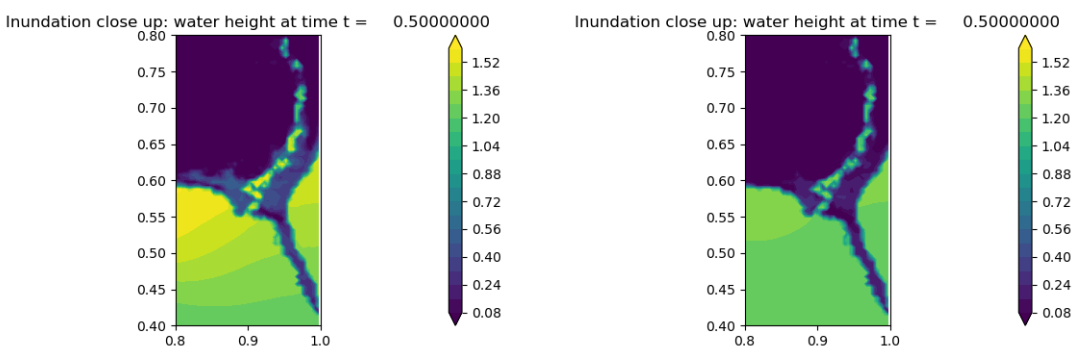


Figure 8.8: Zoomed in region. The blue region highlights where there is no water ($h = 0$) and green region sea surface level (reverse coloring code to highlight inundation).



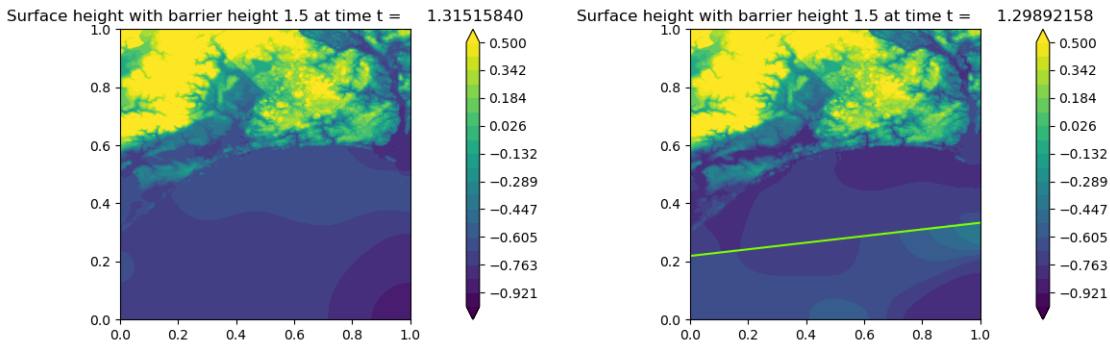
(a) No barrier

(b) With barrier

Figure 8.9: Inland delta region flooding.

8.4 New York City

Finally, we provide our most physical simulation, with the gravitational constant $g = 9.8$ and the bathymetry of NYC region being mostly unprocessed (Fig. 8.12). The only processing done on the real bathymetric data is to rotate the data slightly and flatten the bottom ocean floor so that we can place our barrier on level surface. We zoom in towards the lower Manhattan and nearby bay region using cubic interpolation to fit to our 200×800 grid and perform our simulation. The degrees are turned into distances in km, with the lower left corner being set at $(0, 0)$, the distance



(a) No barrier (b) With barrier

Figure 8.10: Overall difference between barrier and no barrier. The time steps have some discrepancy between the two as the algorithm did not output exact times due to high CFL restrictions in the drying and wetting regions.

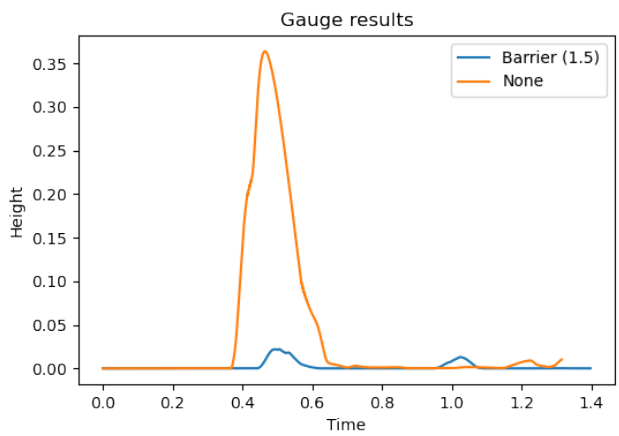
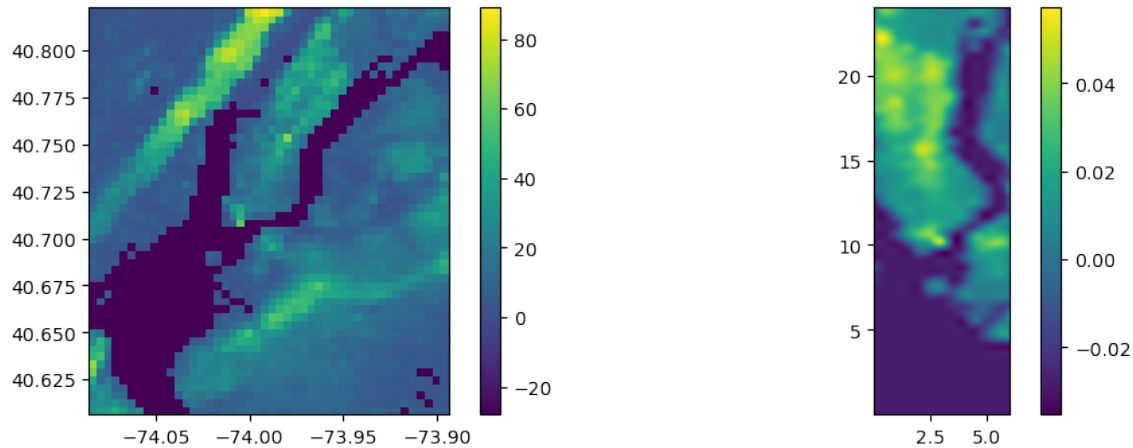


Figure 8.11: Gauge results: blue line showing results with barrier, orange showing without. About 93% protection.

in x direction is 6 km and in the y direction, 24 km. This gives us a resolution of 6/200 km or a 30 m ×30 m cell.

As the units of our spatial dimension are in km, the ocean floor is also set at -35/1000 km, which is the lowest point in the raw bathymetry. We set the dam break jump to be 2.5 meters, which starts from $y = 0.0$ to $y = 0.6$. Admittedly, this is a big dam break, but to highlight the effect of barrier we initialize as such. The barrier height ℓ is set at 1.8 meters above the sea surface located at points (0,1.5) to (6,2.9).

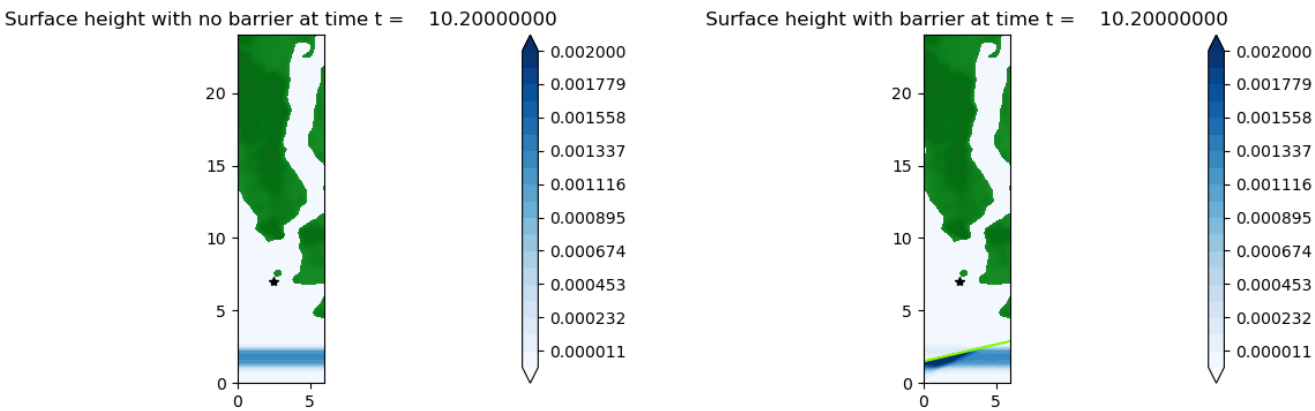
At the barrier, we observe clear abatement of the wave as shown in Fig. 8.13. Just below the tip



(a) Raw bathymetric data

(b) Bottom-flattened, rotated (35°), and zoomed

Figure 8.12: GEBCO bathymetric data over NYC. Left: the raw data with bathymetry given in meters and spatial dimension in latitude/longitude. Right: negative bathymetries flattened using the minimum point; depth and space turned into kilometers; and grid rotated to allow placement of barrier.



(a) No barrier

(b) With barrier 1.8 m above sea surface

Figure 8.13: Dam break simulation over lower Manhattan region with $\Delta h = 2.5\,m$ and $\Delta x = 30m = \Delta y$. Surface height above sea level (0.0) is shown in km. The reduction in wave height in the overtopped wave can be clearly seen. Black asterisk indicates gauge point.

of lower Manhattan, we check the two results by comparing the gauge heights. We put our gauge out in the water, instead of on land, because the realistic bathymetry makes it difficult to find a point in space where we can observe the state going from dry to wet, whereas choosing such a point in the normalized bathymetric studies was relatively straightforward. Our gauge results show about 1.246 m reduction or about 90.9% blockage (Fig. 8.14), which is more than a simple naive

estimate of 1.8/2.5 = 72%. The curve is also flattened, showing a delaying effect of the barrier on the wave:

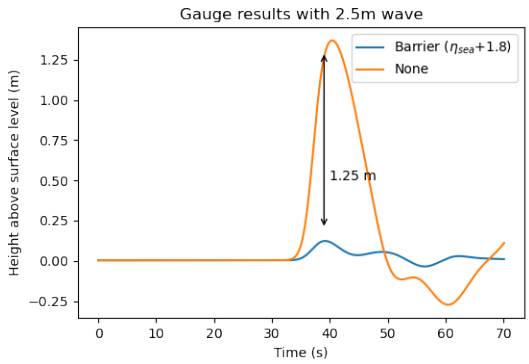
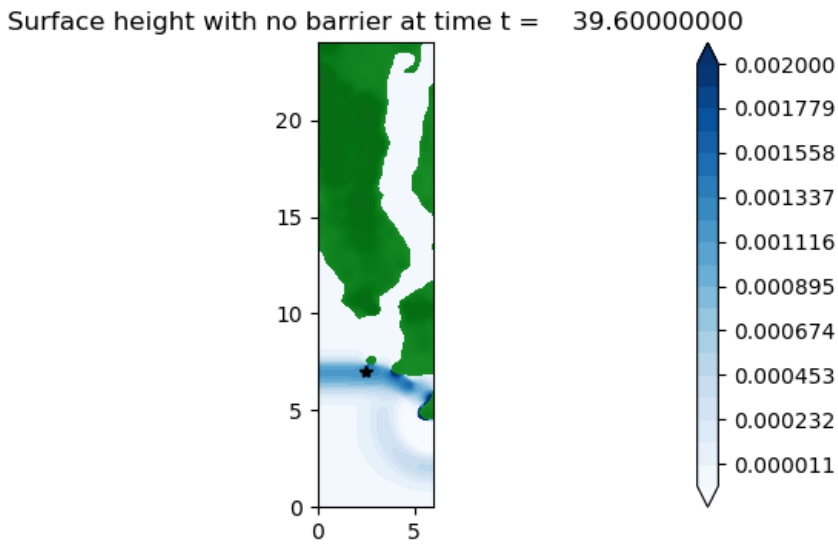
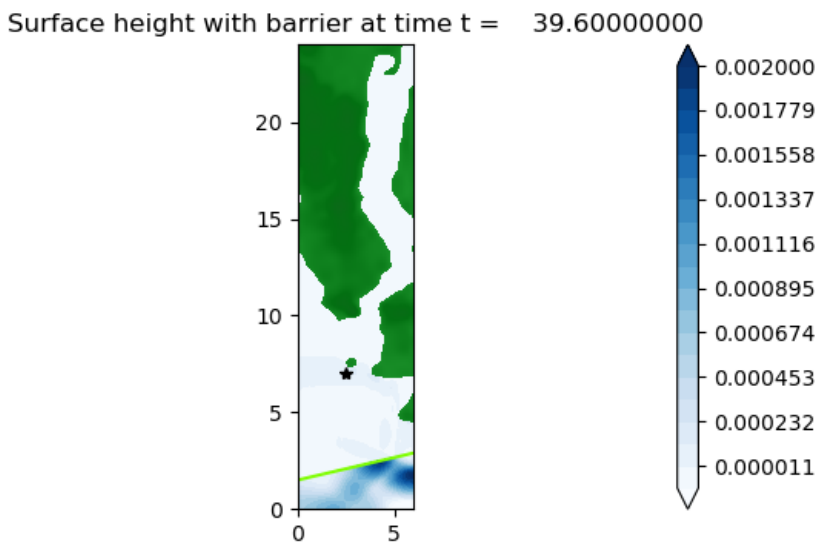


Figure 8.14: Comparison of gauge point off lower Manhattan (2.5,7.0).

We can see how in the no barrier case, the wave almost engulfs Governor's Island while bouncing off from Brooklyn coast (Fig. 8.15) at $t = 39.6s$. However, with the barrier, there is a just slight hump of wave (in light blue) below Governor's Island.



(a) No barrier



(b) With barrier 1.8 m above sea surface

Figure 8.15: Note the difference in waves of two results at the small island (Governor's Island) off lower Manhattan.

Conclusion

Summary of Contributions

In this thesis we have come up with three numerical methods to solve zero width barrier problems in 1D and 2D shallow water equations with varying bathymetry. We tested our methods on two 2D model problems, the linear and V barriers.

For h-box method we were able to only do the slanted, wall-like barrier problem in which only reflection was observed. This was because it was not feasible to find h-box averages that crossed the barrier, which would be required to apply flux-canceling updates to the cut cells. Furthermore, because of ambiguity in h-box updates in the V barrier case (at the tip), h-box method was not suitable to treat such cases. However, we found an alternative simpler grid h-box method that worked on reflection-only cases which did not require finding all possible h-boxes, as done in the original 2D h-box method.

For the state redistribution and cell merging methods we were able to forego using h-box like averages to compute barrier fluctuation and therefore able to treat the more complicated V barrier problem.

To validate our results on the model problem we compared them with mapped grid results, which only used wave redistribution at the barrier edge. We observed greater than one order of convergence on all model results, and second order convergence in the second order cell merging method. We were able to come up with second order cell merging method using a hybrid of wave propagation second order correction terms and gradient approximating and limiting at the barrier

edge.

Finally for realistic example, we computed a wave running up against linear and V barriers in front of a Gaussian island. Also we were able to download real life bathymetric data from GEBCO and run realistic tsunami-like scenarios to observe the effect of the barrier on downscaled South Carolina coast bathymetry and real-scaled New York City.

Besides the cut cell methods, the new contributions of this thesis are summarized below:

- development of 2D wave redistribution
- computation of mapped grid solutions to barrier problems
- implementation of drying-wetting solver in PyClaw
- development of hybrid second order correction using gradient and wave limiting
- use of real (scaled) bathymetric data to compute simulations in PyClaw
- Python-Fortran module that automates cut-cell data generation given barrier

Comparison of the methods

As a way of concluding the thesis, we also compare the three numerical methods, their strengths and drawbacks. We provide a table summarizing our comparison of the three methods in Table 8.1.

Method	Algorithm Description	Cases Implemented	Δt_{avg} factor
HB	Lot of complex geometrical calculations (3)	only RF, linear	2.0
SRD	simple to implement but non-intuitive (2)	both OT and RF, linear and V	5.6
CM	most straightforward to implement (1)	both OT and RF, linear and V	5.2

Table 8.1: RF: reflection, OT: overtopping. Comparisons across all three methods. Ranking is provided for algorithm simplicity, with 1 being the simplest, and for computational savings, 'Δt_{avg} factor' shows the ratio of Δt_{avg} of cut cell method and GeoClaw .

We note that SRD provided the most computational savings in terms of timesteps saved, with CM coming in second and h-box (HB) coming in at last. This is most likely due to the extra stabilization added from the SRD's overlapping neighborhood averaging. HB only has the stabilization from extending one mesh length from the cut cell edge. However, we note that cell merging is much simpler conceptually than SRD and HB. HB method, in addition to being complex with much geometrical computations, does not handle overtopping barrier examples and is not suitable for zero width barrier simulations. SRD and CM both handle reflection and overtopping scenarios.

A side comment on comparison between SRD and CM [12] is that in 3D, SRD has the clear advantage over CM because overlapping neighborhoods are allowed, which allows for more flexible neighborhood search in the higher dimension. CM has to find unique, non overlapping neighborhoods. However, in our 2D cases, no such (dis)advantage appears between SRD and CM. The modeling advantage is that in SRD, the cells in the neighborhood are treated distinctly for the final update, whereas in CM they are merged as the name implies. This inherently brings in more diffusion in CM, although such difference is not noticeable in our examples. On the other hand, SRD technically does have slightly higher computational complexity than CM, as it requires the extra calculation of using overlap counts for the final update, whereas CM only requires one calculation of neighborhood averaging. This is not contradictory to the ranking of computational savings shown in Table 8.1, as by computational savings we mean how much relaxation of CFL condition each method provides.

Engineering conclusions

From our Gaussian island scenario with both the linear and V barrier, there is a design aspect to consider. Given the same height, the shape of the barrier does affect wave height reduced; the linear barrier pushes aside the incoming wave along the direction of the barrier laterally, whereas the V barrier gathers the water to the center and creates a larger overtopping effect, as water pinches up.

On the other hand, the V barrier evenly distributes the wave radially from the tip of the barrier

to the other side. This means that if there were another flood-risk region toward the end of the linear barrier, the barrier would push all the water to that region, whereas only the regions behind the barrier would mostly be protected. This will not be the case with the V barrier due to its even distribution of both reflected and overtopped wave. Also from our NYC simulation, we found that a 1.8 m linear barrier would approximately block about 90% of a 2.5 m wave.

Future Direction

Our work can be of help in modeling storm barriers if further scaled up to handle geophysical shallow water equations (e.g. Coriolis force and storm wind model). Furthermore, our algorithms can be tested in more complicated barrier model problems, such as a circular barrier. There is also the limitation that the barrier has to be from one end of domain to the other, but one can zoom in and crop the desired bathymetry such that the barrier would fit in the selected domain. The issue with completely interior-only barrier is that one has to splice the upper cut cell values and the lower cut cell values appropriately in different regions of the domain. Furthermore, our model requires that we have a level surface on which the barrier can stand. A more involved cut cell method using both bathymetric and surface level interpolation would be required to deal with such cases. To do more flexible, customizable simulations, it would be required to develop a more general model that takes a shape of a barrier as input and applies the cut cell method to solve the barrier problems.

References

[1] J. A. Church and N. J. White, "Sea-level rise from the late 19th to the early 21st century," *Surveys in geophysics*, vol. 32, no. 4, pp. 585–602, 2011.

[2] U. A. Corps of Engineers, "Hurricane sandy aftermath: Hurricane barriers managed by corps engineers in new england prevent 29.7 million in damages," *New England District*, 2012.

[3] Y. Miura, K. T. Mandli, and G. Deodatis, "High-speed gis-based simulation of storm surge–induced flooding accounting for sea level rise," *Natural Hazards Review*, vol. 22, no. 3, p. 04 021 018, 2021.

[4] Y. Miura *et al.*, "A methodological framework for determining an optimal coastal protection strategy against storm surges and sea level rise," *Natural Hazards*, 2021.

[5] M. E. Kress, A. I. Benimoff, W. J. Fritz, C. A. Thatcher, B. O. Blanton, and E. Dzedzits, "Modeling and simulation of storm surge on staten island to understand inundation mitigation strategies," *Journal of Coastal Research*, no. 76 (10076), pp. 149–161, 2016.

[6] R. Leveque, D. George, and M. Berger, "Tsunami modelling with adaptively refined finite volume methods," *Acta Numerica*, vol. 20, pp. 211–289, Apr. 2011, Copyright: Copyright 2011 Elsevier B.V.

[7] M. Berger and C. Helzel, "A simplified h-box method for embedded boundary grids," *SIAM Journal on Scientific Computing*, vol. 34, no. 2, A861–A888, 2012.

[8] M. J. Berger and R. J. LeVeque, "Stable boundary conditions for cartesian grid calculations," INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE and ENGINEERING HAMPTON VA, Tech. Rep., 1990.

[9] M. Berger, "Cut cells: Meshes and solvers," in *Handbook of Numerical Analysis*, vol. 18, Elsevier, 2017, pp. 1–22.

[10] R. J. LeVeque, *Finite volume methods for hyperbolic problems*. Cambridge university press, 2002, vol. 31.

[11] S. Bayyuk, K. Powell, and B van Leer, "An algorithm for the simulation of 2-d unsteady inviscid flows around arbitrarily moving and deforming bodies of arbitrary geometry," in *AIAA 1lth Computational Fluid Dynamics Conference Proceedings*, 1993.

[12] M. Berger and A. Giuliani, "A state redistribution algorithm for finite volume schemes on cut cell meshes," *Journal of Computational Physics*, p. 109 820, 2020.

[13] I.-L. Chern and P. Colella, "A conservative front tracking method for hyperbolic conservation laws," *LLNL Rep. No. UCRL-97200, Lawrence Livermore National Laboratory*, 1987.

[14] M.-H. Chung, "Cartesian cut cell approach for simulating incompressible flows with rigid bodies of arbitrary shape," *Computers & Fluids*, vol. 35, no. 6, pp. 607–623, 2006.

[15] D. Hartmann, M. Meinke, and W. Schröder, "A cartesian cut-cell solver for compressible flows," in *Computational Science and High Performance Computing IV*, E. Krause, Y. Shokin, M. Resch, D. Kröner, and N. Shokina, Eds., Berlin, Heidelberg: Springer, 2014, pp. 363–376.

[16] D. M. Ingram, D. M. Causon, and C. G. Mingham, "Developments in cartesian cut cell methods," *Mathematics and Computers in Simulation*, vol. 61, no. 3-6, pp. 561–572, 2003.

[17] R. J. LeVeque, "Large time step shock-capturing techniques for scalar conservation laws," *SIAM Journal on Numerical Analysis*, vol. 19, no. 6, pp. 1091–1109, 1982.

[18] S. May and M. Berger, "An explicit implicit scheme for cut cells in embedded boundary meshes," *Journal of Scientific Computing*, vol. 71, no. 3, pp. 919–943, 2017.

[19] R. B. Pember, J. B. Bell, P. Colella, W. Y. Curtchfield, and M. L. Welcome, "An adaptive cartesian grid method for unsteady compressible flow in irregular regions," *Journal of computational Physics*, vol. 120, no. 2, pp. 278–304, 1995.

[20] D. M. Causon, D. M. Ingram, C. G. Mingham, G Yang, and R. V. Pearson, "Calculation of shallow water flows using a cartesian cut cell approach," *Advances in water resources*, vol. 23, no. 5, pp. 545–562, 2000.

[21] S. K. Martin, G. Savant, and D. C. McVan, "Lake borgne surge barrier study," ENGINEER RESEARCH, DEVELOPMENT CENTER VICKSBURG MS COASTAL, and HYDRAULICS LAB, Tech. Rep., 2010.

[22] S. S. Barrier, "Environmental impact of tidal energy plant in eastern scheldt," 2018.

[23] A. Niewiarowski, S. Adriaenssens, and R. Pauletti, "A parametrized design space for pneumatic flood barriers," in *IASS Symposium 2019 - 60th Anniversary Symposium of the International Association for Shell and Spatial Structures; Structural Membranes 2019 - 9th International Conference on Textile Composites and Inflatable Structures, FORM and FORCE*, C. Lazaro, K.-U. Bletzinger, and E. Onate, Eds., ser. IASS Symposium 2019 - 60th Anniversary Symposium of the International Association for Shell and Spatial Structures; Structural Membranes 2019 - 9th International Conference on Textile Composites and Inflatable Structures, FORM and FORCE, International Center for Numerical Methods in Engineering, 2019, pp. 882–889.

[24] K. Zhang, Y. Li, H. Liu, J. Rhome, and C. Forbes, "Transition of the coastal and estuarine storm tide model to an operational storm surge forecast model: A case study of the florida coast," *Weather and forecasting*, vol. 28, no. 4, pp. 1019–1037, 2013.

[25] J. M. Torres *et al.*, "Modeling the hydrodynamic performance of a conceptual storm surge barrier system for the galveston bay region," *Journal of Waterway, Port, Coastal, and Ocean Engineering*, vol. 143, no. 5, p. 05 017 002, 2017.

[26] C. P. Jelesnianski, *SLOSH: Sea, lake, and overland surges from hurricanes*. US Department of Commerce, National Oceanic and Atmospheric Administration . . ., 1992, vol. 48.

[27] D. A. Calhoun, C. Helzel, and R. J. LeVeque, "Logically rectangular grids and finite volume methods for pdes in circular and spherical domains," *SIAM review*, vol. 50, no. 4, pp. 723–752, 2008.

[28] R. Leveque, D. George, and M. Berger, "Tsunami modelling with adaptively refined finite volume methods," *Acta Numerica*, vol. 20, pp. 211 –289, May 2011.

[29] J. Li, *An h-box Method for Shallow Water Equations*. Columbia University, 2019.

[30] D. L. George and R. J. LeVeque, "Finite volume methods and adaptive refinement for global tsunami propagation and local inundation," Ph.D. dissertation, University of Washington, Seattle WA, 2006.

[31] D. S. Bale, R. J. Leveque, S. Mitran, and J. A. Rossmanith, "A wave propagation method for conservation laws and balance laws with spatially varying flux functions," *SIAM Journal on Scientific Computing*, vol. 24, no. 3, pp. 955–978, 2003.

[32] B. Einfeldt, "On godunov-type methods for gas dynamics," *SIAM Journal on Numerical Analysis*, vol. 25, no. 2, pp. 294–318, 1988.

[33] D. L. George, "Augmented riemann solvers for the shallow water equations over variable topography with steady states and inundation," *Journal of Computational Physics*, vol. 227, no. 6, pp. 3089–3113, 2008.

[34] P. L. Roe, "Approximate riemann solvers, parameter vectors, and difference schemes," *Journal of computational physics*, vol. 43, no. 2, pp. 357–372, 1981.

[35] M. J. Berger, C. Helzel, and R. J. LeVeque, "H-box methods for the approximation of hyperbolic conservation laws on irregular grids," *SIAM Journal on Numerical Analysis*, vol. 41, no. 3, pp. 893–918, 2003.

[36] R. J. LeVeque, "Wave propagation algorithms for multidimensional hyperbolic systems," *Journal of Computational Physics*, vol. 131, pp. 327–353, 1997.

[37] J. Li and K. T. Mandli, "An h-box method for shallow water equations including barriers," *SIAM Journal on Scientific Computing*, vol. 43, no. 2, B431–B454, 2021.

[38] D. Calhoun and R. J. LeVeque, "A cartesian grid finite-volume method for the advection-diffusion equation in irregular geometries," *Journal of Computational Physics*, vol. 157, no. 1, pp. 143–180, 2000.

[39] N. Gokhale, N. Nikiforakis, and R. Klein, "A dimensionally split cartesian cut cell method for hyperbolic conservation laws," *Journal of Computational Physics*, vol. 364, pp. 186–208, 2018.

[40] J. Ashbourn, L. Geris, A Gerisch, and C. Young, "Numerical simulation of two-dimensional and three-dimensional axisymmetric advection–diffusion systems with complex geometries using finite-volume methods," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 466, no. 2118, pp. 1621–1643, 2010.

[41] M. J. Berger, D. L. George, R. J. LeVeque, and K. T. Mandli, "The geoclaw software for depth-averaged flows with adaptive refinement," *Advances in Water Resources*, vol. 34, no. 9, pp. 1195 –1206, 2011.

[42] M. Berger, "A note on the stability of cut cells and cell merging," *Applied Numerical Mathematics*, vol. 96, pp. 180–186, 2015.

[43] T. Smith, M. Barone, R. Bond, A. Lorber, and D. Baur, "Comparison of reconstruction techniques for unstructured mesh vertex centered finite volume schemes," in *18th AIAA Computational Fluid Dynamics Conference*, 2007, p. 3958.

[44] A. A. Heydari, S. S. Sindi, and M. Theillard, "Conservative finite volume method on deforming geometries: The case of protein aggregation in dividing yeast cells," *Journal of Computational Physics*, vol. 448, p. 110 755, 2022.

[45] M.-H. Chung, "Cartesian cut cell approach for simulating incompressible flows with rigid bodies of arbitrary shape," *computers & fluids*, vol. 35, pp. 607–623, 2006.

[46] D. Calhoun and R. J. LeVeque, "A cartesian grid finite-volume method for the advection-diffusion equation in irregular geometries," *Journal of Computational Physics*, vol. 157, no. 1, pp. 143–180, 2000.

[47] S. C. Medeiros and S. C. Hagen, "Review of wetting and drying algorithms for numerical tidal flow models," *International journal for numerical methods in fluids*, vol. 71, no. 4, pp. 473–487, 2013.

[48] S. Bi, J. Zhou, Y. Liu, and L. Song, "A finite volume method for modeling shallow flows with wet-dry fronts on adaptive cartesian grids," *Mathematical problems in Engineering*, vol. 2014, 2014.

[49] S. J. Williams and N. Ismail, “Climate change, coastal vulnerability and the need for adaptation alternatives: Planning and design examples from egypt and the usa,” *Journal of Marine Science and Engineering*, vol. 3, no. 3, pp. 591–606, 2015.

[50] L. Mayer *et al.*, “The nippon foundation—gebco seabed 2030 project: The quest to see the world’s oceans completely mapped by 2030,” *Geosciences*, vol. 8, no. 2, p. 63, 2018.

Appendix A: Algorithms for cut cell data

We used Fortran to generate cut cell data. The tasks for cut cell data generation was to (1) find the index location of the cut cells, (2) find their edge lengths, (3) find their area, and (4) find their centroids (needed for second order methods.) We provide the pseudocode here; the actual codes can be found on the author's GitHub account here.

A.1 Indices

To find indices we do the following:

```
cut_cell_find:
1. Find slope and angle of barrier using coordinate specification
2. Get intersections between barrier and grid using slope and angle
3. Remove any duplicate intersections due to floating point error
4. Walk along the intersections to determine the index location
```

The biggest challenge was the floating point error sensitivity, where artificial intersections were being found 'between' a point.

A.2 Edges

To find the edge lengths of each cut cell, we do the following:

```
coords:
1. Find edge vertices using index location of cut cell
2. Find coordinates of barrier intersections
3. Find respective distances between coordinates
```

A.3 Areas

To find area of each cut cell, we do the following:

```
area_polygon:
1. Find the vertices of the cut cell
2. Order them in counterclockwise fashion
3. Apply the shoelace formula
```

The Shoelace formula is

$$A = \frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1} y_i).$$

A.4 Centroids

To find centroid of each cut cell, we do the following:

```
find_centroid:
1. Find vertices of cut cell
2. Order them in counterclockwise fashion
3. Apply centroid formula
```

The centroid formula is:

$$C_x = \frac{1}{6A}\sum_{i=0}^{n-1}(x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

$$C_y = \frac{1}{6A}\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1} y_i).$$

Appendix B: Fortran-Python Barrier Module

Using f2py we imported the Fortran library to Python and made an Python object called barrier.

```
class barrier():
    def __init__(self,type,p0,p1):
        os.system('python fortran.py') # f2py library called CutFind
        self.type = type # L or V
        self.p0 = p0 # the first coord of barrier
        self.p1 = p1 # the second or middle coord

    def make(self,mx,my,dx,dy):
        if self.type == 'L':
            CutFind.check_aux_cmL(mx,my,self.p0[0],self.p0[1],\
            self.p1[0],self.p1[1],dx,dy)
        elif self.type = 'V':
            CutFind.check_aux_cmV(mx,my,self.p0[0],self.p0[1],\
            self.p1[0],self.p1[1],dx,dy)
        else:
            raise ValueError('Type not recognized')
```

The object has attributes of type of barrier and the coordinates indicating their location. It also has a method called 'make' which actually runs the Fortran codes to compute the cut cell data, given grid dimensions. This allows us to do all our computation in PyClaw (Python).