Riemann problems for the two-dimensional shallow water equations

EPS 521B, Geophysical Fluid Dynamics

Jonas Katona

November 15, 2022

1 Introduction

For researchers throughout the fields of science and engineering, e.g., in earth sciences, civil engineering, geophysics, and hydrology, the flow of water over a surface or the motion of bodies of water are of great interest with countless applications. These include river flows, tidal waves, channel flows, dam breaches, or even the motion of water in one's own bathtub. [Est+00; CVZ03; KCY07] Since the Navier-Stokes equations (or Euler equations for that matter) for general fluid flows are infamously complicated, quite cumbersome, and usually intractable to solve in their full generality, one usually uses some approximate form of these equations to study water flow in applications, such that the dynamics of the system can still be modeled to a desirable accuracy with much less effort needed to solve them either numerically or analytically (but usually the former). For modeling flows in bodies of water, some frequently encountered equations that can be derived from Navier-Stokes include the Korteweg–De Vries (KdV) equation, nonlinear Schrödinger equation, Burgers' equation, and the shallow-water equations, along with a variety of linearizations and further approximations to these equations. For this paper, we will be concerned with the lattermost set of equations.

The shallow water equations can be derived from the Navier-Stokes equations. We already went over this derivation during lecture; the central, underlying assumption we made was that the vertical length scale, H, is much less than the horizontal length scale, L, of the flow, i.e., $H/L \ll 1$. After several scaling analyses, implications, and computations which follow from this, we thereby derive the shallow-water equations:

$$\begin{cases} \partial_t h + \nabla \cdot (h\mathbf{u}) = 0, \\ \partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} = -g\nabla h, \end{cases} \tag{1}$$

where $h = h\,(x, y, t)$ is the height of the fluid, $\mathbf{u} = (u, v)$ are the x- and y-velocities, respectively, of the fluid, and g is the (local) acceleration of gravity.[1] Furthermore, for larger-scale flows (i.e., those where the horizontal length scale is comparable to the Rossby radius of deformation), we cannot neglect the effects of Earth's rotation on the flow. Hence,

[1]Note that x and y are coordinates for a local, orthonormal coordinate system somewhere on Earth's surface.

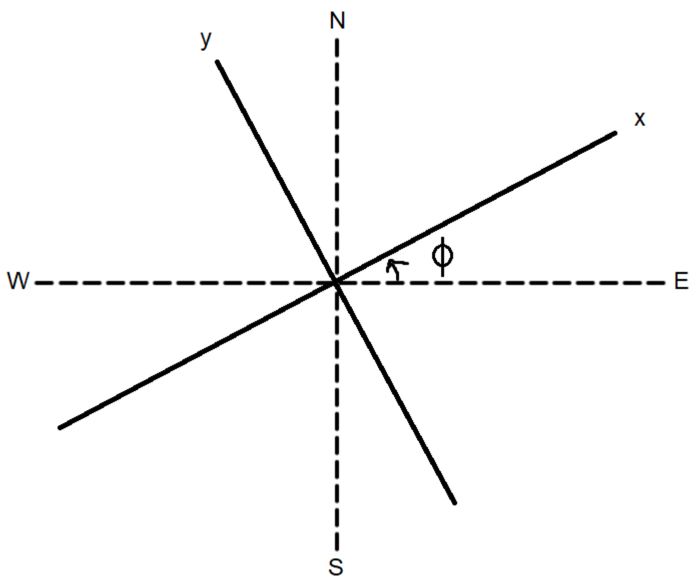

Figure 1: A schematic showing the relationship of the local coordinate system (x, y) with the four cardinal directions and ϕ.

we must consider the inertial force terms which must be added to the momentum equation in (1), such that (1) becomes

$$\begin{cases} \partial_t h + \nabla \cdot (h\mathbf{u}) = 0, \\ \partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} + \mathbf{f} \times \mathbf{u} = -g\nabla h. \end{cases} \tag{2}$$

We can use the beta plane approximation to express the Coriolis parameter as $\mathbf{f} = \mathbf{f}_0 + \beta d = 2\Omega \sin\theta + 2\,(d/a)\,\Omega\cos\theta$, where d is the meridional distance (distance in the N-S direction) from the (fixed) latitude θ which crosses through the origin $(0, 0)$ of our local coordinates, Ω is Earth's rotation frequency, and a is the Earth's (local) radius. Finally, we can also express (2) in so-called conservation (or flux) form:

$$\begin{cases} \partial_t h + \partial_x U + \partial_y V = 0, \\ \partial_t U + \partial_x \left(\frac{U^2}{h} + \frac{1}{2}gh^2\right) + \partial_y \left(\frac{UV}{h}\right) = fV = \left[\sin\theta + \frac{1}{a}\left(x\sin\phi + y\cos\phi\right)\cos\theta\right] 2\Omega V, \\ \partial_t V + \partial_x \left(\frac{UV}{h}\right) + \partial_y \left(\frac{V^2}{h} + \frac{1}{2}gh^2\right) = -fU = -\left[\sin\theta + \frac{1}{a}\left(x\sin\phi + y\cos\phi\right)\cos\theta\right] 2\Omega U, \end{cases} \tag{3}$$

where $U = hu$, $V = hv$, and ϕ is the (fixed) angle relative to the local vertical (see Figure 1). (3) is called a conservation form because it reflects the flux of conserved quantities, as we can see by integrating both sides of (3) (for $f = 0$) over some control volume and applying the divergence theorem. Namely, we still have conservation of mass (the top equation) and conservation of linear momentum (the bottom two equations), which one already imposes when deriving the Navier-Stokes equations.

The shallow water equations (2)/(3) are certainly simpler than the full Navier-Stokes equations or the Euler equations; in particular, they do not have any dependence on thermodynamic variables. Nevertheless, they can seldom be solved exactly. Furthermore, as we will explore in this paper, (2)/(3) often exhibit dynamics which are still computationally difficult to deal with, e.g., discontinuous solutions, solutions with corners or discontinuous derivatives, or with turbulence that necessitate fine spatiotemporal resolution and specialized numerical schemes. [LeV02; Tor99] In the modeling of real-world scenarios, one could also consider the effects of friction on (often moving) boundaries as well as topography, which only add further complications.

Using the code which we will discuss in Section 3 of this project, we demonstrate a solution to the full shallow-water equations which develops into a shock wave over time. Consider (2)/(3) with the Gaussian initial conditions

$$h\left(x,y,0\right)=H+\frac{H}{5}\left[\exp\left(-\frac{400x^2}{D^2}\right)-1\right],\quad u\left(x,y,0\right)=v\left(x,y,0\right)=0, \tag{4}$$

and boundary conditions discussed in Sections 3.1-3.2. We integrate the initial conditions (4) forward in time from $t=0$ to $t=\left(8\right)\left(60\right)\left(60\right)$ seconds $=8$ hours using the parameters $\theta=-30^\circ$, $\phi=15^\circ$, $a=6360\times10^3$ m, $g=9.80665$ m/s^2, $\Omega=7.2921\times10^{-5}$ rad/s, $H=50$ m, $D=200$ km, $L=1000$ km, $\Delta t=2$ s, $\Delta x=5L\times10^{-3}$, and $\Delta y=5D\times10^{-3}$. The results of this simulation can be found here. The initial Gaussian profile spreads away from the line $x=0$ over time, leaving behind a middle region with height and velocities that are both smaller in magnitude than in the expanding fronts going in both directions in x. But as these two fronts travel outwards, they gradually steepen into two shocks moving away from each other in all variables. Furthermore, since we are at a latitude of -30° and our channel is facing (in positive x and y) $90^\circ-15^\circ=75^\circ$ to the northeast, we know that the Coriolis force will gradually cause our waves to veer slightly off course from just going straight down the channel. Indeed, we see this influence; as the simulation progresses, v slowly grows and eventually develops into a Kelvin wave structure. By the end of the simulation, the height is mostly flat yet slanted to balance the Coriolis force. Meanwhile, the velocity profiles indicate that fluid parcels traveling rightwards (NE) along the channel will tend to be pushed towards the bottom of the channel and further up, while fluids parcels traveling leftwards (SW) will tend to be pushed towards the top of the channel and further down, but in either case, these fluid parcels will oscillate (i.e., join a Kelvin wave) near their respective walls of the channel as they gradually propagate downstream or upstream, respectively. This gradual evolution towards Kelvin waves and geostrophic balance is a key feature that we will observe throughout the Riemann problem solutions we test for the 2D shallow-water equations.

We already dealt with the linearized shallow water equations in lecture and on problem sets. The solutions in the linearized case are also solutions for (2)/(3) in a sense, i.e., they are relatively close to certain exact solutions for the full, nonlinear shallow water equations under certain assumptions. Hence, one larger goal of this project is to analyze some of the most basic solutions which only the full, nonlinear equations could exhibit: solutions to Riemann problems. We will define what a Riemann problem is in Section 2.1.

2 The shallow-water equations in 1D

Suppose for now that we have no velocity in the y-direction ($v=0$).[2] Then, for this to always be the case at every latitude, (2)/(3) show that we must also neglect the effects of rotation, i.e., $\mathbf{f}=0$, lest $\partial_t V\neq0$. Hence, in this case, (3) becomes

$$\begin{cases}\partial_t h+\partial_x\left(hu\right)=0,\\ \partial_t\left(hu\right)+\partial_x\left(hu^2+\frac{1}{2}gh^2\right)=0.\end{cases} \tag{5}$$

[2]This is a decent approximation for low-velocity flows in channels which only undergo negligible friction with the outsides of the channel, as well as surface runoff. [CST19; ELT20]

(5) is also called the Saint-Venant equations; these are also often modified by adding an extra term to the momentum equation that models wall shear stresses, but we will ignore these for sake of simplicity. [Sai71] The linearized version of (5) (valid for small amplitudes only) has wave-like solutions called gravity waves which move at characteristic velocities $u_0 \pm \sqrt{gh_0}$, i.e., with speeds $\pm\sqrt{gh_0}$ relative to the fluid. These are driven by hydrostatic pressure resulting from gravity, *not* acoustic waves (which we are ignoring by assuming incompressibility). For larger amplitude waves, this linearization breaks down, and we are required to consider nonlinearities.

2.1 Riemann problems

A Riemann problem is an initial value problem for conservation equations which consists of piecewise constant initial data that is discontinuous on a closed subset of codimension one[3] (i.e., a point in 1D, line in 2D, surface in 3D or higher) in our domain of interest that divides the domain into two regions. [Tor99; LeV02] For the case of our 1D Saint-Venant equations, since these equations are invariant under translations in both time and space, a general Riemann problem would look like the following:

$$\begin{cases} h(x,0) = h_l, \\ h(x,0)u(x,0) = h_l u_l \end{cases} (x < 0)\,, \quad \begin{cases} h(x,0) = h_r, \\ h(x,0)u(x,0) = h_r u_r \end{cases} (x > 0)\,, \tag{6}$$

where $h_l \neq h_r$, $u_l \neq u_r$, or both. For a general hyperbolic system of m equations, the solution to a Riemann problem generally consists of m waves moving at different speeds, and what we observe in our solution is a superposition of these waves. [LeV02] In particular, the solution to (5) for *any* Riemann problem consists of two waves which may or may not be moving in different directions.

What is especially nice about Riemann problems is that they can be used to “build” more general solutions to conservation laws, because solutions to local Riemann problems do not interact with those starting in other regions until these solutions touch with one another.[4] [Tem15] For instance, if a hyperbolic system has two shock waves that started at two different points in the domain, these shock waves can be evolved separately and independently of one another unless they collide, in which case we must solve another Riemann problem at the point in spacetime where these shocks interact. The point in spacetime where two such solutions interact coincides precisely and equivalently where the characteristic curves for the PDE collide, i.e., curves in (x, t) on which the independent variables are constant.

Two commonly-encountered of waves encountered are shock waves and rarefaction waves. A shock wave does not change its shape with time, and only moves across the domain. However, a rarefaction wave spreads out or *rarifies* as it moves. In fact, these are the only types of waves which are found in solutions to any Riemann problem for the 1D Saint-Venant equations; we will have two waves moving in two directions (possibly the same), each of which is either a shock or a rarefaction wave, with a constant intermediate state between them. [LeV02]

[3]Suppose that we have a submanifold N of a manifold M. Then, the codimension of N is defined to be $\operatorname{codim}(N) = \dim(M) - \dim(N)$.

[4]This is very unlike the behavior of solutions to, say, parabolic PDEs (e.g., the diffusion equation), where disturbances in one part of the domain generally reach any other part of the domain instantaneously.

2.2 Rankine–Hugoniot conditions and shock waves

For a 1D conservation law for a single quantity q of the form $\partial_t q + \partial_x f(q) = 0$, the speed s of a shock wave is given by the Rankine-Hugoniot conditions, or

$$s = \frac{f(q_r) - f(q_l)}{q_r - q_l}, \tag{7}$$

where q_l is the value of q at the left side of the shock and q_r is for the right side.[5] If we want to generalize (7) to a system

$$\partial_t \mathbf{q} + \nabla \cdot (F(\mathbf{q})) \tag{8}$$

of m conserved quantities (where $F : \mathbb{R}^m \to \mathbb{R}^m$ is a vector field), then $F(\mathbf{q}), \mathbf{q} \in \mathbb{R}^m$, and hence, (7) is meaningless since we cannot "divide" vectors in a naïve sense. Instead, we have that

$$F(\mathbf{q}_*) - F(\mathbf{q}) = s(\mathbf{q}_* - \mathbf{q}), \tag{9}$$

where $\mathbf{q}_* = \mathbf{q}_r, \mathbf{q}_l$ and (9) must hold for *all* components of both sides. Since s is a scalar, if we can think of F as being linear, then (9) somewhat looks like an eigenvalue problem. In fact, if $F(\mathbf{q}) = A\mathbf{q}$ for some matrix $A \in \mathbb{R}^{m\times m}$ (i.e., our system of conservation of laws is linear), then (9) becomes $A(\mathbf{q}_* - \mathbf{q}) = s(\mathbf{q}_* - \mathbf{q})$, and we see in fact that s is an eigenvalue of the matrix A. As expected, as long as A is not degenerate, A has m eigenvalues, each of which corresponds to a "possible" shock wave.

In the case of the Saint-Venant equations, (9) becomes

$$s(h_* - h) = h_* u_* - hu, \quad s(h_* u_* - hu) = h_* u_*^2 - hu^2 + \frac{1}{2} g\left(h_*^2 - h^2\right). \tag{10}$$

By using the left equation in (10) to solve for s and then plugging the resultant expression into the right equation, we can eliminate s from the system. After simplifying, we get a quadratic equation for u, which can be solved to obtain two solutions for u:

$$\boxed{u(h) = u_* \pm \sqrt{\frac{g}{2}\left(\frac{h_*}{h} - \frac{h}{h_*}\right)(h_* - h)}.} \tag{11}$$

(11) will be satisfied by *all* values of u and h immediately on either side of the shock. The two roots in (11) correspond to left and right shocks, as long as we associate the correct signs with the left and right states. Namely, h_l and u_l correspond to the negative root, while h_r and u_r corresponds to the positive root. As long as the expression under the radical in (11) is positive — and one can check that it always will be unless $h = h_*$, in which case we just get $u = u_*$ as expected — both shocks solve the Riemann problem and move in their respective directions with speed s.

Furthermore, if we know that a Riemann problem has a "physical" solution (we will talk about what this means in the next section) consisting of a shock, then the intermediate state q_m we talked about earlier will satisfy (11) for $* = l$ or $* = r$, depending on if this shock is on the left ($* = l$) or on the right ($* = r$). In fact, in the situation where we have two shocks moving in opposite directions, since we are given u_l, h_l, u_r, and h_r in the Riemann problem, (11) gives us two equations for two unknowns: u_m and h_m, which take the places

[5] A derivation of (7) can be found in Section 11.8 of [LeV02].

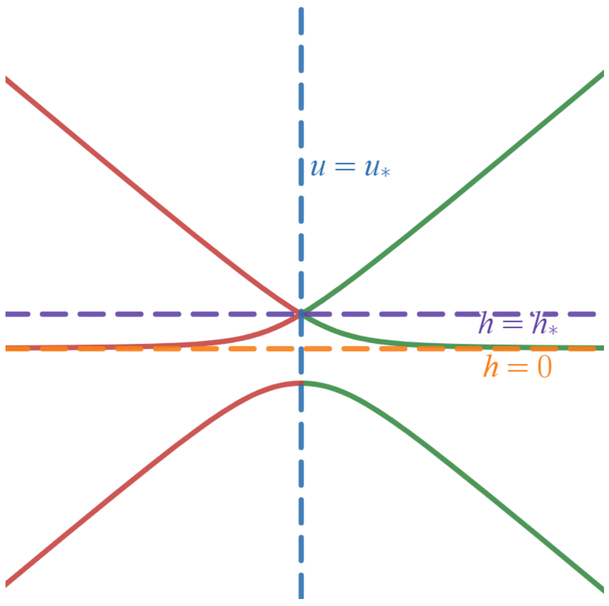

Figure 2: A schematic made in Desmos which shows an example of what the Huguinot loci given by (11) look like in the (u, h)-plane for the same, fixed values of u_* and h_*. The green curve corresponds to right-shocks while the red corresponds to left-shocks.

of u and h, respectively. It turns out that the resultant system cannot be solved explicitly, and necessitates an iterative method. [Tor99; LeV02; KLR20] From here, we can use the Rankine–Hugoniot conditions (10) to solve for the wave speeds $\boxed{s_\pm = u_* \pm \sqrt{g h_m \frac{h_* + h_m}{2h_*}}}$ explicitly, and thereby construct our piecewise solution: A shock to the left moving at speed s_- with value (h_l, u_l), and shock to the right moving at speed s_+ with value (h_r, u_r), and a region in the middle which joins along for the ride with value (h_m, u_m),

2.3 Hugoniot loci

The set of all h and u in the (h, u)-plane (or equivalently, the set of all hu and u in the (hu, u)-plane) form curves called Hugoniot loci. [LeV02; KLR20] Each locus corresponds to a starting condition (h_*, u_*), and (h_m, u_m) must lie on such a locus. Hence, for an all-shock solution, (h_m, u_m) lies on the intersection of the two loci corresponding to (h_l, u_l) and (h_r, u_r). We plot the two "families" of loci, one family for each root, that (11) gives in Figure 2. By moving the lines $u = u_*$ and $h = h_*$ separately to adjust for different left- and right-initial states for the Riemann problem, and then moving the red and green curves, respectively, to account for this accordingly, one can visually imagine how the green and red curves would intersect at the intermediate state (h_m, u_m) for all-shock solutions.

However, we know that when characteristics cross, i.e., where two different values for our PDE solution "originate" from the same point in spacetime, then we know that uniqueness (and possibly even existence, generally speaking) of solutions could be violated. [Eva10] In particular, this holds for Riemann problems for conservation laws. Thus, we need some way to further narrow down our choice of solutions and to cull out physically-unrealistic solution, not to mention that we have not even considered rarefaction waves yet. We need an entropy condition.

2.4 Entropy condition

The following definition is from [LeV02]. Consider the hyperbolic system of PDEs (8) and let DF be the Jacobian of F with ordered eigenvalues $\lambda_1 < \lambda_2 < \cdots < \lambda_m$. Then, a discontinuity separating states $\mathbf{q}_l$ and $\mathbf{q}_r$, propagating at speed s, satisfies the Lax entropy condition if there is an index $p \in \{1, \ldots, m\}$ such that $\lambda_p(\mathbf{q}_l) > s > \lambda_p(\mathbf{q}_r)$ so that p-characteristics (eigenvector corresponding to λ_p) are impinging on the discontinuity (the wave in question associated with the pth eigenvector is traveling with the discontinuity), while the other characteristics are crossing the discontinuity, i.e., $\lambda_j(\mathbf{q}_l), \lambda_j(\mathbf{q}_r) < s$ for $j < p$ and $\lambda_j(\mathbf{q}_l), \lambda_j(\mathbf{q}_r) > s$ for $j > p$.

In particular, for the Saint-Venant equations, the eigenvalues for the Jacobian are $\lambda_\pm = u \pm \sqrt{gh}$ (which should be expected anyways from our derivation for gravity waves). Hence, by the Lax entropy condition and our expression for s earlier in terms of u_*, h_*, u_m, and h_m, one can derive that physically-correct shocks (i.e., those satisfying the Lax entropy condition) must satisfy the inequality $h_m > h_*$. Physically speaking, this means that fluid particles passing through a shock experience a sudden increase in depth. This is akin to the physical entropy condition in gas dynamics, which states that gas particles experience an increase in entropy as they pass through a shock wave. [LeV02] Finally, from Figure 2, we see that entropy-violating shock solutions have values of h which lie below the line $h = h_*$, or in other words, physically-correct shocks will lie on either the green or red curve above the dashed purple line.

But what happens if our values of (h_l, u_l) and (h_r, u_r) are such that our left-shock or our right-shock violates the Lax entropy condition? In that case, to get a physically-correct solution, we must replace that entropy-violating shock with a rarefaction wave.

2.5 Integral curves and rarefaction waves

The method of characteristics is valid for any hyperbolic PDE, but solutions derived via the method of characteristics satisfy certain continuity assumptions which shock waves do not satisfy. For systems of hyperbolic PDEs, the intuition for integrating along characteristics is similar as for applying the method of characteristics for a single hyperbolic PDE; in both cases, we want to parameterize some characteristic curves in phase space upon which our PDE becomes a system of ODEs that can be integrated with respect to this parameterization. [Eva10] For systems of hyperbolic PDEs, instead of having just one invariant, we have a family of invariants which (hopefully) should match the number of independent variables in our system of PDEs. These are called Riemann invariants. [Tor99; LeV02] Each Riemann invariant represents an eigenvector for the Jacobian of our flux function F, since each eigenvector essentially "spans" the characteristic surface upon which the invariant is conserved.

A rarefaction wave is associated with one characteristic surface, one invariant, and one eigenvector. Rarefaction waves can be derived by integrating along the characteristics for a hyperbolic PDE or a system of hyperbolic PDEs. Hence, by what we noted above, since rarefaction wave solutions are derived via integrating along characteristics, we intuitively deduce that these solutions are smooth. They can also be derived by guessing a similarity solution of the form $\mathbf{q}(x,t) = \widetilde{\mathbf{q}}(x/t)$, but it turns out that these two methods are equivalent in a sense, since under this similarity transformation, we are essentially projecting our solution onto a family of integral curves. [LeV02]

More concretely, all values $\widetilde{\mathbf{q}}$ on a rarefaction wave lie on a curve defined by $J\widetilde{\mathbf{q}}(\xi) =$

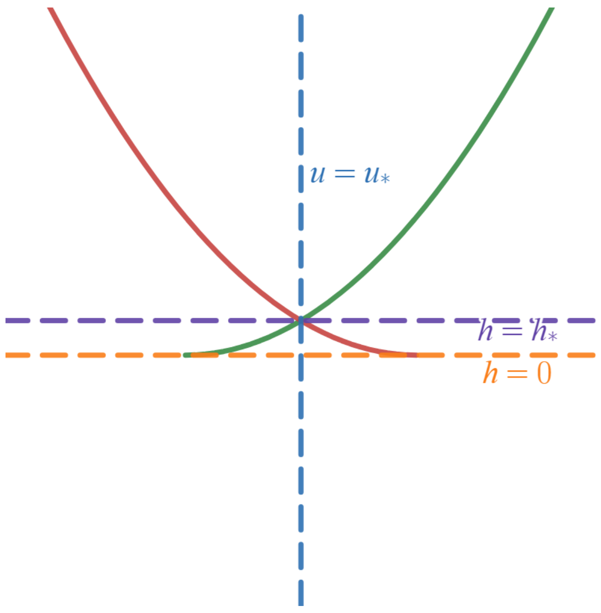

Figure 3: A schematic made in Desmos which shows an example of what the integral curves given by (13) look like in the (u, h)-plane for the same, fixed values of u_* and h_*. The green curve corresponds to right-rarefactions while the red corresponds to left-rarefactions.

$\mathbf{r}_p\left(\widetilde{\mathbf{q}}\left(\xi\right)\right)$. In particular, for the Saint-Venant equations,

$$h'\left(\xi\right) = 1, \quad \left(hu\right)'\left(\xi\right) = u \pm \sqrt{gh}, \tag{12}$$

where the $-$ sign in (12) applies for left waves while the $+$ corresponds to right waves. Hence, since (12) is a system of ODEs, once we have the values of h and u at some point on the integral curve, we have our entire rarefaction wave which we can evolve forward in time according to ξ. But as (12) demonstrates, we do not even need to know what ξ is explicitly to relate h and u. (Although, from the similarity solution approach, we know that $\xi = x/t$.) The first equation in (12) implies that $h - h_* = \xi$ (choose ξ such that $h\left(0\right) = h_*$), whence we can integrate the second directly and obtain that

$$hu = hu_* \mp 2h\left(\sqrt{gh_*} - \sqrt{gh}\right) \Rightarrow \boxed{u \mp 2\sqrt{gh} = u_* \mp 2\sqrt{gh_*}.} \tag{13}$$

(13) implies that $u - 2\sqrt{gh}$ is constant along integral curves corresponding to right waves, while $u + 2\sqrt{gh}$ is constant along integral curves for left waves. Hence, these are the Riemann invariants we talked about earlier which are conserved along the integral curves. As we did in Figure 2 for the Hugoniot loci, we plot the level curves of these invariants in Figure 3.

As expected from Sections 2.2-2.3, there are cases in which imposing that both waves in our solution are rarefaction waves leads to unphysical solutions. In fact, these non-physical situations correspond to cases where our rarefaction waves become multi-valued, since essentially, the rarefaction spreads in a way that it goes underneath or above the solution curve. This further supports how we should replace the rarefactions in these cases with shocks; these shocks essentially represent how such an unphysical rarefaction is "stopped" by the shock, which blocks the solution from becoming multi-valued and propagates this blockage as a discontinuity. Conversely, rarefactions increase the entropy in the system, thereby taking the place of entropy-violating shocks. Section 2.4 also gives us a precise criteria for when we should expect a rarefaction wave: Rarefaction waves satisfy $h_m < h_*$, i.e., left rarefactions occur when $h_m < h_l$ while right rarefactions occur when $h_m < h_r$.

Recall from Sections 2.2-2.3 that a key ingredient to solving a Riemann problem is finding the intermediate state (h_m, u_m). The intermediate state lies on the solution curve that belongs to either a shock, rarefaction, or both. Hence, in cases where we have precisely one rarefaction and one shock, then Figures 2 and 3 show us how to construct a solution pretty quickly: We need to find the intersection of a red curve corresponding to a left shock (or a left rarefaction) and a green curve corresponding to a right shock (or a right rarefaction, respectively), where both curves match our initial left state (h_l, u_l) and our right state (h_r, u_r). As long as these curves intersect in a physically-valid region, i.e., where the shock does not violate the Lax entropy condition, then the intersection gives us our intermediate state. And in cases where they do not, we should try to look for a two-shock solution or a two-rarefaction solution, depending on how h_m compares to h_*.

2.6 Example 1(a): The dam-breaking problem

The dynamics of liquids following the release of some barrier has a multitude of scientific and engineering applications, e.g., in making predictions following natural disasters or dam failures. [VCZ02; ELT20] Intuitively speaking, when water of two different levels at rest is suddenly released, we expect that the water from the higher side will immediately start to spill into the lower side from the bottom, but in the process of doing so, the water will immediately start to drain from the higher side. From a physics perspective, this is because the height difference between the neighboring liquids carries an intrinsic gravitational potential energy, and when the dam breaks, this gravitational potential energy transfers into a cascade of kinetic energy, which is first released from the bottom of the liquid as it moves to the lower side. Why? This is because releasing fluid parcels from the bottom first leads to the greatest decrease in bulk gravitational potential energy over time (principle of least action) throughout the fluid, since by conservation of mass, this will also lower the level of the fluid above it. If fluid had been released from the top of the higher side first, then this would only decrease the gravitational potential energy for those higher fluid parcels, and not the ones below them.

From here, let us confirm that our expectations from the physics match those that we can derive from the mathematics. Without loss of generality, the initial conditions for the Riemann problem in this case can be given as

$$h(x,0) = h_l \quad (x<0)\,, \quad h(x,0) = h_r \quad (x>0)\,, \quad u(x,0) = u_l = u_r = 0 \quad (x \in \mathbb{R}) \tag{14}$$

for $h_l > h_r$. By physical considerations, we should expect a left-moving rarefaction wave and a right-moving shock. The right shock corresponds to the release of kinetic energy due to water flowing into the lower region, while the left rarefaction is caused by the lowering of water due to loss of mass in the higher region. Hence, by (11) and (13), we know that the intermediate state lies on the intersection of the following curves:

$$u_m = u_r + \sqrt{\frac{g}{2}\left(\frac{h_r}{h_m} - \frac{h_m}{h_r}\right)(h_r - h_m)} \Rightarrow u_m^2 = \frac{g}{2}\left(\frac{h_r}{h_m} - \frac{h_m}{h_r}\right)(h_r - h_m)\,, \tag{15}$$

$$u_m + 2\sqrt{gh_m} = u_l + 2\sqrt{gh_l} \Rightarrow u_m = 2\sqrt{g}\left(\sqrt{h_l} - \sqrt{h_m}\right). \tag{16}$$

Unfortunately, (15) and (16) cannot be solved explicitly for u_m and h_m, but given g, h_r, and h_m, they can be dealt with numerically. However, we know that the right shock travels rightward with speed $s = \sqrt{gh_m \frac{h_r+h_m}{2h_r}}$, while the left rarefaction has the form

$$\begin{cases} u_l & x/t \leq -\sqrt{gh_l}, \\ \widetilde{u}\left(x/t\right) & -\sqrt{gh_l} \leq x/t \leq -\sqrt{gh_m}, \\ u_m & x/t \geq -\sqrt{gh_m}, \end{cases} \qquad \begin{cases} h_l & -\sqrt{gh_l}, \\ \widetilde{h}\left(x/t\right) & -\sqrt{gh_l} \leq x/t \leq -\sqrt{gh_m}, \\ h_m & x/t \geq -\sqrt{gh_m}, \end{cases} \tag{17}$$

for some functions $\widetilde{u}$ and $\widetilde{h}$ to be determined, which is valid up to where the right shock starts. The speed for the ends of the rarefaction wave, as given in (17), come from the eigenvalue $\lambda_1 = u - \sqrt{gh}$ (corresponding to left waves) of the Jacobian matrix for the flux F, since the similarity variable $\xi = x/t$ is essentially an eigenvalue for the Jacobian (by analogy with how $J\widetilde{\mathbf{q}}\left(\xi\right) = \mathbf{r}_p\left(\widetilde{\mathbf{q}}\left(\xi\right)\right)$ from earlier). More concretely, we have the condition

$$\lambda_1 = \widetilde{u} - \sqrt{g\widetilde{h}} = x/t \Rightarrow \widetilde{h} = \frac{\left(\widetilde{u} - x/t\right)^2}{g}, \tag{18}$$

whilst the fact that rarefaction waves preserve Riemann invariants implies that

$$\widetilde{u} + 2\sqrt{g\widetilde{h}} = u_l + 2\sqrt{gh_l} = 2\sqrt{gh_l}. \tag{19}$$

Putting (18) and (19) together, we conclude that

$$\boxed{\widetilde{h}\left(x/t\right) = \frac{\left(2\sqrt{gh_l} - x/t\right)^2}{9g}, \quad \widetilde{u}\left(x/t\right) = \frac{2}{3}\sqrt{gh_l} + \frac{2x}{3t}.} \tag{20}$$

Note that (20) implies that the rarefaction wave spreads out quadratically for h while it only spreads out linearly for u.

A sketch of the solution to the dam-break problem appears in Figure 4.

2.7 Example 2(b): The wave-crash problem

Suppose that we have two shock waves traveling towards each other in the water. The left has initial velocity $u_l > 0$ and height $h_l > 0$ while the right has initial velocity $u_r < 0$ and height $h_r > 0$. When the two waves collide, they exchange momentum and start traveling (still, as shocks) in opposite directions by conservation of momentum. In fact, when $h_l = h_r$, then the momentum is only transferred via a change in velocity, and we can check in fact that the following holds via the Saint-Venant formulation:

$$s_+ + s_- = u_r - \sqrt{gh_m \frac{h_r + h_m}{2h_r}} + u_l + \sqrt{gh_m \frac{h_l + h_m}{2h_l}} = u_r + u_l$$
$$-\sqrt{gh_m \frac{h_l + h_m}{2h_l}} + \sqrt{gh_m \frac{h_l + h_m}{2h_l}} = u_r + u_l,$$

i.e., momentum of the shocks is conserved in the usual, discrete sense if we treat the shocks like solid objects colliding elastically.

Let us check that our mathematical solution confirms the physics above. Consider the following Riemann problem:

$$h\left(x,0\right) = h_l = h_r = H \quad \left(x \in \mathbb{R}\right), \quad u(x,0) = u_l > 0 \quad \left(x < 0\right),$$
$$u(x,0) = u_r < 0 \quad \left(x > 0\right). \tag{21}$$

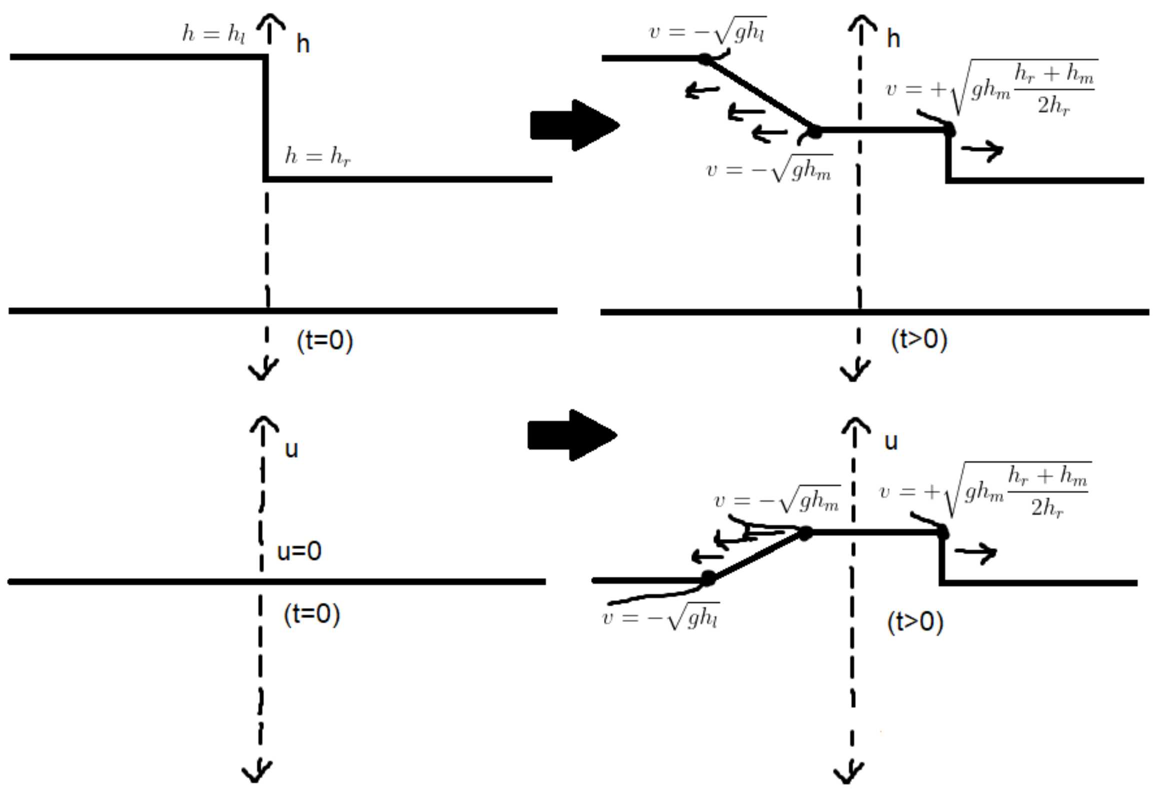

Figure 4: A sketch showing the solution to the dam-break problem, as well as visually how the shock and rarefaction waves look like. The wave speeds are indicated on the plots, as well as the initial velocity and height fields.

We assume that both waves that come from the initial-value problem (21) are shocks by physical considerations. By (11), h_m and u_m must satisfy the following two equations simultaneously:

$$u_m = u_r + \sqrt{\frac{g}{2}\left(\frac{h_r}{h_m} - \frac{h_m}{h_r}\right)(h_r - h_m)} = u_r + \sqrt{\frac{g}{2}\left(\frac{H}{h_m} - \frac{h_m}{H}\right)(H - h_m)}, \tag{22}$$

$$u_m = u_l - \sqrt{\frac{g}{2}\left(\frac{h_l}{h_m} - \frac{h_m}{h_l}\right)(h_l - h_m)} = u_l - \sqrt{\frac{g}{2}\left(\frac{H}{h_m} - \frac{h_m}{H}\right)(H - h_m)}. \tag{23}$$

By adding (22) and (23) together, we see that

$$2u_m = u_r + u_l \Rightarrow \boxed{u_m = \frac{1}{2}(u_l + u_r),} \tag{24}$$

a very intuitive result showing that the velocity of the flow left over after the shocks separate lies in-between their velocities.

It remains to solve for h_m. Plugging in the above expression into (22),

$$\frac{1}{2}(u_l + u_r) = u_r + \sqrt{\frac{g}{2}\left(\frac{H}{h_m} - \frac{h_m}{H}\right)(H - h_m)}$$

$$\Rightarrow (u_l - u_r)^2 = 2g\left(\frac{H}{h_m} - \frac{h_m}{H}\right)(H - h_m) = 2gH\left(\frac{H}{h_m} - \frac{h_m}{H}\right)\left(1 - \frac{h_m}{H}\right). \tag{25}$$

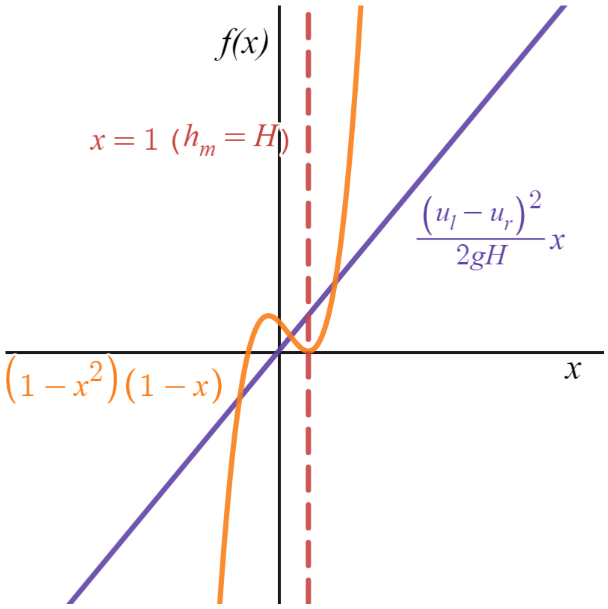

Figure 5: A plot made in Desmos showing the graphs of each function on both sides of the equality in (26), as well as a dashed line showing where $h_m = H$. In particular, the places where the two curves intersect represent roots which solve (26).

Letting $x = h_m/H$, we can rewrite (25) as the following cubic polynomial:

$$\frac{(u_l - u_r)^2}{2gH} = \left(\frac{1}{x} - x\right)(1-x) \Rightarrow \frac{(u_l - u_r)^2}{2gH}x = \left(1 - x^2\right)(1-x)\,. \tag{26}$$

(26) is directly solvable via Cardano's formula for solving cubic equations, but the expressions for all three roots are very messy, and hence, we omit them. Nevertheless, we can still sketch what the intersection of the curves $\frac{(u_l-u_r)^2}{2gH}x$ and $(1-x^2)(1-x)$ look like in $\mathbb{R}^2$, as we do in Figure 5. Since our only free parameter in controlling solutions to (26) is the slope, $\frac{(u_l-u_r)^2}{2gH} > 0$, of the purple line, it is clear from Figure 5 that (26) will always have three real roots, one of which is negative, and two of which are positive. However, if x is negative, then either one of h_m or H is negative, which is non-physical, since our fluid cannot drop below the bottom of the ground. (But would it not be cool if colliding water waves dug out the ground underneath them?) Hence, we know that the negative root for x can be excluded. Furthermore, out of the positive roots for x, one is less than one, while the other is greater than one, which correspond to $h_m < H$ and $h_m > H$, respectively. But the Lax entropy condition is just $h_m > h_* = H$, such that the root corresponding to $x < 1$ is an entropy-violating solution. Thus, we conclude that the rightmost root shown in Figure 5 is the physically-correct root for (26).

Now, we could stop here and conclude that the speeds for the left shock and the right shock are $s_- = u_l - \sqrt{gh_m\frac{H+h_m}{2H}}$ and $s_+ = u_r + \sqrt{gh_m\frac{H+h_m}{2H}}$, respectively, where $h_m = Hx$ and x is represented by a rather ugly, obscure expression. However, to make our problem more tractable and comprehensible, suppose that $(u_l - u_r)^2 \ll 2gH$, i.e., that our shock waves initially collide with each other at reasonable speeds and/or that our waves are adequately tall.[6] Then, we can let $\varepsilon^2 = \frac{(u_l-u_r)^2}{2gH} \ll 1$[7] and expand x in a perturbation series about ε: $x = x_0 + \varepsilon x_1 + \varepsilon^2 x_2 + \varepsilon^3 x_3 + \mathcal{O}\left(\varepsilon^4\right)$.[8] Plugging this series into (26),

$$\begin{gathered}
\varepsilon^2 x = \left(1-x^2\right)(1-x) \Rightarrow \\
\varepsilon^2 x_0 + \varepsilon^3 x_1 + \mathcal{O}\left(\varepsilon^4\right) = \\
\left[1 - x_0^2 - 2\varepsilon x_0 x_1 - \varepsilon^2\left(2x_0x_2 + x_1^2\right) - \varepsilon^3\left(2x_1x_2 + 2x_0x_3\right)\right]\left(1 - x_0 - \varepsilon x_1 - \varepsilon^2 x_2 - \varepsilon^3 x_3\right) = \\
1 - x_0 - x_0^2 + x_0^3 - x_1\varepsilon - 2x_0x_1\varepsilon + 3x_0^2x_1\varepsilon - x_1^2\varepsilon^2 + 3x_0x_1^2\varepsilon^2 - x_2\varepsilon^2 - 2x_0x_2\varepsilon^2 + 3x_0^2x_2\varepsilon^2 \\
+x_1^3\varepsilon^3 - 2x_1x_2\varepsilon^3 + 6x_0x_1x_2\varepsilon^3 - x_3\varepsilon^3 - 2x_0x_3\varepsilon^3 + 3x_0^2x_3\varepsilon^3 + \mathcal{O}\left(\varepsilon^4\right)
\end{gathered}$$

$$\Rightarrow \mathcal{O}(1): \quad x_0^3 - x_0^2 - x_0 + 1 = \left(1-x_0^2\right)(1-x_0) = 0 \tag{27}$$

$$\mathcal{O}(\varepsilon): \quad 3x_0^2x_1 - 2x_0x_1 - x_1 = 0 \tag{28}$$

$$\mathcal{O}\left(\varepsilon^2\right): \quad 3x_0^2x_2 + 3x_0x_1^2 - 2x_0x_2 - x_1^2 - x_2 = x_0, \tag{29}$$

$$\mathcal{O}\left(\varepsilon^3\right): \quad x_1^3 - 2x_1x_2 + 6x_0x_1x_2 - x_3 - 2x_0x_3 + 3x_0^2x_3 = x_1. \tag{30}$$

(27)-(29) can be solved in sequence. At first, the roots for (27) can be immediately read off as $x_0 = \pm 1$, and since $x_0 > 0$, we must have $x_0 = 1$. Then, (28) gives us that $3x_1 - 2x_1 - x_1 = 0$, which means that x_1 is currently undetermined. Next, (29) gives us that $3x_2 + 3x_1^2 - 2x_2 - x_1^2 - x_2 = 2x_1^2 = 1 \Rightarrow x_1 = \pm 1/\sqrt{2}$. But then, note that choosing $x_1 = -1/\sqrt{2}$ gives us the root corresponding to $x < 1$, an entropy-violating solution, whence we choose $x_1 = 1/\sqrt{2}$. Finally, (30) implies that

$$\begin{gathered}
\left(\frac{1}{\sqrt{2}}\right)^3 - 2\left(\frac{1}{\sqrt{2}}\right)x_2 + 6\left(\frac{1}{\sqrt{2}}\right)x_2 - x_3 - 2x_3 + 3x_3 = \frac{1}{\sqrt{2}} \\
\Rightarrow \left(\frac{1}{\sqrt{2}}\right)^2 + 4x_2 = 1 \Rightarrow x_2 = \frac{1}{8}.
\end{gathered}$$

Thus, putting everything together,

$$x = 1 + \frac{1}{\sqrt{2}}\varepsilon + \frac{1}{8}\varepsilon^2 + \mathcal{O}\left(\varepsilon^3\right) \Rightarrow$$

$$\boxed{h_m = H\left[1 + \frac{|u_l - u_r|}{2\sqrt{gH}} + \frac{(u_l-u_r)^2}{16gH} + \mathcal{O}\left(\frac{|u_l-u_r|^3}{g^{3/2}H^{3/2}}\right)\right].} \tag{31}$$

(31) gives us shock speeds

$$\boxed{s_- = u_l - \sqrt{gh_m\frac{H+h_m}{2H}},} \tag{32}$$

$$\boxed{s_+ = u_r + \sqrt{gh_m\frac{H+h_m}{2H}},} \tag{33}$$

[6]If we test any normal physical parameters (i.e., nothing utterly extreme), then we will find that $(u_l-u_r)^2 \ll 2gH$ is pretty easily satisfied. For instance, even if $u_l - u_r = 10$ m/s, $g = 9.81$ m/s^2, and $H = 100$ m, then $(u_l-u_r)^2 = 100$ m^2/s^2 and $2gH = 1962$ m^2/s^2, such that $\frac{(u_l-u_r)^2}{2gH} \approx 0.051 \ll 1$.

[7]From Figure 5, we know that both positive roots vanish when $\frac{(u_l-u_r)^2}{2gH} < 0$. Hence, if we naively just let $\varepsilon = \frac{(u_l-u_r)^2}{2gH}$ and try to find a series expression in terms of ε for one of these positive roots, then we will reach a contradiction. (I tried it.)

[8]We will see that such an expansion actually only allows us to go up to second-order.

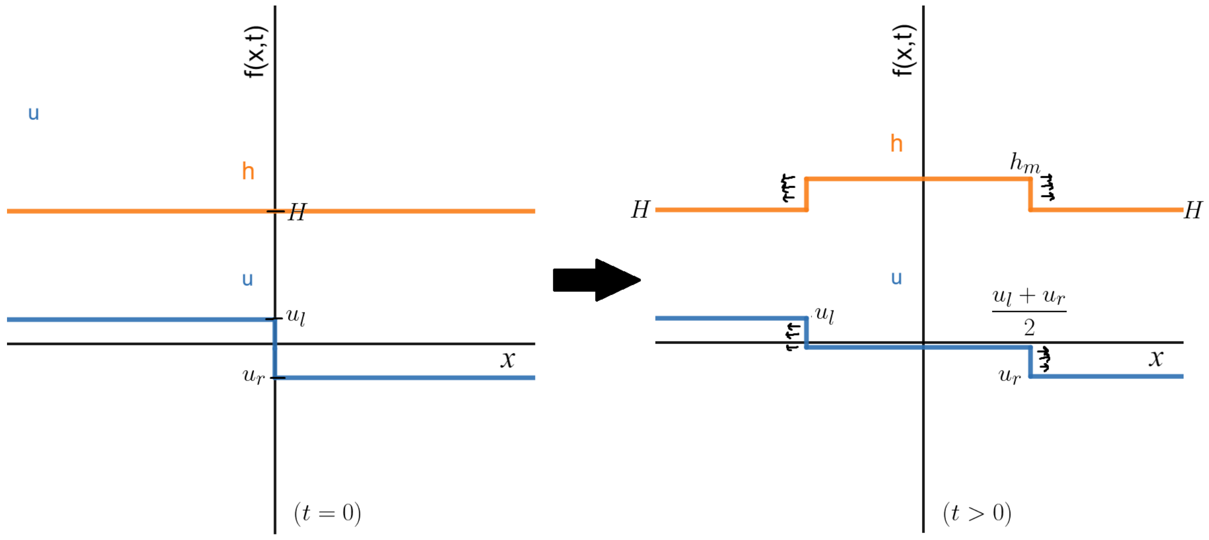

Figure 6: A sketch (partially made in Desmos) showing the solution to the wave-crash problem, clearly displaying the two shock waves moving away from the line $x = 0$ at different speeds. In particular, we used our series solutions (up to second-order terms) with $H = 10$, $u_l = 1.8$, and $u_r = -2.6$ to help complete the sketch. Note in particular that the left shock is traveling faster than the right shock, which makes sense from conservation of energy and momentum; if two objects of equal masses collide, the one which was traveling at a slower speed before the collision will bounce away with a quicker speed, and vice-versa.

where we can substitute (31) into (32) and (33). Therefore, by (24) and (31), our solution to the Riemann problem (21) is

$$\boxed{h(x,t) = \begin{cases} H & x < s_-t, \\ H\left[1 + \frac{|u_l-u_r|}{2\sqrt{gH}} + \frac{(u_l-u_r)^2}{16gH} + \mathcal{O}\left(\frac{|u_l-u_r|^3}{g^{3/2}H^{3/2}}\right)\right] & s_-t < x < s_+t, \\ H & x > s_+t, \end{cases}}$$

$$\boxed{u(x,t) = \begin{cases} u_l & x < s_-t, \\ \frac{1}{2}(u_l+u_r) & s_-t < x < s_+t, \\ u_r & x > s_+t, \end{cases}}$$

where s_- and s_+ are defined in (32) and (33), respectively.

A sketch of the solution to the wave-crash problem appears in Figure 6.

3 The shallow-water equations in 2D

For the two-dimensional problem, we will now let there be a non-negligible velocity in the y-direction ($v \neq 0$) and consider the effects of rotation, i.e., $\mathbf{f} \neq 0$. Then, we have to deal with (3) in its entirety, which we write out again here for convenience:

$$\begin{cases} \partial_t h + \partial_x U + \partial_y V = 0, \\ \partial_t U + \partial_x\left(\frac{U^2}{h} + \frac{1}{2}gh^2\right) + \partial_y\left(\frac{UV}{h}\right) = fV = \left[\sin\theta + \frac{1}{a}(x\sin\phi + y\cos\phi)\cos\theta\right]2\Omega V, \\ \partial_t V + \partial_x\left(\frac{UV}{h}\right) + \partial_y\left(\frac{V^2}{h} + \frac{1}{2}gh^2\right) = -fU = -\left[\sin\theta + \frac{1}{a}(x\sin\phi + y\cos\phi)\cos\theta\right]2\Omega U. \end{cases}$$

3.1 Problem setup

We now need to consider the domain in the y-direction as well. To simulate something analogous to the 1D Riemann problems we explored above, suppose that our domain is a long, rectangular channel of length L, width D, and height H,[9] where the line $y = 0$ splits the channel down the middle. Hence, our x-coordinate goes *along* the channel, while our y-coordinate goes *across* the channel.

Although, this also means that we need to impose boundary conditions for our flow and height at the edges of the channel. At first, we take the boundaries to be solid, such that we have the no-penetration boundary condition, $v\left(x, \pm D/2, t\right) = 0$, which basically states that there is no fluid flux through the walls of the channel. We also impose an inviscid wall boundary condition: $\widehat{\mathbf{n}} \cdot \nabla u\left(x, \pm D/2, t\right) = \partial_y u\left(x, \pm D/2, t\right) = 0$. [Yod16] This basically enforces that at the wall, the fluid flows tangentially to the surface. In viscous fluids, we actually would impose the no-slip condition, $u\left(x, \pm D/2, t\right) = 0$, instead, and even low-viscosity fluids in real life have a viscous layer near to the wall. [Day90; SG17] However, we can assume that D is large enough such that these effects are negligible and our domain of interest is large enough that we do not really care what happens to the flow close to the wall. Finally, we have $\widehat{\mathbf{n}} \cdot \nabla h\left(x, \pm D/2, t\right) = \partial_y h\left(x, \pm D/2, t\right) = 0$, which basically states that the net flux of mass through the wall is zero, since $h \sim M$. The rest of our boundary conditions are the same: we want to solve the problem for all $x \in \mathbb{R}$.

3.2 Numerical method

Ideally, we could solve the problem for all $x \in \mathbb{R}$. However, unless we use some numerical Green's function approach, we need to solve this problem on a finite domain, which means that we need to impose boundary conditions once we truncate our domain in x. The least restrictive boundary conditions are $\partial_y h\left(\pm L/2, y, t\right) = \partial_y u\left(\pm L/2, y, t\right) = \partial_y v\left(\pm L/2, y, t\right) = 0$. These are basically "open" boundary conditions, which, up to first-order, are equivalent with saying that we approximate values immediately outside the domain in x with their neighboring values in the domain. Hence, we heuristically can say that such boundary conditions effectively approximate the case where our waves continue on through the boundary without restrictions. This is akin to free-end boundary conditions for the wave equation.

To solve the PDE system (3) with the boundary conditions given above, we need to use a robust numerical solver that can capture shocks and other discontinuities which could (and will) arise in our numerical solutions. Sometimes, when scientists and engineers want to solve conservation laws of the form (8) numerically, they just discretize the PDE using finite differences and naively throw a finite-difference method at it. If discontinuities appear, then they can either smooth these out by adding a viscosity term of the form $\varepsilon \nabla^2 \mathbf{q}$ to the right side for sufficiently small $\varepsilon > 0$ (effectively, this adds some diffusion to the problem) or choose not to smooth these out and deal with any stiffness, Gibbs phenomenon, etc. by using an extremely fine grid resolution in both space and time. [LeV02; FM01; GPP11]

However, by writing (8) out in integral form as

$$\int_{\Omega} \partial_t \mathbf{q} dV = -\int_{\Omega} \nabla \cdot \left(F\left(\mathbf{q}\right)\right) dV = -\int_{d\Omega} F\left(\mathbf{q}\right) \cdot \widehat{\mathbf{n}} dS \tag{34}$$

[9]This can be used as rather angular approximation to, say, an arbitrary river, channel, stream, or other body of water.

via the divergence theorem for some arbitrary control volume Ω in our domain, we can approximate the integrals on both sides (i.e., the fluxes) and thereby derive a class of numerical schemes called finite volume methods. [Tor99; LeV02] Depending on how one approximates these integrals (some methods actually just reproduce the same methods one could derive via finite differences), one can not only capture discontinuous or non-differentiable solutions that exhibit shocks and rarefactions, but even filter out entropy-violating solutions.

One could write an entire paper which goes into detail regarding the theory behind monotonic schemes, TVD (total variation diminishing) schemes, Godunov's theorem, flux limiters, etc., as well as the scheme which we will use to solve (3), but since this is not a numerical analysis course, we will just mention the numerical scheme we used, some more specifics about its implementation, as well as some of its properties. To solve (3), we used a **M**onotonic **U**pstream-centered **S**cheme for **C**onservation **L**aws (or a MUSCL scheme for short) called the Kurganov and Tadmor (KT) central scheme with the frequently-used superbee flux limiter. [Lee79; KT00] The KT central scheme does not require a Riemann solver (i.e., one that requires information about solutions to the Riemann problem for the given conservation law), is high-resolution (i.e., free of dissipation and smearing around shocks and discontinuities) and second-order, and uses MUSCL reconstruction.[10] We apply this scheme to approximate (34) at each timestep and across some volume-based discretization of the spatial domain, and to evolve the scheme forward in time, we apply the explicit midpoint method (a.k.a, RK2). [BFB16]

However, the Coriolis terms fall outside the scope of application for finite volume methods, which means that we cannot just apply the KT central scheme to solve (3) completely. This is because finite-volume methods only apply to the contributions from the flux terms that come from the conservation law, i.e., those which are on the left-hand side (3). Nevertheless, we can still express (3) in the form $\mathbf{q}_t = (\mathcal{A} + \mathcal{B})\,\mathbf{q}$ for some (in general, nonlinear) differential operators $\mathcal{A}$ and $\mathcal{B}$. In our case, $\mathcal{A}$ represents $\nabla \cdot \mathbf{F}$ while $\mathcal{B}$ is just a linear operator which represents the Coriolis terms. We want to apply what is called a fractional-step method. [LeV02] These methods integrate $\mathbf{q}$ in time by taking partial timesteps in integrating $\mathbf{q}_t = \mathcal{A}\mathbf{q}$ and $\mathbf{q}_t = \mathcal{B}\mathbf{q}$ separately in some combination, and then use these results to approximate integrating the full system. Since the explicit midpoint method is second-order in time, we want to apply a fractional-step method which is also second-order in time. The method we apply is called Strang splitting, which goes as follows for each timestep of size Δt:

1. Integrate $\widetilde{\mathbf{q}}_t = \mathcal{A}\widetilde{\mathbf{q}}$ over a half timestep of length $\Delta t/2$.
2. Using $\widetilde{\mathbf{q}}$ as an initial condition, integrate $\widehat{\mathbf{q}}_t = \mathcal{A}\widehat{\mathbf{q}}$ over a full timestep of length Δt.
3. Finally, using $\widehat{\mathbf{q}}$ as an initial condition, integrate $\mathbf{q}_t = \mathcal{A}\mathbf{q}$ over another half timestep of length $\Delta t/2$.

To integrate each PDE, ODE, etc. above, one can use exact or approximate methods, but since Strang splitting introduces a $\mathcal{O}\left((\Delta t)^2\right)$ error, one might observe a reduction of order of convergence if one uses a numerical integrator of order greater than two. Furthermore, in theory, one could apply different numerical integrators for each step above, but for simplicity's sake, we apply RK2 to integrate at each step for Strang splitting.

[10]MUSCL schemes use a flux limiter, e.g., van Leer, superbee, van Albada, ospre, etc., to limit the slope when piecewise-linearly extrapolating the function $\mathbf{q}$ within each volume cell in the spatial domain. [Lee74; vvR82; Roe86; LeV02; WD07] This prevents spurious oscillations near shocks and discontinuities.

To implement the above numerical scheme in solving (3), I wrote the following script in MATLAB. Note that one should change parameters and variables (e.g., initial conditions, choice of flux limiter, step sizes, etc.) as necessary within the script directly, as opposed to inputting these into a function. However, there are multiple helper functions defined throughout the script.

```
close all

movienumber = 0; % change for movie name

% physical parameters
T = 8 * 60 * 60; % [s]
theta = 60; % latitude, in degrees
phi = 45; % local angle about vertical in degrees
% x goes along the channel, y goes across the channel
a = 6360e3; % [m]
g = 9.80665; % [m/s^2]
Omega = 7.2921e-5; % [rad/s]
H = 50; % [m]
D = 200e3; % [m]
L = 1000e3; % [m]
s = 5; % scaling parameter (>2)

% numerical parameters
dt = 2;
dx = L * 5e-3;
dy = D * 5e-3;
fluxlimiter = 'superbee'; % set flux limiter
% options for flux limiter include minmod, superbee, ospre,
%  vanleer, vanalbada

% simulation parameters
M = round(T / dt);
Nx = 2 * round(L / (2 * dx)) + 1;
Ny = 2 * round(D / (2 * dy)) + 1;
t = 0;

% initial conditions
soli = zeros(Nx, Ny, 3);
% % dam-breaking
% soli(1 : (Nx - 1) / 2, :, 1) = H + (H / s);
% soli((Nx + 1) / 2 : end, :, 1) = H - (H / s);
% water slam
soli(:, :, 1) = H;
soli(1 : (Nx - 1) / 2, :, 2) = H;
soli((Nx + 1) / 2 : end, :, 2) = -H;
% % hump of water
% xq = 1 : Nx;
% soli(:, :, 1) = (H + (H / s) * (exp(-(400 *...
% ((xq / Nx) - 1 / 2).^2)) - 1)).' * ones(1, Ny);

% plotting parameters
```

```
plotno = Inf;
plotevery = 20;
plotno = min(min(Nx, plotno), Ny);
xvec = 1 : round(Nx / plotno) : Nx;
yvec = 1 : round(Ny / plotno) : Ny;
[X, Y] = meshgrid(dx * xvec - L / 2, dy * yvec - D / 2);

f0 = 2 * Omega * sin(theta * pi / 180);
beta = 2 * Omega * cos(theta * pi / 180) / a;
xq = 1 : Nx;
yq = 1 : Ny;
[Xq, Yq] = meshgrid(dx * xq - L / 2, dy * yq - D / 2);
Xq = Xq.';
Yq = Yq.';
f = f0 + beta * (Xq * sin(phi * pi / 180) + Yq * cos(phi * pi / 180));

moviename = ['MUSCL_', num2str(movienumber), '.mp4'];

fh = figure(1);
fh.WindowState = 'maximized';
vidfile = VideoWriter(moviename, 'MPEG-4');
vidfile.FrameRate = 5;
open(vidfile);
for k = 1 : M
    solf = func(soli, f, dt, dx, dy, Nx, Ny, g, fluxlimiter);
    t = t + dt;

    % plotting
    if round(k / plotevery) == (k / plotevery)
        % height
        tiledlayout(2, 3)
        ax1 = nexttile;
        hold on
            h = surf(ax1, X / 1e3, Y / 1e3, solf(xvec, yvec, 1).',...
                'FaceColor', 'interp', 'FaceAlpha',0.8);
            ground = surf(ax1, X / 1e3, Y / 1e3,...
                zeros(length(yvec), length(xvec)),...
                'FaceColor', [0.8 0.8 0.2]);
        hold off
        set(h, 'LineStyle', 'none');
        set(ground, 'LineStyle', 'none');
        title(ax1, 'Water height', 'FontWeight', 'normal')
        xlabel('x [km]')
        ylabel('y [km]')
        zlabel('h [m]')
        zlim([0 - (H / s) H + 2 * (H / s)])
        grid on
        colorbar('southoutside')
        caxis([H / 2 3 * H / 2])
        view(ax1, -phi, 30)

        % x-velocity
```

```
ax2 = nexttile;
h = surf(ax2, X / 1e3, Y / 1e3,...
    (solf(xvec, yvec, 2) ./ solf(xvec, yvec, 1)).',...
    'FaceColor', 'interp','FaceAlpha',0.8);
set(h, 'LineStyle', 'none');
[title1, title2] = title(ax2,...
    ['Minutes elapsed: ' num2str(t / 60)],...
    'Velocity along the channel');
title1.FontSize = 16;
xlabel('x [km]')
ylabel('y [km]')
zlabel('u (x-velocity) [m/s]')
zlim([-1 1] * 40 / (s - 1))
grid on
colorbar('southoutside')
caxis([-1 1] * 40 / (s - 1))
view(ax2, -phi, 30)

% y-velocity
ax3 = nexttile;
h = surf(ax3, X / 1e3, Y / 1e3,...
    (solf(xvec, yvec, 3) ./ solf(xvec, yvec, 1)).',...
    'FaceColor', 'interp','FaceAlpha',0.8);
set(h, 'LineStyle', 'none');
title(ax3, 'Velocity across the channel',...
    'FontWeight', 'normal')
xlabel('x [km]')
ylabel('y [km]')
zlabel('v (y-velocity) [m/s]')
zlim([-1 1] * 15 / (s - 1))
grid on
colorbar('southoutside')
caxis([-1 1] * 15 / (s - 1))
view(ax3, -phi, 30)

ax4 = nexttile;
hold on
    h = surf(ax4, X / 1e3, Y / 1e3, solf(xvec, yvec, 1).',...
        'FaceColor', 'interp','FaceAlpha',0.8);
    ground = surf(ax4, X / 1e3, Y / 1e3,...
        zeros(length(yvec), length(xvec)),...
        'FaceColor', [0.8 0.8 0.2]);
hold off
set(h, 'LineStyle', 'none');
set(ground, 'LineStyle', 'none');
xlabel('x [km]')
ylabel('y [km]')
zlabel('h (water height) [m]')
zlim([0 H + 2 * (H / s)])
grid on
caxis([H / 2 3 * H / 2])
view(ax4, 90 - phi, 30)
```

```
        ax5 = nexttile;
        h = surf(ax5, X / 1e3, Y / 1e3,...
            (solf(xvec, yvec, 2) ./ solf(xvec, yvec, 1)).',...
            'FaceColor', 'interp','FaceAlpha',0.8);
        set(h, 'LineStyle', 'none');
        xlabel('x [km]')
        ylabel('y [km]')
        zlabel('u (x-velocity) [m/s]')
        zlim([-1 1] * 40 / (s - 1))
        grid on
        caxis([-1 1] * 40 / (s - 1))
        view(ax5, 90 - phi, 30)

        ax6 = nexttile;
        h = surf(ax6, X / 1e3, Y / 1e3,...
            (solf(xvec, yvec, 3) ./ solf(xvec, yvec, 1)).',...
            'FaceColor', 'interp','FaceAlpha',0.8);
        set(h, 'LineStyle', 'none');
        xlabel('x [km]')
        ylabel('y [km]')
        zlabel('v (y-velocity) [m/s]')
        zlim([-1 1] * 15 / (s - 1))
        grid on
        caxis([-1 1] * 15 / (s - 1))
        view(ax6, 90 - phi, 30)

        colormap(ax1, cold)
        colormap(ax2, jet)
        colormap(ax3, jet)
        colormap(ax4, cold)
        colormap(ax5, jet)
        colormap(ax6, jet)

        drawnow
        F = getframe(fh);
        writeVideo(vidfile, F);
    end
    soli = solf;
end
close(vidfile)

function F = xflux(sol, Nx, Ny, grav)
    F = zeros(Nx, Ny, 3);
    ETA = sol(:, :, 1);
    U = sol(:, :, 2) ./ ETA;
    V = sol(:, :, 3) ./ ETA;
    F(:, :, 1) = U;
    F(:, :, 2) = U .* U + 0.5 * grav * ETA;
    F(:, :, 3) = U .* V;
    F = ETA .* F;
end
```

```
function G = yflux(sol, Nx, Ny, grav)
    G = zeros(Nx, Ny, 3);
    ETA = sol(:, :, 1);
    U = sol(:, :, 2) ./ ETA;
    V = sol(:, :, 3) ./ ETA;
    G(:, :, 1) = V;
    G(:, :, 2) = U .* V;
    G(:, :, 3) = V .* V + 0.5 * grav * ETA;
    G = ETA .* G;
end

function Fspeed = aF(solL, solR, g)
    ETAL = solL(:, :, 1);
    UL = solL(:, :, 2) ./ ETAL;
    ETAR = solR(:, :, 1);
    UR = solR(:, :, 2) ./ ETAR;
    FcompareL = repmat(UL, [1 1 3]);
    FcompareL(:, :, 2) = FcompareL(:, :, 2) - sqrt(g * ETAL);
    FcompareL(:, :, 3) = FcompareL(:, :, 3) + sqrt(g * ETAL);
    FcompareR = repmat(UR, [1 1 3]);
    FcompareR(:, :, 2) = FcompareR(:, :, 2) - sqrt(g * ETAR);
    FcompareR(:, :, 3) = FcompareR(:, :, 3) + sqrt(g * ETAR);
    largestL = max(abs(FcompareL), [], 3);
    largestR = max(abs(FcompareR), [], 3);
    Fspeed = max(largestL, largestR);
end

function Gspeed = aG(solL, solR, g)
    ETAL = solL(:, :, 1);
    VL = solL(:, :, 3) ./ ETAL;
    ETAR = solR(:, :, 1);
    VR = solL(:, :, 3) ./ ETAR;
    GcompareL = repmat(VL, [1 1 3]);
    GcompareL(:, :, 2) = GcompareL(:, :, 2) - sqrt(g * ETAL);
    GcompareL(:, :, 3) = GcompareL(:, :, 3) + sqrt(g * ETAL);
    GcompareR = repmat(VR, [1 1 3]);
    GcompareR(:, :, 2) = GcompareR(:, :, 2) - sqrt(g * ETAR);
    GcompareR(:, :, 3) = GcompareR(:, :, 3) + sqrt(g * ETAR);
    largestL = max(abs(GcompareL), [], 3);
    largestR = max(abs(GcompareR), [], 3);
    Gspeed = max(largestL, largestR);
end

function phi = fluxlim(vec, fluxlimiter)
    if strcmp(fluxlimiter, 'minmod')
        phi = max(0, min(1, vec)); % minmod
    elseif strcmp(fluxlimiter, 'ospre')
        phi = 1.5 * ((vec .* vec) + vec) ./...
            ((vec .* vec) + vec + 1); % ospre
        phi(isnan(phi)) = 1.5;
    elseif strcmp(fluxlimiter, 'superbee')
```

```
        phi = zeros(length(vec(:, 1, 1)),...
            length(vec(1, :, 1)), 3, 3); % superbee
        phi(:, :, :, 2) = min(2 * vec, 1);
        phi(:, :, :, 3) = min(vec, 2);
        phi = max(phi, [], 4);
    elseif strcmp(fluxlimiter, 'vanleer')
        phi = (vec + abs(vec)) ./ (1 + abs(vec)); % vanleer
        phi(isnan(phi)) = 2;
    elseif strcmp(fluxlimiter, 'vanalbada')
        phi = ((vec .* vec) + vec) ./...
            ((vec .* vec ) + 1); % vanalbada(1)
        phi(isnan(phi)) = 1;
    else
        phi = zeros(length(vec(:, 1, 1)),...
            length(vec(1, :, 1)), 3, 3); % superbee (default)
        phi(:, :, :, 2) = min(2 * vec, 1);
        phi(:, :, :, 3) = min(vec, 2);
        phi = max(phi, [], 4);
    end
end

function dF = dFfluxdiff(soli, Nx, Ny, g, fluxlimiter)
    % transform into u and v for BCs
    soli(:, :, 2 : 3) = soli(:, :, 2 : 3) ./ soli(:, :, 1);

    % implement Neumann (no-penetration) BCs
    % x = -D / 2
    solba = circshift(soli, 1, 1);
    solba(1, :, :) = (4 * soli(1, :, :) - soli(2, :, :)) / 3;

    % x = -D / 2 -
    solbaba = circshift(solba, 1, 1);
    solbaba(1, :, :) = soli(1, :, :);

    % x = D / 2
    solfr = circshift(soli, -1, 1);
    solfr(end, :, :) = (4 * soli(end, :, :)...
        - soli(end - 1, :, :)) / 3;

    % x = D / 2 +
    solfrfr = circshift(solfr, -1, 1);
    solfrfr(end, :, :) = soli(end, :, :);

    % transform back into uh and vh
    soli(:, :, 2 : 3) = soli(:, :, 2 : 3) .* soli(:, :, 1);
    solba(:, :, 2 : 3) = solba(:, :, 2 : 3) .* solba(:, :, 1);
    solbaba(:, :, 2 : 3) = solbaba(:, :, 2 : 3) .* solbaba(:, :, 1);
    solfr(:, :, 2 : 3) = solfr(:, :, 2 : 3) .* solfr(:, :, 1);
    solfrfr(:, :, 2 : 3) = solfrfr(:, :, 2 : 3) .* solfrfr(:, :, 1);

    soliba = soli - solba;
    solfri = solfr - soli;
    solfrfrfr = solfrfr - solfr;
    oneflux = fluxlim(soliba ./ solfri, fluxlimiter);
    solLplus = soli + 0.5 * oneflux .* solfri;
    solLminus = solba + 0.5 * fluxlim((solba - solbaba) ./...
        soliba, fluxlimiter) .* soliba;
    solRplus = solfr - 0.5 * fluxlim(solfri ./...
        solfrfrfr, fluxlimiter) .* solfrfrfr;
    solRminus = soli - 0.5 * oneflux .* solfri;

    Fminus = 0.5 * (xflux(solRminus, Nx, Ny, g) +...
        xflux(solLminus, Nx, Ny, g) - aF(solLminus, solRminus, g)...
        .* (solRminus - solLminus));
    Fplus = 0.5 * (xflux(solRplus, Nx, Ny, g) +...
        xflux(solLplus, Nx, Ny, g) - aF(solLplus, solRplus, g)...
        .* (solRplus - solLplus));
    dF = Fplus - Fminus;
end

function dG = dGfluxdiff(soli, Nx, Ny, g, fluxlimiter)
    % transform into u and v for BCs
    soli(:, :, 2 : 3) = soli(:, :, 2 : 3) ./ soli(:, :, 1);

    % implement Neumann (no-penetration) BCs, aside for v at the walls,
    % which will have kinematic (homogeneous Dirichlet) BCs, i.e., v=0
    % y = 0
    solbot = circshift(soli, 1, 2);
    solbot(:, 1, 1 : 2) = (4 * soli(:, 1, 1 : 2)...
        - soli(:, 2, 1 : 2)) / 3;
    solbot(:, 1, 3) = 0;

    % y = 0 -
    solbotbot = circshift(solbot, 1, 2);
    solbotbot(:, 1, 1 : 2) = soli(:, 1, 1 : 2);
    solbotbot(:, 1, 3) = 0;

    % y = L
    soltop = circshift(soli, -1, 2);
    soltop(:, end, 1 : 2) = ...
        (4 * soli(:, end, 1 : 2) - soli(:, end - 1, 1 : 2)) / 3;
    soltop(:, end, 3) = 0;

    % y = L +
    soltoptop = circshift(soltop, -1, 2);
    soltoptop(:, end, 1 : 2) = soli(:, end, 1 : 2);
    soltoptop(:, end, 3) = 0;

    % transform back into uh and vh
    soli(:, :, 2 : 3) = soli(:, :, 2 : 3) .* soli(:, :, 1);
    solbot(:, :, 2 : 3) = solbot(:, :, 2 : 3) .* solbot(:, :, 1);
    solbotbot(:, :, 2 : 3) = solbotbot(:, :, 2 : 3)...
        .* solbotbot(:, :, 1);
    soltop(:, :, 2 : 3) = soltop(:, :, 2 : 3) .* soltop(:, :, 1);
    soltoptop(:, :, 2 : 3) = soltoptop(:, :, 2 : 3)...
        .* soltoptop(:, :, 1);

    solibot = soli - solbot;
    soltopi = soltop - soli;
    soltoptoptop = soltoptop - soltop;
    oneflux = fluxlim(solibot ./...
        (soltop - soli), fluxlimiter);
    solLplus = soli + 0.5 * oneflux .* soltopi;
    solLminus = solbot + 0.5 * fluxlim((solbot - solbotbot) ./...
        solibot, fluxlimiter) .* solibot;
    solRplus = soltop - 0.5 * fluxlim(soltopi ./...
        soltoptoptop, fluxlimiter) .* soltoptoptop;
    solRminus = soli - 0.5 * oneflux .* soltopi;

    Gminus = 0.5 * (yflux(solRminus, Nx, Ny, g) +...
        yflux(solLminus, Nx, Ny, g) - aG(solLminus, solRminus, g)...
        .* (solRminus - solLminus));
    Gplus = 0.5 * (yflux(solRplus, Nx, Ny, g) +...
        yflux(solLplus, Nx, Ny, g) - aG(solLplus, solRplus, g)...
        .* (solRplus - solLplus));
    dG = Gplus - Gminus;
end

function vec = cor(sol, f, Nx, Ny)
    vec = zeros(Nx, Ny, 2);
    vec(:, :, 1) = f .* sol(:, :, 3);
    vec(:, :, 2) = -f .* sol(:, :, 2);
end

function solf = func(sol, f, dt, dx, dy, Nx, Ny, g, fluxlimiter)
% Strang splitting with midpoint method for all timesteps
    dt = dt / 2;
    k1 = -((dFfluxdiff(sol, Nx, Ny, g, fluxlimiter) / dx) + ...
        (dGfluxdiff(sol, Nx, Ny, g, fluxlimiter) / dy));
    k2 = -((dFfluxdiff(sol + dt * k1 / 2,...
        Nx, Ny, g, fluxlimiter) / dx) + ...
        (dGfluxdiff(sol + dt * k1 / 2, Nx, Ny, g, fluxlimiter) / dy));
    solf = sol + dt * k2; % first time step

    dt = 2 * dt;
    % do not need to update h for Coriolis terms
    k1(:, :, 2 : 3) = cor(solf, f, Nx, Ny);
    k2 = cor(solf + dt * k1 / 2, f, Nx, Ny);
    solf(:, :, 2 : 3) = solf(:, :, 2 : 3) + dt * k2; % second time step

    dt = dt / 2;
    k1 = -((dFfluxdiff(solf, Nx, Ny, g, fluxlimiter) / dx) + ...
        (dGfluxdiff(solf, Nx, Ny, g, fluxlimiter) / dy));
    k2 = -((dFfluxdiff(solf + dt * k1 / 2,...
        Nx, Ny, g, fluxlimiter) / dx) + ...
        (dGfluxdiff(solf + dt * k1 / 2, Nx, Ny, g, fluxlimiter) / dy));
    solf = solf + dt * k2; % final time step
end
```

For all simulations linked below as movies, we used the parameters $\theta = 60°$,[11] $a = 6360 \times 10^3$ m, $g = 9.80665$ m/s^2, $\Omega = 7.2921 \times 10^{-5}$ rad/s, $H = 50$ m, $D = 200$ km, $L = 1000$ km, $\Delta t = 2$ s, $\Delta x = 5L \times 10^{-3}$, and $\Delta y = 5D \times 10^{-3}$.

3.3 Example 1(b): The dam-break problem

Analogously to the 1D case, we use initial conditions

$$h\left(x<0,y,0\right) = H + \frac{H}{s}, \quad h\left(x>0,y,0\right) = H - \frac{H}{s}, \quad u\left(x,y,0\right) = v\left(x,y,0\right) = 0, \tag{35}$$

where $s > 1$ is a scaling parameter which determines the height of the "dam" which breaks at $t = 0$. s mainly changes the qualitative behavior of the solution; the higher the dam, the more kinetic energy that will be imparted into our shock wave, and hence, the faster and higher the shock will be. For sake of concreteness, we let $s = 5$ for all of our numerical solutions listed below:

- $\phi = 0°$. Click here for the video.
- $\phi = 45°$. Click here for the video.
- $\phi = 90°$. Click here for the video.

[12]

3.4 Example 2(b): The wave-crash problem

This time, we use initial conditions

$$h\left(x,y,0\right) = H, \quad u\left(x<0,y,0\right) = -u\left(x>0,y,0\right) = 1, \quad v\left(x,y,0\right) = 0, \tag{36}$$

to compute the following numerical solutions:

- $\phi = 0°$. Click here for the video.
- $\phi = 45°$. Click here for the video.
- $\phi = 90°$. Click here for the video.

In all of the examples above, we initially see qualitative behavior that is very similar as in the 1D case. However, the Coriolis terms slowly begin to dominate the solution over time, and not just in the y-direction. We find major qualitative changes in the "middle" region, where the profile rises greatly in the direction of the Coriolis force, especially for the dam-break problem. This is because fluid parcels (in a similar manner as with Kelvin waves) get pushed against some wall of the channel and get stuck in that vicinity, thereby

[11] θ mainly controls the strength and direction of the Coriolis force on our waves. Hence, for our qualitative understanding, it suffices to fix θ; using our physical understanding, we can imagine how our solution would look at other values of θ.

[12] Do not forget to adjust the video quality! For some reason, when these videos play from Google Drive, they are in poorer quality by default, and I have had to click "720p" manually for them to adjust.

causing the water to rise in elevation there. However, on the other side of the channel, we are left with less fluid than on the other side, which means that fluid parcels will experience a greater x-velocity there, since they have to travel over a larger height difference. This in turn gradually accelerates the part of the shock lying on that side of the channel, while the shock decelerates on the other side.

Hence, for waterways that are on the scale of the deformation radius or larger, we conclude that the effects of rotation have very non-negligible and non-trivial effects on the evolution of shocks and rarefactions in solutions for the 2D shallow-water equations. We cannot just extrapolate our 1D results across the entire channel; not only are the 1D profiles at fixed y heavily distorted by the influence of Coriolis forces, but the local speeds of fluid waves at different y-values are affected by the distribution of fluid throughout the channel. A future research question which could be of interest is to track exactly how the wavefront of a shock or rarefaction in 2D is affected and bent by Coriolis forces over time, as well as how the speeds vary across a distorting wavefront in this situation.

References

[BFB16] Richard L. Burden, J. Douglas Faires, and Annette M. Burden. *Numerical Analysis.* Tenth edition. Cengage Learning, 2016.

[CVZ03] Valerio Caleffi, Alessandro Valiani, and Andrea Zanni. "Finite volume method for simulating extreme flood events in natural channels". In: *Journal of Hydraulic Research* 41.2 (Mar. 2003), pp. 167–177.

[CST19] Octavia Crompton, Anneliese Sytsma, and Sally Thompson. "Emulation of the Saint Venant Equations Enables Rapid and Accurate Predictions of Infiltration and Overland Flow Velocity on Spatially Heterogeneous Surfaces". In: *Water Resources Research* 55.8 (Aug. 2019), pp. 7108–7129.

[Day90] Michael A. Day. "The no-slip condition of fluid dynamics". In: *Erkenntnis* 33.3 (Nov. 1990), pp. 285–296.

[ELT20] Mehmet Ersoy, Omar Lakkis, and Philip Townsend. "A Saint-Venant Model for Overland Flows with Precipitation and Recharge". In: *Mathematical and Computational Applications* 26.1 (Dec. 2020), p. 1.

[Est+00] M Esteves et al. "Overland flow and infiltration modelling for small plots during unsteady rain: numerical results versus observed values". In: *Journal of Hydrology* 228.3-4 (Mar. 2000), pp. 265–282.

[Eva10] Lawrence Evans. *Partial Differential Equations.* American Mathematical Society, Mar. 2010.

[FM01] Maurizio Falcone and Charalampos Makridakis. *Numerical Methods for Viscosity Solutions and Applications.* World Scientific, 2001.

[GPP11] Jean-Luc Guermond, Richard Pasquetti, and Bojan Popov. "Entropy viscosity method for nonlinear conservation laws". In: *Journal of Computational Physics* 230.11 (May 2011), pp. 4248–4267.

[KLR20] David I. Ketcheson, Randall J. LeVeque, and Mauricio J. del Razo. *Riemann Problems and Jupyter Solutions.* Society for Industrial and Applied Mathematics, Jan. 2020.

[KCY07] Dae-Hong Kim, Yong-Sik Cho, and Yong-Kon Yi. “Propagation and run-up of nearshore tsunamis with HLLC approximate Riemann solver”. In: *Ocean Engineering* 34.8-9 (June 2007), pp. 1164–1173.

[KT00] Alexander Kurganov and Eitan Tadmor. “New High-Resolution Central Schemes for Nonlinear Conservation Laws and Convection–Diffusion Equations”. In: *Journal of Computational Physics* 160.1 (May 2000), pp. 241–282.

[Lee74] Bram van Leer. “Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second-order scheme”. In: *Journal of Computational Physics* 14.4 (Mar. 1974), pp. 361–370.

[Lee79] Bram van Leer. “Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov’s method”. In: *Journal of Computational Physics* 32.1 (July 1979), pp. 101–136.

[LeV02] Randall J. LeVeque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge University Press, Aug. 2002.

[Roe86] P L Roe. “Characteristic-Based Schemes for the Euler Equations”. In: *Annual Review of Fluid Mechanics* 18.1 (Jan. 1986), pp. 337–365.

[Sai71] Saint-Venant. “Théorie du mouvement non permanent des eaux, avec application aux crues des rivières et à l’introduction des marées dans leur lit”. In: *C. R. Acad. Sci., Paris* 73 (1871), pp. 147–154.

[SG17] Hermann Schlichting and Klaus Gersten. *Boundary-Layer Theory.* Springer Berlin Heidelberg, 2017.

[Tem15] Blake Temple. “Math 22C Lecture Notes 9 - Shock Waves and the Riemann Problem”. Mar. 2015. URL: https://www.math.ucdavis.edu/~temple/MAT22C/LecturesMat22CW15/9-Shock_Waves_and_RiemProb-22C-W15.pdf.

[Tor99] Eleuterio F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics.* Springer Berlin Heidelberg, 1999.

[VCZ02] Alessandro Valiani, Valerio Caleffi, and Andrea Zanni. “Case Study: Malpasset Dam-Break Simulation using a Two-Dimensional Finite Volume Method”. In: *Journal of Hydraulic Engineering* 128.5 (2002), pp. 460–472.

[vvR82] G. D. van Albada, B. van Leer, and Jr. Roberts W. W. “A comparative study of computational methods in cosmic gas dynamics”. In: *Astronomy & Astrophysics* 108.1 (Apr. 1982), pp. 76–84.

[WD07] N.P. Waterson and H. Deconinck. “Design principles for bounded higher-order convection schemes – a unified approach”. In: *Journal of Computational Physics* 224.1 (May 2007), pp. 182–207.

[Yod16] Dennis Yoder. *Boundary Conditions.* 2016. URL: https://www.grc.nasa.gov/www/winddocs/user/bc.html.