# Generalizing Shallow Water Simulations with Dispersive Surface Waves

STEFAN JESCHKE, NVIDIA, USA
CHRIS WOJTAN, Institute of Science and Technology Austria (ISTA), Austria

Fig. 1. This composite image shows how our model can compute realistic boat wakes for a slow sailboat (left) and a fast motorboat (middle), dispersive reflections, and flooding behaviors (right). This scene covers 512m in each direction and is simulated and rendered in real time.

This paper introduces a novel method for simulating large bodies of water as a height field. At the start of each time step, we partition the waves into a *bulk flow* (which approximately satisfies the assumptions of the shallow water equations) and *surface waves* (which approximately satisfy the assumptions of Airy wave theory). We then solve the two wave regimes separately using appropriate state-of-the-art techniques, and re-combine the resulting wave velocities at the end of each step. This strategy leads to the first heightfield wave model capable of simulating complex interactions between both deep and shallow water effects, like the waves from a boat wake sloshing up onto a beach, or a dam break producing wave interference patterns and eddies. We also analyze the numerical dispersion created by our method and derive an *exact* correction factor for waves at a constant water depth, giving us a numerically perfect re-creation of theoretical water wave dispersion patterns.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; **Real-time simulation**.

Additional Key Words and Phrases: Water animation, real-time animation, natural phenomena

Authors' addresses: Stefan Jeschke, jeschke@stefan-jeschke.com, NVIDIA, 2788 San Tomas Expy, Santa Clara, CA, USA, 95051; Chris Wojtan, chris.wojtan@ist.ac.at, Institute of Science and Technology Austria (ISTA), Am Campus 1, Klosterneuburg, Austria, 3400.


0730-0301/2023/1-ART1
https://doi.org/10.1145/3592098

**ACM Reference Format:**
Stefan Jeschke and Chris Wojtan. 2023. Generalizing Shallow Water Simulations with Dispersive Surface Waves. *ACM Trans. Graph.* 1, 1, Article 1 (January 2023), 12 pages. https://doi.org/10.1145/3592098

## 1 INTRODUCTION

The motion of water is well-described by the incompressible Euler equations

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + \mathbf{g} \qquad (1)$$
$$\nabla \cdot \mathbf{u} = 0$$

where $\mathbf{u}$ is the water velocity, $D\mathbf{u}/Dt$ is the material derivative, $\rho$ is water density, $p$ is pressure, and $g$ is acceleration due to gravity. Numerically approximating this equation is prohibitively expensive for the animation of large bodies of water, so researchers reduce the complexity by assuming the water surface takes the form of a height field, where the water height and velocity are both just functions of 2D spatial coordinates, instead of 3D. The most common reductions of these equations used in computer graphics are Airy wave theory and shallow water approximations.

Airy wave theory [Airy 1841] assumes that the motion is a potential flow (velocity is the gradient of some potential, $\mathbf{u} = \nabla\phi$) and that the wave amplitude $a$ is small relative to the wavelength $\lambda$ ($a \ll \lambda$, or equivalently $ka \ll 1$ for wavenumber $k = 2\pi/\lambda$). These assumptions produce linearized water wave equations which are used extensively throughout the computer graphics literature [Canabal et al. 2016; Tessendorf 2004a]. This linear wave theory produces

waves with an angular frequency $\omega$ depending on wavenumber $k$ and water depth $h$

$$\omega(k,h) = \sqrt{gk\tanh(kh)}. \tag{2}$$

This particular dependence of $\omega$ on $k$ is called the *dispersion relationship*, and it tells us how the frequency varies with different wavelengths. It also effectively prescribes the surface wavespeed, which is equal to $\omega/k$. Accurately reproducing this dispersion relation is essential for many water-related phenomena like raindrop ripples and particular interference patterns in the wake behind boats, as illustrated in Figure 1. One drawback to this approach, however, is that its $ka \ll 1$ assumption prevents the wave heights from having a major effect on the fluid domain boundaries. For example, it cannot model water waves sloshing up a sloped beach, or spilling over a dam and filling a basin, because these scenarios require the water domain to change shape depending on the motion of the surface waves.

Another popular technique for simplifying water dynamics is to assume that the depth of the water $h$ is much smaller than the wavelength ($h \ll \lambda$, or equivalently $kh \ll 1$). This assumption gives rise to the *shallow water equations* (SWE), which *do* allow waves to slosh around and spill over terrain. However, this long wavelength / shallow depth assumption is only appropriate in exceptionally limited scenarios, because SWE produces a dispersion relationship

$$\omega(k,h) = k\sqrt{gh}, \tag{3}$$

which is drastically different from Equation 2 when $h \not\ll \lambda$. Notably, SWE is unable to produce the signature water wave interference patterns described above, and instead produces ripples similar to acoustic shock waves.

Although both of these methods for animating water are theoretically sound and offer robust and efficient implementations, they both have restrictive assumptions that lead to visually obvious breakdowns in common scenarios: Airy wave simulators have to avoid flooding scenarios and gently sloped solid obstacles; shallow water solvers either add procedural textures to approximate a more appropriate dispersion relation for deeper water [Chentanez and Müller 2010], or they fail to convincingly animate short wavelengths altogether.

Interestingly, both of these simulation techniques excel where the other fails. The purpose of this paper is to introduce a principled unification of the two fluid regimes together into a single model that lacks all of the failure modes described above. We do this by decomposing the water into two regimes of water motion: a *bulk flow* that is best described by the shallow water assumption $h \ll \lambda$, and the remaining *surface waves* that fail the shallow water test but still obey Airy wave theory. The bulk flow naturally includes most of the fluid's mass and momentum (it includes the biggest waves with the longest wavelengths), and so we simulate it using a shallow water solver capable of simulating flooding and convective eddies. Conversely, the surface waves contain all of the surface details and high frequencies that are essential for producing convincing water wave animations, so we simulate them with a highly detailed Airy wave solver. Re-computing the decomposition at each timestep allows water motion to continuously shift between shallow and deep regimes and furthermore it permits the fluid domain to significantly change over time.

In summary this paper offers the following contributions:

- The first height field method capable of simulating both Airy wave interference patterns and large bulk motions like flooding and 2D convective eddies in the same simulation;
- A practical algorithm for producing a spatially-varying decomposition of shallow and surface flows;
- A derivation of surface wave numerical dispersion errors, and a novel scheme for canceling them *exactly* in a constant depth;
- Exact volume conservation and real-time performance appropriate for games and other interactive scenarios.

The remainder of this paper is organized as follows: Section 2 discusses related work, and Section 3 provides additional background from fluid dynamics and introduces the theoretical framework behind our method. Section 4 describes our algorithm in detail, including the decomposition, the shallow water solver, the modified Airy solver, and implementation details. Section 5 evaluates our method and discusses results and future work, Section 6 concludes the paper.

## 2 RELATED WORK

### 2.1 Linearized water surface waves

Airy [1841] noted that the surface of an inviscid, irrotational, and incompressible fluid under the influence of gravity will exhibit waves according to the very specific dispersion relation in Equation 2. Lord Kelvin noted that the interference between these waves traveling at different speeds completely explains the visually recognizable patterns we see in wakes behind boats [Thomson 1887].

Computer graphics researchers devised numerous ways to mimic these dispersive water waves. Peachey [1986] and other researchers [Fournier and Reeves 1986; Hinsinger et al. 2002; Mastin et al. 1987] animate waves by directly computing the inverse Fourier transform of Equation 2. Tessendorf [2004a] use convolutions of specially-designed kernel functions on a regular grid. Subsequent research simplified the kernel computation by modifying the dispersion relation [Loviscach 2002, 2003], increased performance and stability with an exponential integrator [Tessendorf 2014], and increased its accuracy in the presence of boundaries and obstacles [Canabal et al. 2016].

Airy wave theory exhibits a convenient mathematical structure that benefits from numerous techniques from the analysis of partial differential equations. The irrotational and incompressible assumptions of Airy wave theory imply that the velocity is the gradient of a harmonic potential function $\phi$, further explained in Section 3. Researchers exploit this harmonicity using boundary integral [Keeler and Bridson 2014] and surface-only [Da et al. 2016] models for moving vertices on the liquid surface. Other researchers use fundamental solutions of this potential [Schreck et al. 2019] to analytically compute dispersive waves in open domains. Researchers have also exploited mathematical properties of the waves themselves by computing the evolution of wavefronts [Jeschke and Wojtan 2015], moving packets of wave energy [Jeschke and Wojtan 2017], and

stationary surface wavelets [Jeschke et al. 2018] to animate water waves interactively.

### 2.2 Non-linear water surface waves

Airy theory is sufficient for animating visually striking interference patterns, and its linearity gives rise to several computationally convenient mathematical transformations. However, this simplicity also prevents it from being useful for animating non-linear effects, like surface eddies, the fluid domain depending on wave height (waves sloshing up onto a beach), and the wave speed depending on wave height.

Direct simulation of 3D liquids is one way to achieve these non-linear effects, but they are prohibitively complex for simple scenarios where a heightfield simulation is sufficient. Thus we consider the rich literature of 3D fluid simulation in computer graphics out of scope for this discussion, and we will discuss only theoretical simplifications of the fluid equations which give rise to 2D equations. Boussinesq [1872] developed non-linear 2D equations with a polynomial expansion of the wave height, giving rise to a non-linear dependence between wave speed and wave height. While these equations are commonly used in coastal modeling, they are only applicable for very long wavelengths and are not commonly used in computer graphics.

Saint-Venant [1871] developed a 1D version of what we now consider the *shallow water equations (SWE)*. These equations are non-dispersive, but they allow turbulence in the form of swirling eddies, and their computation easily allows for a changing domain in flooding scenarios. Presumably for these reasons and for their ease of computation, SWE are popular in computer animation. Kass & Miller [1990] introduced a linearized SWE to graphics, and Layton & Van de Panne [2002] introduced a version of SWE with non-linear convective terms. Researchers subsequently refined SWE simulation with improved numerical integration and GPU implementation [Hagen et al. 2005], simulation on surface meshes [Wang et al. 2007], increased stability and non-reflecting boundary conditions [Chentanez and Müller 2010], and enhanced convection [Pan et al. 2012]. Azencot [2018] introduced a structure-preserving integrator for the Euler-Poincaré (EPDiff) equation, a particular class of SWE that does not exhibit the linear superposition of Airy wave theory.

All of these non-linear wave equations feature more interesting wave motions than simple linear Airy theory. However, they are either too computationally expensive to solve at large scales, or they are limited to a narrow range of phenomena (like only modeling shallow water).

### 2.3 Extended and combined methods

We are not the first to try to extend the capabilities of the Airy and SWE models beyond their original restrictive constraints. While the original Airy theory assumes that the steepness of the wave $kh$ is small, Trochoidal waves [Gerstner 1809; Rankine 1863] allow for steeper waves by allowing points on the surface to follow a circular path, and Biesel waves [Biesel 1952] approximate breaking waves by expanding these circular motions into ellipses. Both of these models have found use in computer graphics [Fournier and Reeves 1986; Tessendorf 2004b], where applications require visually interesting choppy waves. Unfortunately, while these modifications increase the visual appeal of surface waves in open water, they exacerbate the problems that Airy waves have with boundaries. Techniques like Wave Cages [Jeschke et al. 2020] can effectively prevent these waves from penetrating walls and sea floors, but until now no technique allows Airy waves to change their simulation domain (*e.g.*, by flooding into a new region).

The work of Yu et al. [2011] is noteworthy in that it advects strips of Airy interference patterns on top of a SWE simulation. The detailed surface waves utilize closed-form Airy wave interference patterns [Fedorov and Melville 1998], but the wave motion is governed by non-dispersive SWE and is limited to the special case of standing shock waves near an obstacle.

Several researchers aimed to compensate for the restrictions of 2D wave simulations by combining them with fully 3D liquid simulations. Thuerey et al. [2006] and Chentanez et al. [2015] coupled a liquid simulation with the shallow water equations, Huang et al. [2021] coupled 3D liquid to a quasi-2D surface-only water simulation [Da et al. 2016], and Schreck et al. [2022] coupled 3D liquid to an Airy wave solver based on fundamental solutions [Schreck et al. 2019]. These methods couple the 3D and 2D solvers using domain decomposition; some regions are simulated in 3D, while different regions (presumably further away or in areas requiring less detail) are limited to 2D simulation.

More recently, researchers used a different coupling strategy to add a 2D Airy solver directly on top of another fluid simulation solved with a different technique. Kim et al. [2013] added dispersive waves to a 3D fluid solver by one-way coupling a convolution-based solver with the closest-point method, and Skrivan et al. [2020] simulated the motion of Lagrangian wave packets [Jeschke and Wojtan 2017] on the 3D surface. These 2D/3D coupling techniques work exceptionally well when one can afford the computational demand of a fully 3D solver, but they are overkill for height field-based water dynamics. In addition, their one-way coupling strategy limits surface waves essentially to high-frequency textures, instead of a means to transport water into new domains. This paper addresses precisely these limitations.

Although methods like eWave [Tessendorf 2014] make it look easy, animating wake patterns is extremely difficult in general. SWE cannot create wake patterns, because all wavelengths travel at the same speed. 3D solvers are too dissipative and low-resolution for interference patterns to appear, so specialized 2D surface wave methods are the only way to create wake patterns in practice. Thus, our work is the only practical method for animating two-way coupling effects like those in Figure 3 (wakes interacting with vortices) and Figure 1 (wakes interacting with the shore), and it does so in real time.

## 3 THEORETICAL FRAMEWORK

Here we provide additional fluid mechanics background in Sections 3.1, 3.3, and 3.2, then introduce theoretical concepts behind our method in Sections 3.4 and 3.5.

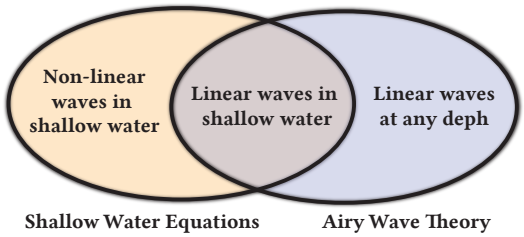


Fig. 2. *SWE vs. Airy wave theory.* The shallow water equations (SWE) are capable of complex non-linear wave behaviors, but are limited to constant wave speeds in shallow depths. Airy wave theory models waves at all possible depths, but they are restricted to small-amplitude linear waves. Our method combines both, significantly expanding the range of behaviors possible within a single heightfield method.

### 3.1 Notation

For the remainder of this paper we assume the water is a height field $h(\mathbf{x})$ over 2D space $\mathbf{x} = (x, y)$ with surface velocity $\mathbf{u} = (u, v)$. Wave theory often decomposes the wave height function into an average water depth $\bar{h}$ and a perturbed height function $\tilde{h}$, such that $h = \bar{h} + \tilde{h}$, where most of the interesting wave dynamics happens to $\tilde{h}$, while $\bar{h}$ is relatively static. We will use similar notation to decompose other functions, with a $-$ over averaged quantities and $\sim$ over perturbed quantities.

When discussing waves, $\mathbf{k}$ is the wavevector with wavenumber $k = \|\mathbf{k}\|$, wavelength $\lambda = 2\pi/k$, angular frequency $\omega$, and amplitude $a$. Airy theory assumes a velocity potential $\phi$ such that $\mathbf{u} = \nabla\phi$.

### 3.2 Linear surface waves

According to Airy wave theory [Johnson 1997], the vertical displacement $\tilde{h}$ from a reference surface with water depth $h$ is

$$\tilde{h} = a\cos(\mathbf{k}\cdot\mathbf{x} - \omega t) \tag{4}$$

with $\omega$ defined in Equation 2. The velocity potential $\phi$ is defined as

$$\phi = a\frac{\omega}{k}\frac{\cosh(k(z+h))}{\sinh(kh)}\sin(\mathbf{k}\cdot\mathbf{x} - \omega t) \tag{5}$$

where $z$ is the depth coordinate. The horizontal velocity at any water depth $z$ is defined as

$$\frac{\partial\phi}{\partial\mathbf{x}} = a\omega\frac{\mathbf{k}}{k}\frac{\cosh(k(z+h))}{\sinh(kh)}\cos(\mathbf{k}\cdot\mathbf{x} - \omega t). \tag{6}$$

Integrating this velocity along the water depth gives us the horizontal volumetric flow rate in linear wave terms:

$$\tilde{q} = \int_{z=-h}^{\tilde{h}}\frac{\partial\phi}{\partial\mathbf{x}}\,dz \approx \int_{z=-h}^{0}\frac{\partial\phi}{\partial\mathbf{x}}\,dz = a\frac{\omega}{k}\cos(\mathbf{k}\cdot\mathbf{x} - \omega t). \tag{7}$$

Integrating along depth to 0 instead of $\tilde{h}$ simplifies the terms and makes $\tilde{\mathbf{q}}$ independent of water depth. This approximation is consistent with the $ka \ll 1$ assumption of Airy theory.

### 3.3 Shallow water equations

In the shallow water regime ($h \ll \lambda$), the water surface is assumed to be so close to the sea floor that the velocity is roughly constant throughout an entire column of water. The shallow water equations in conservative form are:

$$\frac{\partial h}{\partial t} + \nabla\cdot\mathbf{q} = 0 \tag{8}$$

$$\frac{\partial\mathbf{q}}{\partial t} + \nabla\cdot(\mathbf{q}\otimes\mathbf{u}) = -\nabla\left(\frac{1}{2}gh^2\right) \tag{9}$$

where $\mathbf{q} = h\mathbf{u}$ is the horizontal flow rate, and $\otimes$ is the dyadic product. In our implementation, $\mathbf{q}$ is the amount of liquid flowing from one grid cell to another, through a rectangle of height $h$ and width $\Delta x$. These equations have transport terms which are non-linear in $\mathbf{u}$, $\mathbf{q}$, and $h$. If we linearize these equations, we arrive at the popular "pipe" model [Kass and Miller 1990], which is a wave equation with dispersion relation $\omega = k\sqrt{gh}$. As an interesting aside, these linear waves are actually a subset of Airy wave theory, which prescribes $\omega = \sqrt{gk\tanh(kh)}$, because $\tanh(kh) \approx kh$ for small values of $kh$. SWE and Airy theory overlap exactly in the linearized shallow water regime, but SWE is better for non-linear behaviors, and Airy theory is better for deeper water. This range of behaviors is conceptually illustrated in Figure 2. Figure 3 shows how our method can simulate both non-linear and Airy effects together.

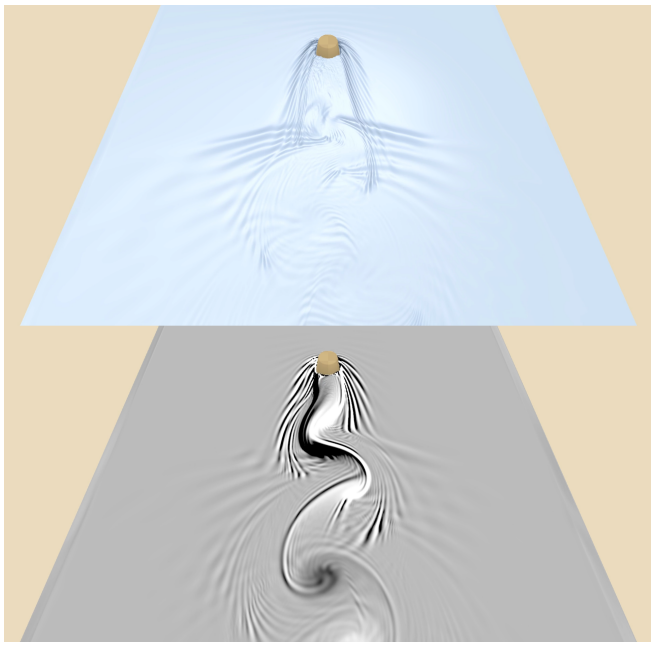

Fig. 3. *Flow past a cylinder.* Our method combines surface waves with features from the bulk flow. Here, a disturbance in the flow creates a turbulent wake featuring both wave interference patterns and a vortex street. The lower image visualizes the 2D vorticity in grayscale.

### 3.4 Dynamics of decomposed waves

We generalize these wave dynamics by incorporating two different types of waves together in the same mathematical model. We first decompose the water height $h = \bar{h} + \tilde{h}$ as mentioned earlier. As before, $\tilde{h}$ is a height field with relatively high frequencies compared to $\bar{h}$: it models small ripples on top of $\bar{h}$, and it obeys Airy theory.

Then, instead of assuming that $\bar{h}$ is a static average water depth, we consider it a more general low-frequency water height function modeling the "bulk" of the fluid. $\bar{h}$ is allowed to move over time according to fluid dynamics. To tractably model these dynamics, we assume that the low-frequency $\bar{h}$ function consists of wavelengths so long that $h \ll \lambda$ and it obeys shallow water dynamics. We then seek to know how the dynamics of $\bar{h}$ affect $\tilde{h}$ and vice-versa.

We first note that linear wave dynamics obey the superposition principle: if $\bar{h}$ and $\tilde{h}$ are both Airy waves, then so is $h = \bar{h} + \tilde{h}$, and the two waves have no influence on each other. In other words, it is trivial to combine these two flows together when $\bar{h}$ is calm and obeys linearized SWE.

Although we restrict $\tilde{h}$ to linear wave theory, we allow $\bar{h}$ to be nonlinear. We can estimate the effect of *non-linear* terms by plugging $h = \bar{h} + \tilde{h}$ and $\mathbf{u} = \bar{\mathbf{u}} + \tilde{\mathbf{u}}$ into the continuity equation (Equation 8):

$$\begin{aligned}\frac{\partial h}{\partial t} &= -\nabla \cdot (h\mathbf{u}) \\ &= -\nabla \cdot \Big( (\bar{h} + \tilde{h})(\bar{\mathbf{u}} + \tilde{\mathbf{u}}) \Big) \\ &= \underbrace{-\nabla \cdot (\bar{h}\bar{\mathbf{u}})}_{\text{bulk}} \underbrace{-\nabla \cdot (\tilde{h}\tilde{\mathbf{u}})}_{\text{surface}} \underbrace{-\nabla \cdot (\tilde{h}\bar{\mathbf{u}})}_{\text{transport}} \underbrace{-\cancel{\nabla \cdot (\bar{h}\tilde{\mathbf{u}})}}_{\text{non-linear}}. \end{aligned} \tag{10}$$

The first term represents advection through the bulk flow, which we model directly by solving the shallow water equations for $\bar{h}$ and $\bar{\mathbf{u}}$. The second term is the evolution of perturbed variables, which we model with Airy wave theory. The third term is the transport of high-frequency surface displacements $\tilde{h}$ through the smooth bulk velocity field $\bar{\mathbf{u}}$, which we explicitly model with an additional advection step. Finally, the fourth term is the influence of the surface wave velocity $\tilde{\mathbf{u}}$ across the entire water depth $\bar{h}$; this term is zero by our assumption that $\tilde{\mathbf{u}}$ obeys Airy wave theory and cannot have large enough amplitudes to affect the entire depth $\bar{h}$. We therefore do not model it in this work. In the end, we are left with 3 terms that model bulk flow, surface waves, and transport.

Although this discussion focused on the evolution of $h$, similar reasoning applies for the transport of flow rates $\mathbf{q}$ (left-hand side of Equation 9), giving us three analogous terms for the update of $\mathbf{q}$. Henceforth, we use $-$ to denote "bulk" terms, $\sim$ to denote "surface" terms, and $\simeq$ to denote "transport" terms.

### 3.5 Decomposing waves

We want to decompose $h$ and $\mathbf{q}$ such that their low-frequency components $\bar{h}$ and $\bar{\mathbf{q}}$ satisfy the shallow water assumption $h \ll \lambda$. There is no exact wavelength cut-off for this property to hold, but we know that it should scale monotonically with water depth $h$. This will allow, for example, long waves to still obey pure Airy theory in deep water, and short waves to obey SWE if the water is shallow enough. We also wish to prevent extremely steep waves from appearing in $\tilde{h}$, as they violate the small amplitude $ka \ll 1$ assumption of Airy wave theory, and they are better treated as shock waves in the non-linear SWE via $\bar{h}$.

Rather than filter the raw water depth function $h$, we consider the height field $H = h + \tau$, which combines the depth and terrain height $\tau$. We do this because a smooth $H$ will give us a physically reasonable water surface, regardless of what the terrain looks like below. We use a diffusion equation as a low-pass filter [Hamming 1998]:

$$\frac{\partial H}{\partial T} = \nabla \cdot (\alpha \nabla H) \tag{11}$$

where $\alpha$ is a spatially-varying diffusion coefficient, and $T$ is the fictitious time over which we integrate the diffusion. Fourier analysis of this equation reveals that it filters out high frequencies proportional to the diffusion coefficient $\alpha$, the wavenumber squared, and the integrated time $t$:

$$\hat{H}(k, T_0 + T) = \hat{H}(k, T_0) e^{-k^2 \alpha T} \tag{12}$$

where $\hat{H}(k, T)$ is the Fourier component of $H$ corresponding to wavenumber $k$ at fictitious time $T$. Thus diffusion acts as a filter which removes the high-frequency components $\tilde{H}$ from $H$ to get a low-frequency $\bar{H}$. We then recompose $\bar{h} = \bar{H} - \tau$ and $\tilde{h} = h - \bar{h}$ from the filtered height field. The filter strength depends on $\alpha T$, which should increase monotonically with water depth and decay with wave steepness. We use the same technique to compute $\mathbf{q} = \bar{\mathbf{q}} + \tilde{\mathbf{q}}$.

We choose to decompose the $h$ and $\mathbf{q}$ variables (and reconstruct them again) every simulation step, because the wave frequencies may dramatically change over time. By recomputing the wave decomposition, we allow waves to adapt their behaviors as they move around, change frequencies, and change water depth.

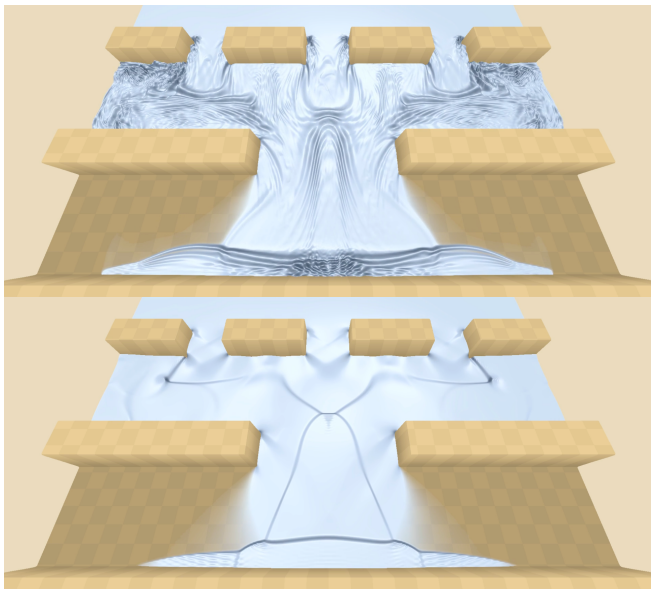

Fig. 4. *Triple dam break.* Simulated wakes, reflections, and eddies form naturally as the flow accelerates through narrow channels and expands again afterward. The lower image shows a SWE simulation for comparison.

## 4 OUR GENERALIZED WAVE MODEL

Following Chentanez and Müller and others [Chentanez and Müller 2010; Stelling and Duinmeijer 2003], we use a staggered grid discretization of the water heightfield, with $h$ stored on grid cell centers, and components of velocity and flow rate stored on cell boundaries. The water heights are time-integrated with the explicit Verlet leapfrog scheme, so $h$ is evaluated one half time step later than

**Algorithm 1:** Overview for one time step

Input $\mathbf{q}^t$ and $h^{t+\Delta t/2}$
$\left\{\bar{\mathbf{q}}^t, \tilde{\mathbf{q}}^t, \bar{h}^{t+\Delta t/2}, \tilde{h}^{t+\Delta t/2}\right\}$ := Decompose $\mathbf{q}^t$ and $h^{t+\Delta t/2}$
$\bar{\mathbf{q}}^{t+\Delta t}$ := Simulate bulk fluid flow
$\tilde{\mathbf{q}}^{t+\Delta t}$ := Simulate Airy waves
$\tilde{\bar{\mathbf{q}}}^{t+\Delta t}$ := Transport surface flow rate $\tilde{\mathbf{q}}^{t+\Delta t}$ through $\bar{\mathbf{u}}$
$\tilde{\bar{h}}^{t+3\Delta t/2}$ := Transport surface height $\tilde{h}^{t+\Delta t/2}$ through $\bar{\mathbf{u}}$
$\mathbf{q}^{t+\Delta t}$ := Compute flow rate from $\left\{\bar{\mathbf{q}}^{t+\Delta t}, \tilde{\bar{\mathbf{q}}}^{t+\Delta t}\right\}$
$h^{t+3\Delta t/2}$ := Compute wave height from $\left\{\mathbf{q}^{t+\Delta t}, \tilde{\bar{h}}^{t+3\Delta t/2}\right\}$

$\mathbf{q}$. This staggered discretization represents a *finite volume* scheme which enhances simulation stability and guarantees volume conservation; each substep of the algorithm accumulates the flow rates $\mathbf{q}$ needed to update the wave heights $h = \bar{h}+\tilde{h}$ and the fluid momentum encoded as $\mathbf{q} = \bar{\mathbf{q}} + \tilde{\mathbf{q}}$. At the end of each time step, we reconstruct the final $h$ from these integrated flow rates, guaranteeing volume preservation, and resulting in complex flows like Figure 4.

This section is divided into sub-sections representing major substeps of our algorithm, which is also outlined in Algorithm 1. We first decompose $h$ and $\mathbf{q}$ into their bulk and surface components (Section 4.1), then simulate the bulk components via the shallow water equations (Section 4.2), and simulate the surface components with an Airy wave solver (Section 4.3). We transport the surface variables to account for the third term in Equation 10 (Section 4.4), and finally integrate the overall height $h$ through time at the end of the time step (Section 4.5).

## 4.1 Wave decomposition

To decompose $h$ and $\mathbf{q}$ into bulk and surface components, we follow Section 3.5 by implementing a forward-in-time centered-in-space explicit diffusion solver on the GPU. This operation converts $h$ and $\mathbf{q}$ into $\bar{h}$ and $\bar{\mathbf{q}}$, which we use for the bulk flow. We use $\tilde{h} = h - \bar{h}$ and $\tilde{\mathbf{q}} = \mathbf{q} - \bar{\mathbf{q}}$ for computing surface waves. By increasing the integration time and diffusion strength $\alpha$, we remove more and more high frequencies from $\bar{h}$ and $\bar{\mathbf{q}}$ — no diffusion produces a pure SWE simulation with $h = \bar{h}$, and diffusing all the way to a steady state produces a pure Airy wave simulation with $h = \tilde{h}$.

In theory, waves should always travel slower in shallow water, but blindly applying SWE in deep water makes them travel impossibly fast. We employ a frequency filter to minimize this nonphysical behavior: for a given water depth, there exists a wavelength where the shallow water wave speed exactly equals the deep water wave speed, $\lambda_{\text{cutoff}} = 2\pi h$. Wavelengths shorter than $\lambda_{\text{cutoff}}$ should not be treated with SWE. We employ diffusion as a classical Gaussian filter with "cutoff wavelength" (where power is reduced by 3db) equal to $\lambda_{\text{cutoff}}$, giving us a diffusion coefficient $\alpha = \frac{h^2}{64}$, as derived in Appendix C. Setting $\alpha$ to this function of depth turns the diffusion equation into a low pass filter, preventing short wavelengths in deep water from being modeled with SWE.

To discourage very large, steep waves from being mis-classified as high-frequency surface ripples, we add another penalty term $e^{-d|\nabla h|^2}$ to get

$$\alpha = \frac{h^2}{64} \cdot e^{-d|\nabla h|^2} \tag{13}$$

with tuning parameter $d = \frac{1}{100}$. This heuristic discourages steep gradients in surface waves and otherwise achieves monotonic depth-dependent decomposition. We find that the gradient penalty $e^{-d|\nabla h|^2}$ is only needed in extreme situations in practice, like dam breaks where the height field is a step function with arbitrarily high frequencies. The effect of penalizing steep waves in the decomposition is visualized in Figure 5.

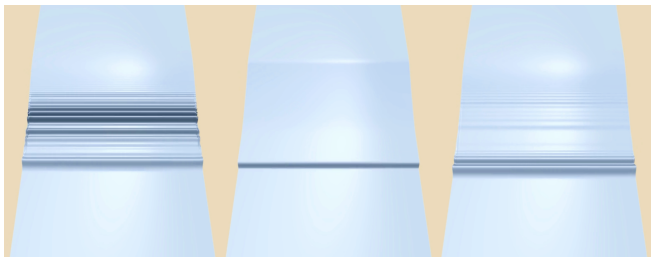

Fig. 5. *Gradient penalty evaluation.* This example started with an extremely steep wave (a step-function). Without the gradient penalty term in Equation 13 (left), the surface explodes into a large number of steep high-frequency surface waves. Enabling that term (right) gives results that look more like a SWE simulation (center) with enriched details.

To simulate this diffusion, we sample $\alpha$ on a staggered grid between $h$ samples. This diffusion scheme is an effective filter for our purposes, but its implementation has practical limitations. Although our system runs in real-time, this decomposition is the most expensive part of our algorithm. Explicit integration requires small time steps (128 sub-steps during each time step of wave simulation) and we explicitly clamp $\Delta T$ to its maximum stable value to avoid instability in arbitrarily deep water. A more efficient decomposition (better diffusion solver or different filtering operation) should accelerate computation further.

## 4.2 Simulating bulk fluid flow

Given our newly decomposed $\bar{\mathbf{q}}^t$ along with $\bar{h}^{t-\Delta t/2}$ from the previous time step, we compute $\bar{\mathbf{u}}^t = \bar{\mathbf{q}}^t / \bar{h}^{t-\Delta t/2}$; this $\bar{h}$ is stored in cell centers while $\bar{\mathbf{q}}$ is stored on cell boundaries, so we use first-order up-winding to select the most appropriate $\bar{h}$. We then numerically integrate the shallow water equations to update $\bar{\mathbf{u}}^{t+\Delta t}$; for this purpose, we adopt the conservative finite-volume scheme of Stelling and Duinmeijer [2003]. This method gives explicit time stepping rules for updating $\bar{h}$ and $\bar{\mathbf{u}}$, which we reproduce in Appendix A. Please see Section 5 of [Stelling and Duinmeijer 2003] for full implementation details.

At the end of this operation, we have new $\bar{\mathbf{u}}^{t+\Delta t}$ values evaluated at the boundaries of grid cells. We convert back to $\bar{\mathbf{q}}^{t+\Delta t} = \bar{\mathbf{u}}^{t+\Delta t}\bar{h}^{t+\Delta t/2}$, with the most recent $\bar{h}$ given by the surface decomposition in the previous section, again using first order up-winding to evaluate it on the staggered grid. Our decision to use $\bar{h}$ and $\bar{\mathbf{q}}$ from different steps is done such that it exactly reproduces the leapfrogged SWE simulation of [Stelling and Duinmeijer 2003] in

the simplified case of pure shallow water. After this operation, we have computed the flow rates $\tilde{\mathbf{q}}^{t+\Delta t}$ corresponding to the "bulk" term in Equation 10.

### 4.3 Simulating Airy waves

We use the exponential integrator of eWave [2014] to update the flow rates $\tilde{\mathbf{q}}$ using the derivatives of surface wave heights $\tilde{h}$. We present pseudocode for this step in Algorithm 2, where *FFT* is the fast Fourier transform computed globally over the whole domain, with *iFFT* as its inverse; the hat ˆ notation indicates that the function exists in Fourier space.

At the start of Algorithm 2, the exponential integrator expects both $\tilde{h}$ and $\tilde{q}$ to be sampled at the same time $t$, but the SWE solver from the previous section uses Leapfrog integration, leaving $\tilde{h}$ and $\tilde{q}$ staggered in time. We re-sample $\tilde{h}$ at time $t$ via linear interpolation, then compute its Fourier transform to get $\hat{\tilde{h}}^t$.

In the third step of Algorithm 2, $\omega$ is the dispersion relation from Equation 2. We use the decomposed smooth water depth $\bar{h}$ from Section 4.1 for this purpose, so that $\omega = \sqrt{gk \tanh(k\bar{h})}$. However, the spatially varying water depth $\bar{h}$ is not available in Fourier space, so we approximate the behavior of waves at different water depths by calculating $\tilde{\mathbf{q}}_i^{t+\Delta t}$ for $N$ different water depths ($N = 4$ in our implementation with $\bar{h}_i \in \{1m, 4m, 16m, 64m\}$, refer to Figure 8 for an error analysis), evaluating $\tilde{\mathbf{q}}_i^{t+\Delta t}$ for each value of $\bar{h}_i$, and linearly interpolating the $\tilde{\mathbf{q}}_i^{t+\Delta t}$ corresponding to the two closest depths.

The calculation of $\tilde{\mathbf{q}}_i^{t+\Delta t}$ needs the derivative $\frac{\partial \hat{\tilde{h}}^t}{\partial \mathbf{x}}$, which we compute directly in Fourier space [Johnson 2011]. Because it is sampled on a staggered grid compared to $\tilde{\mathbf{q}}$, we off-set $\frac{\partial \hat{\tilde{h}}^t}{\partial x}$ by half a grid cell in Fourier space via the Fourier shift theorem (multiplying $\hat{\tilde{h}}$ by $e^{-ik\Delta x/2}$).

*4.3.1 Eliminating numerical dispersion.* We mentioned at the start of this section that the final wave heights $h$ will be reconstructed in a volume-preserving manner from the flow rates $\mathbf{q}$ at the end of the time step. This finite-volume reconstruction is based on a numerical approximation of the divergence operator in Equation 8, and it causes numerical dispersion. Consequently, our surface waves do not actually obey the desired dispersion relation if we set $\omega = \sqrt{gk \tanh(k\bar{h})}$. In Appendix B we derive the numerical dispersion and a correction factor $\beta$, so that we achieve perfectly accurate wave speeds when we set $\omega = \frac{1}{\beta}\sqrt{gk \tanh(k\bar{h})}$ in Algorithm 2. Figure 6 illustrates the impact of this correction where we measure wavespeed with standing waves in a rectangular pool similar to [Schreck and Wojtan 2022] at a water depth of 4 m, a gridcell size of 1 m, and a timestep $\Delta t = \frac{1}{60}s$.

**Algorithm 2:** Simulate Airy Waves

$\hat{\tilde{h}}^t := FFT\left[\frac{\tilde{h}^{t-\Delta t/2} + \tilde{h}^{t+\Delta t/2}}{2}\right]$

$\hat{\tilde{\mathbf{q}}}^t := FFT[\tilde{\mathbf{q}}^t]$

$\tilde{\mathbf{q}}^{t+\Delta t} := iFFT[\cos(\omega\Delta t)\hat{\tilde{\mathbf{q}}}^t - \sin(\omega\Delta t)\frac{\omega}{k^2}\frac{\partial \hat{\tilde{h}}^t}{\partial x}]$

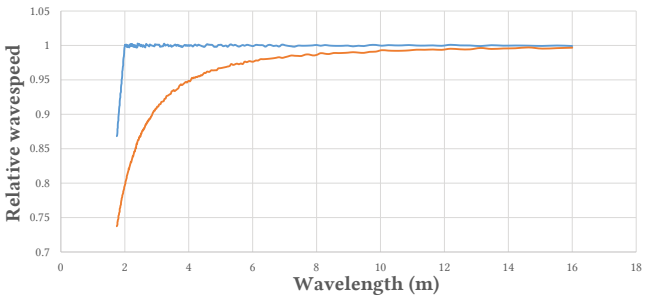


Fig. 6. *Surface wave dispersion error.* We plot the measured wavespeed $\frac{\omega}{k}$ relative to the exact value; a value less than 1 indicates artificially slow waves, while a value greater than 1 means the waves are too fast. The uncorrected simulation (orange) exhibits numerical dispersion, while our corrected method (blue) is practically perfect even down to the Nyquist limit of $\lambda = 2\Delta x$.

At the end of this step, we have updated flow rates $\tilde{\mathbf{q}}^{t+\Delta t}$ corresponding to the "surface" term in Equation 10.

### 4.4 Transporting surface waves

Next we transport the surface flow rates $\tilde{\mathbf{q}}$ and wave heights $\tilde{h}$ through the bulk velocity $\bar{\mathbf{u}}$:

$$\frac{\partial \tilde{h}}{\partial t} = -\nabla \cdot (\tilde{h}\bar{\mathbf{u}}) = -\bar{\mathbf{u}} \cdot \nabla \tilde{h} - \tilde{h}(\nabla \cdot \bar{\mathbf{u}}) \tag{14}$$

$$\frac{\partial \tilde{\mathbf{q}}}{\partial t} = -\nabla \cdot (\tilde{\mathbf{q}} \otimes \bar{\mathbf{u}}) = -\bar{\mathbf{u}} \cdot \nabla \tilde{\mathbf{q}} - \tilde{\mathbf{q}}(\nabla \cdot \bar{\mathbf{u}}). \tag{15}$$

We choose to apply operator splitting to these equations and numerically integrate each term separately, and we evaluate the derivatives at the midpoint of the timestep, in the style of a "leapfrog" integrator. The $\nabla \cdot \bar{\mathbf{u}}$ term amplifies or damps waves based on the convergence or divergence of the underlying bulk flow. By holding $\bar{\mathbf{u}}$ constant over the time step, this becomes a linear ODE that we integrate in closed form using exponential integration. In nature, this term is responsible for wave shoaling — the sharp steepening, and eventual tumbling, of surface waves when they enter shallower water with slower flows. Our model cannot handle breaking effects, so we follow previous works (which artificially damp steep waves [Jeschke and Wojtan 2015] or omit the growth term altogether [Tessendorf 2017]) by adding an optional user-tunable factor $\gamma$ that damps this term only when it amplifies the wave in a converging background flow. We compute the $\bar{\mathbf{u}} \cdot \nabla$ terms using Semi-Lagrangian advection with cubic spatial interpolation [Fedkiw et al. 2001] and avoid overshooting by clamping the interpolated value to lie within the range of the four neighboring values.

For convenience, we transport the most up-to-date version of $\tilde{q}$ (after simulating Airy waves in Algorithm 2), so the resulting term $\tilde{\mathbf{q}}$ encodes both the "surface" and "transport" terms of the flow-rate version of Equation 10. In contrast, $\tilde{h}$ is not directly updated in Algorithm 2, so it is not available to be combined with the transport term. Thus, $\tilde{h}$ encodes only the "transport" term in Equation 10. The detailed steps for transporting surface flow rates $\tilde{\mathbf{q}}$ and heights $\tilde{h}$ are written in Algorithms 3 and 4.

**Algorithm 3:** Transport surface flow rate $\tilde{\mathbf{q}}$ through $\bar{\mathbf{u}}$

Input $\tilde{\mathbf{q}}^{t+\Delta t}$, $\bar{\mathbf{u}}^{t}$, and $\bar{\mathbf{u}}^{t+\Delta t}$
Input wave amplification damping factor $\gamma = 1/4$
$\bar{\mathbf{u}}^{t+\Delta t/2} := \left(\bar{\mathbf{u}}^{t} + \bar{\mathbf{u}}^{t+\Delta t}\right)/2$
$G^{t+\Delta t/2} := \min\left[-(\nabla \cdot \bar{\mathbf{u}}^{t+\Delta t/2}), -\gamma(\nabla \cdot \bar{\mathbf{u}}^{t+\Delta t/2})\right]$
$\tilde{\mathbf{q}}^{t+\Delta t} := \tilde{\mathbf{q}}^{t+\Delta t}\exp\left(G^{t+\Delta t/2}\Delta t\right)$
$\tilde{\mathbf{q}}^{t+\Delta t} :=$ Semi-Lagrangian advect $\tilde{\mathbf{q}}^{t+\Delta t}$ through $\bar{\mathbf{u}}^{t+\Delta t/2}$

**Algorithm 4:** Transport surface height $\tilde{h}$ through $\bar{\mathbf{u}}$

Input $\tilde{h}^{t+\Delta t/2}$ and $\bar{\mathbf{u}}^{t+\Delta t}$
Input wave amplification damping factor $\gamma = 1/4$
$G^{t+\Delta t} := \min\left[-(\nabla \cdot \bar{\mathbf{u}}^{t+\Delta t}), -\gamma(\nabla \cdot \bar{\mathbf{u}}^{t+\Delta t})\right]$
$\tilde{\tilde{h}}^{t+3\Delta t/2} := \tilde{h}^{t+\Delta t/2}\exp\left(G^{t+\Delta t}\Delta t\right)$
$\tilde{\tilde{h}}^{t+3\Delta t/2} :=$ Semi-Lagrangian advect $\tilde{\tilde{h}}^{t+3\Delta t/2}$ through $\bar{\mathbf{u}}^{t+\Delta t}$

### 4.5 Merging bulk and surface flow components

At this point in the time step we have computed all of the bulk, surface, and transport terms needed to update $h$ and $\mathbf{q}$ according to Equation 10. The integrated bulk flow rate is stored in $\bar{\mathbf{q}}^{t+\Delta t}$, and the surface wave flow rates were first updated then transported and stored in $\tilde{\mathbf{q}}^{\mathbf{t+\Delta t}}$. The final $\mathbf{q}$ value is simply the sum of these terms:

$$\mathbf{q}^{t+\Delta t} = \bar{\mathbf{q}}^{t+\Delta t} + \tilde{\mathbf{q}}^{\mathbf{t+\Delta t}}. \tag{16}$$

Next we wish to compute the final height $h$. To guarantee volume conservation during this process, we construct $h$ from the divergence of flow rates (Equation 8) rather than updating it directly. The flow rates corresponding to the bulk and surface terms in Equation 10 have just recently been collected and stored in $\mathbf{q}^{t+\Delta t}$, so all that remains is to compute a new flow rate $\breve{\mathbf{q}}$ corresponding to the "transport" term for updating $h$. We do this by multiplying the transported height by the bulk velocity: $\breve{\mathbf{q}} = \tilde{\tilde{h}}\bar{\mathbf{u}}$. Recall that $\mathbf{q}$ and $\mathbf{u}$ are sampled at time $t + \Delta t$ and are located on a staggered grid, so we need to re-sample $\tilde{\tilde{h}}^{t+3\Delta t/2}$ in space and time to make it coincide with $\bar{\mathbf{u}}^{t+\Delta t}$. To perform this re-sampling, we re-purpose our semi-Lagrangian advection algorithm to transport $\tilde{\tilde{h}}^{t+3\Delta t/2}$ backward in time by $\Delta t/2$, to get a $\tilde{\tilde{h}}^{t+\Delta t}$ which is stored on the boundaries between grid cells. We then compute $\breve{q}^{t+\Delta t} = \tilde{h}^{t+\Delta t}\bar{\mathbf{u}}^{t+\Delta t}$.

We plug these flow rates into Equation 8 to produce the final wave heights:

$$h^{t+3\Delta t/2} = h^{t+\Delta t/2} + \Delta t\nabla \cdot \left(\mathbf{q}^{t+\Delta t} + \breve{\mathbf{q}}^{t+\Delta t}\right). \tag{17}$$

The time step is now finished, and these $\mathbf{q}$ and $h$ variables will now feed into the next time step, where they will get decomposed into bulk and surface components and updated all over again.

### 4.6 Additional implementation details

To interact with dry regions, our algorithm prescribes solid boundary conditions as in [Chentanez and Müller 2010]: terrain that is higher than the water level sets the flow rate $\mathbf{q}$ to zero (effectively creating reflecting boundary conditions); flow rates in dry regions can otherwise be overwritten by the shallow water solver, in order to flood into new terrain. We perform this update once per time step, so the maximum flood rate is limited by the CFL condition. Using these boundary conditions allow us to compute the FFT in Algorithm 2 over the entire square domain (including both wet and dry regions), without any further special treatment for irregular water domains.

To avoid instabilities in the presence of fast flows, we use the CFL condition to clamp velocities and flow rates to their maximum stable values $|\mathbf{u}|_{\max} = \Delta x/(4\Delta t)$ and $|\mathbf{q}|_{\max} = h\Delta x/(4\Delta t)$. We apply this limiter wherever any velocity and flow rate variables are updated: after SWE integration, after transport of $\tilde{h}$ and $\tilde{\mathbf{q}}$, and after summing flow rates in Equation 16. This limiter guarantees SWE stability at the expense of incorrect flow speeds at large time steps. This is not a problem in our experience, though it can cause two small visual artifacts: slower flooding than normal, and an artificially large wave height in places where the water would spread out quickly if it was not throttled.

For extra visual effect in Figure 1, we add a small amount of normal-mapped "texture" to the wave surfaces by computing low-amplitude deep water waves via FFT [Mastin et al. 1987], and adding the resulting surface normal to our water surface. We also use the technique of Tessendorf [2017] to add horizontal displacements to our waves in this example, in addition to the vertical wave displacement $h$. We render the waves with a transparency that scales with depth, so the water becomes more transparent in shallow depths.

To produce boat wakes in our examples, we add point disturbances in $\mathbf{q}$ at every time step beneath the path of the boat. We also move the boat's vertical position to lie on the water level $h$. We leave more thorough solid-fluid coupling to future work.

## 5 RESULTS AND DISCUSSION

We implemented our wave simulation algorithm in CUDA on an NVIDIA RTX2080 Max-Q Laptop. Our examples run in real-time (comfortably over 40fps), including the cost of rendering. Without rendering, our simulation runs around 100fps. The wave decomposition algorithm (explicit integration of a diffusion equation) takes 87% of the simulation time and is the clear bottleneck of our solver. For our results we set $\Delta x$ to 1m and the grid size to $512 \times 512$; we set $\Delta t$ to 1/60s for Figures 1 and 3 and to 1/30s for our other examples.

Figure 1 shows how our algorithm computes both deep water and shallow water effects in the same scene. Two boats travel at different speeds through deeper water, producing significantly different wake patterns. An incoming wave floods onto the shore and creates a new pool of water.

Figure 4 features a dam break scenario, where a wall with three open slots separates a wet domain from a dry one. The water floods through the narrow channels, creating standing wake patterns, vortices, and numerous reflected waves on the water surface. Our method uses the bulk flow to model the flooding, and the Airy waves simulate the surface details.

Figure 3 models a cylindrical pillar in a steady background flow of moving at 0.45 pillar diameters per second. The pillar sheds vortices/whirlpools and forms a von Kármán vortex street, as well as

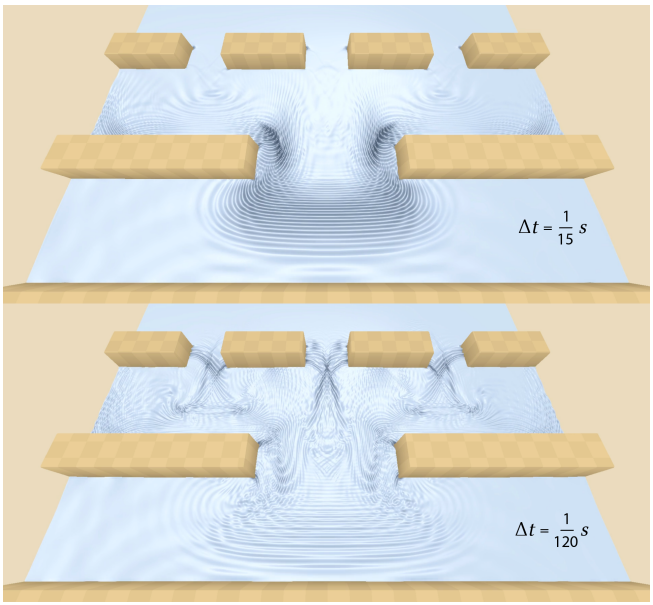


Fig. 7. *Varying* $\Delta t$. Smaller time steps resolve more high-frequency waves, with a more accurate flooding speed.

a Kelvin wake pattern. The whirlpools and surface waves interact with each other. In our model, the bulk flow simulates (shallow water) vortices, and the Airy wave model creates the wake interference patterns. Finally, Figure 10 features waves spilling in from the boundaries of a square pool and interacting with complex obstacles that dynamically grow and change over time.

## 5.1 Stability and validation

To make our method robust enough for real-time simulation, we take care to keep each of the substeps of our algorithm as numerically stable as possible. The SWE solver enforces stability by clamping flow rates to their maximum stable value, which can be inaccurate for large time steps. The Airy solver and wave transport algorithms (Semi-Lagrangian advection and closed-form wave amplification) are each unconditionally stable when taken in isolation. The final wave height integration guarantees volume conservation exactly by construction. We merge these various methods together using operator splitting, which is generally first order accurate and *not* guaranteed to be stable.

We analyze the effect of varying the time step size in Figure 7, using $\Delta t = 1/120$s and $1/15$s. Simulations with smaller timesteps produce smaller wavelength waves which endure longer, implying that our method exhibits some numerical damping for large $\Delta t$. Conversely, the large wavelengths can amplify in the simulations with larger time steps, because artificially clamping $\bar{u}$ for stability can cause velocity to slow down, potentially reducing $\nabla \cdot \bar{u}$ and increasing $\partial \bar{h}/\partial t$.

Section 4.3 uses a piecewise linear function to sample surface wave behaviors at different water depths. Figure 8 shows that the accuracy of this approach depends on how many samples we use: using more samples makes for more accurate wave speeds, at the

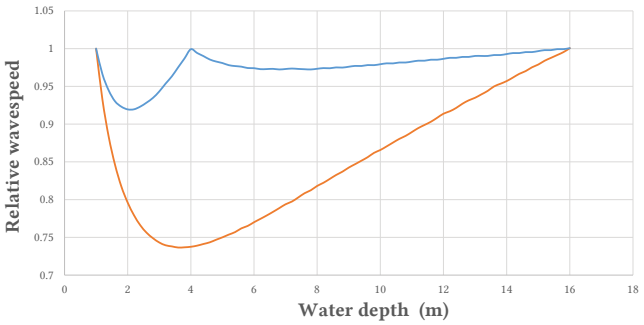


Fig. 8. *Error from discretizing water depth.* We plot the measured wavespeed $\frac{\omega}{k}$ of a wave with wavelength $\lambda = 4m$ relative to the exact value for different water depths. The orange curve shows the error from a simulation which samples $\omega$ at only two depths (1m, 16m), and the blue curve at three depths, respectively (1m, 4m, 16m).

expense of additional surface wave simulations. Our simulations use four depth samples, giving a maximum of 8% error (less in practice, because most surface waves do not depend much on water depth). One could further optimize these discrete values to minimize the error on all relevant water depth and wavelength combinations, but we did not deem this necessary for satisfactory visual results.

Figure 9 shows the relationship between the accuracy of the surface wave speed and the wave decomposition (Section 4.1). The different curves represent different numbers of iterations in the diffusion solver each time step. Diffusing by a larger amount makes a more effective frequency filter, so we see a relationship between the number of explicit diffusion steps and the accuracy of the wave speeds. Less diffusion causes a larger proportion of waves to be treated as shallow water with a faster wave speed; a larger diffusion lowers this error, turning most of the simulation into Airy waves.

To measure the accuracy of our method's effective dispersion relation in Figures 6, 8, and 9, we measured $\omega$ by counting the period of standing waves at different wavelengths, just like Schreck & Wojtan [2022].

Lastly, we note that our framework generalizes both SWE and Airy wave solvers; if we force our wave decomposition to set $\bar{h} = h$ and $\bar{\mathbf{q}} = \mathbf{q}$, then we get a SWE simulator identical to that of [Stelling and Duinmeijer 2003]. Conversely, if we set $\bar{h}$ constant and $\bar{\mathbf{q}} = 0$, we get an Airy wave solver indistinguishable from [Tessendorf 2014].

## 5.2 Limitations and future work

To create realistic wake patterns, we take care that our method exhibits the correct wave dispersion. Our surface waves obey the exact dispersion relation of gravity waves, as discussed in Section 4.3.1 and illustrated in Figure 6. However, because our bulk flow simulator [Stelling and Duinmeijer 2003] uses a numerical discretization of spatial derivatives, it does show some unrealistic numerical dispersion in the presence of very steep waves (like in dam break scenarios).

We note that our model — and most other practical methods used in computer graphics — is based on theories that assume small wave amplitudes, so we cannot expect it to behave accurately in the presence of steep waves. Also, the choice of whether a given

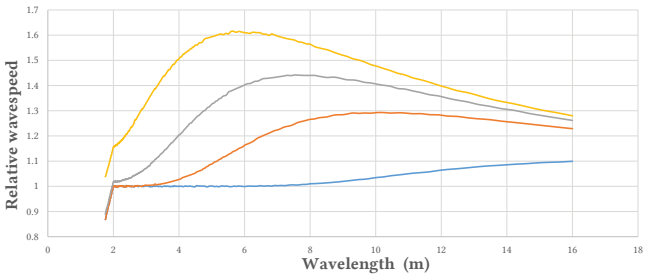


Fig. 9. *Effect of wave decomposition on wave speed.* We plot the measured wavespeed $\frac{\omega}{k}$ relative to the theoretical surface wave speed for simulations with varying diffusion times in our wave decomposition at a water depth of 4m. The yellow, gray, orange, and blue curves correspond to 8, 16, 32, 128 diffusion iterations, respectively.


Fig. 10. *Flooding into detailed, dynamically changing terrain.*

wave is considered "bulk flow" or "surface wave" depends on our decomposition algorithm.

Our model derives its efficiency by discarding the potentially complicated wave behaviors associated with the non-linear $\nabla \cdot (\bar{h}\tilde{\mathbf{u}})$ term in Equation 10. Neglecting this term implies that velocity perturbations caused by surface waves $\tilde{\mathbf{u}}$ do not reach all the way down to the sea floor — coupled interactions between the water surface and sea floor are associated with vortex shedding and wave breaking in nature. Linear theory suffices for interference patterns (Kelvin derived the wake pattern using linear theory), but is insufficient for these rotational effects.

Our surface wave solver operates in frequency space with an exponential integrator that assumes waves do not intersect boundaries over the course of a single time step. Consequently, depending on the time step and wave speed, waves may travel through thin boundaries. We could use the shadow kernels of Canabal et al. [2016] to prevent waves passing through boundaries in the future.

The decomposition in Section 4.1 uses an easy-to-implement explicit integration of a spatially inhomogeneous diffusion equation as a proof-of-concept. While it is already effective, this algorithm could be made more precise and efficient in the future, perhaps by using implicit integration, a spectral method, or any number of techniques from the digital signal processing literature. We look forward to refining this method in the future.

Our method assumes that the bulk and surface waves consist of low- and high-frequency height fields, respectively. This implies, because the bulk flow $\bar{h}$ interacts with the ocean floor (and because the surface waves $\tilde{h}$ do not), that the ocean floor is also a low-frequency height field. To make our method work well with high-frequency ocean floors, we would have to take the ocean floor into account in the decomposition. Our current implementation does not do this, so extra work is needed to exhibit correct behavior with high-frequency terrain.

Equation 9 adds gravity as an external force. Although real water also experiences surface tension and viscosity forces, we did not include them in our implementation. Surface tension waves should be straightforward to add if we modify the dispersion relationship, though it might add an additional time step restriction on the bulk flow solver due to the high speed of capillary waves. Many explicit solvers add artificial damping to ensure stability, so we intentionally left it out to demonstrate robustness. In reality, viscosity damps shorter wavelengths faster than longer ones — a property that is currently missing from our simulator apart from numerical viscosity caused by Semi-Lagrangian transport of surface waves. In the future, we could add damping by approximating viscosity.

Our method also currently assumes that gravity $g$ is aligned with our height field direction. More work would be needed to simulate strong tangential forces or water droplets on vertical surfaces, perhaps by building upon the work of Wang et al. [2007].

Our shallow water solver can produce flows with significant vorticity, but our surface wave solver obeys linear wave theory, which assumes that $\tilde{\mathbf{q}}$ is curl-free. Our decomposition in Section 4.1 does not make this distinction, and transfers high-frequency vorticity to the surface wave solver, where it is ignored. If desired, one could try to prevent the transfer of this high-frequency vorticity or add a vorticity solver to the surface wave simulation.

Finally, our algorithm for reconstructing the water surface should be improved. Our bi-cubic surface reconstruction requires a two-cell-thick extrapolation into the nearby solid boundaries, and we apply a threshold on the water height to classify whether a surface is wet or dry. This combination of extrapolation and thresholding can cause the water surface to unrealistically flicker near steep walls that are close to the wet/dry threshold.

## 6 CONCLUSION

We have presented a height field water simulation algorithm that combines the benefits of shallow water simulation with those of surface wave simulation. Our results produce useful effects like flooding on variable terrain heights, wake patterns and ripples, and combinations of both surface waves and vortices in the same animation. We introduced a novel wave decomposition heuristic and a principled approach to simulating each of the terms in the decomposed fluid equations. Our method preserves surface wave speed and volume exactly, and it is well suited for parallel GPU implementation.

## ACKNOWLEDGMENTS

We thank Georg Sperl for helping with early research for this paper, Mickael Ly and Yi-Lu Chen for proofreading, and members of the ISTA Visual Computing Group for general feedback. This project was funded in part by the European Research Council (ERC Consolidator Grant 101045083 *CoDiNA*).

The motorboat and sailboat were modeled by Sergei and the palmtrees by YadroGames. The environment map was created by Emil Persson.

## REFERENCES

George Biddell Airy. 1841. Tides and waves. *Encyclopedia Metropolitana, Mixed Sciences* 3 (1841).

Omri Azencot, Orestis Vantzos, and Mirela Ben-Chen. 2018. An explicit structure-preserving numerical scheme for EPDiff. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 107–119.

Fa Biesel. 1952. Study of wave propagation in water of gradually varying depth. *Gravity waves* (1952), 243–253.

Joseph Boussinesq. 1872. Théorie des ondes et des remous qui se propagent le long d'un canal rectangulaire horizontal, en communiquant au liquide contenu dans ce canal des vitesses sensiblement pareilles de la surface au fond. *Journal de mathématiques pures et appliquées* (1872), 55–108.

José A. Canabal, David Miraut, Nils Thuerey, Theodore Kim, Javier Portilla, and Miguel A. Otaduy. 2016. Dispersion Kernels for Water Wave Simulation. *ACM Trans. Graph.* 35, 6, Article 202 (2016), 10 pages.

Nuttapong Chentanez and Matthias Müller. 2010. Real-Time Simulation of Large Bodies of Water with Small Scale Details. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 197–206.

Nuttapong Chentanez, Matthias Müller, and Tae-Yong Kim. 2015. Coupling 3D Eulerian, Heightfield and Particle Methods for Interactive Simulation of Large Scale Liquid Phenomena. *IEEE Transactions on Visualization and Computer Graphics* 21, 10 (2015), 1116–1128.

Fang Da, David Hahn, Christopher Batty, Chris Wojtan, and Eitan Grinspun. 2016. Surface-only liquids. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–12.

B De St Venant. 1871. Theorie du mouvement non-permanent des eaux avec application aux crues des rivers et a l'introduntion des Marees dans leur lit. *Academic de Sci. Comptes Redus* 73, 99 (1871), 148–154.

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 15–22.

Alexey V Fedorov and W Kendall Melville. 1998. Nonlinear gravity–capillary waves with forcing and dissipation. *Journal of Fluid Mechanics* 354 (1998), 1–42.

Alain Fournier and William T Reeves. 1986. A simple model of ocean waves. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 75–84.

Franz Gerstner. 1809. Theorie der Wellen. *Annalen der Physik* 32, 8 (1809), 412–445.

Trond Runar Hagen, Jon M Hjelmervik, K-A Lie, Jostein R Natvig, and M Ofstad Henriksen. 2005. Visual simulation of shallow-water waves. *Simulation Modelling Practice and Theory* 13, 8 (2005), 716–726.

Richard Wesley Hamming. 1998. *Digital filters*. Courier Corporation.

Damien Hinsinger, Fabrice Neyret, and Marie-Paule Cani. 2002. Interactive animation of ocean waves. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 161–166.

Libo Huang, Ziyin Qu, Xun Tan, Xinxin Zhang, Dominik L Michels, and Chenfanfu Jiang. 2021. Ships, splashes, and waves on a vast ocean. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–15.

S. Jeschke, C. Hafner, N. Chentanez, M. Macklin, M. Müller-Fischer, and C. Wojtan. 2020. Making Procedural Water Waves Boundary-Aware. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, Article 5, 8 pages.

Stefan Jeschke, Tomáš Skřivan, Matthias Müller-Fischer, Nuttapong Chentanez, Miles Macklin, and Chris Wojtan. 2018. Water Surface Wavelets. *ACM Trans. Graph.* 37, 4, Article 94 (2018), 13 pages.

Stefan Jeschke and Chris Wojtan. 2015. Water Wave Animation via Wavefront Parameter Interpolation. *ACM Trans. Graph.* 34, 3, Article 27 (2015), 14 pages.

Stefan Jeschke and Chris Wojtan. 2017. Water Wave Packets. *ACM Trans. Graph.* 36, 4, Article 103 (2017), 12 pages.

Robin Stanley Johnson. 1997. *A modern introduction to the mathematical theory of water waves*. Number 19. Cambridge university press.

Steven G Johnson. 2011. Notes on FFT-based differentiation. *MIT Applied Mathematics, Tech. Rep.* (2011).

Michael Kass and Gavin Miller. 1990. Rapid, stable fluid dynamics for computer graphics. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*. 49–57.

Todd Keeler and Robert Bridson. 2014. Ocean waves animation using boundary integral equations and explicit mesh tracking. In *ACM SIGGRAPH 2014 Posters*. 1–1.

Theodore Kim, Jerry Tessendorf, and Nils Thuerey. 2013. Closest point turbulence for liquid surfaces. *ACM Transactions on Graphics (TOG)* 32, 2 (2013), 1–13.

Anita T Layton and Michiel van de Panne. 2002. A numerically efficient and stable algorithm for animating water waves. *The Visual Computer* 18, 1 (2002), 41–53.

Jörn Loviscach. 2002. A Convolution-Based Algorithm for Animated Water Waves. In *Eurographics short papers*.

Jörn Loviscach. 2003. Complex Water Effects at Interactive Frame Rates. *Journal of WSCG* 11 (2003), 2003.

John Marshall, Kerry Emanuel, and Alistair Adcroft. 2004. Atmospheric And Oceanic Modeling Course 12.950, Lecture 2. *Massachusetts Institute of Technology: MIT OpenCouseWare, https://ocw.mit.edu/. License: Creative Commons BY-NC-SA*. (Spring 2004).

Gary A Mastin, Peter A Watterberg, and John F Mareda. 1987. Fourier synthesis of ocean scenes. *IEEE Computer Graphics and Applications* 7, 3 (1987), 16–23.

Zherong Pan, Jin Huang, Yiying Tong, and Hujun Bao. 2012. Wake synthesis for shallow water equation. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 2029–2036.

Darwyn R Peachey. 1986. Modeling waves and surf. *ACM Siggraph Computer Graphics* 20, 4 (1986), 65–74.

William John Macquorn Rankine. 1863. VI. On the exact form of waves near the surface of deep water. *Philosophical transactions of the Royal society of London* 153 (1863), 127–138.

Camille Schreck, Christian Hafner, and Chris Wojtan. 2019. Fundamental solutions for water wave animation. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–14.

Camille Schreck and Chris Wojtan. 2022. Coupling 3D Liquid Simulation with 2D Wave Propagation for Large Scale Water Surface Animation Using the Equivalent Sources Method. In *Computer Graphics Forum*, Vol. 41. 343–353.

Tomas Skrivan, Andreas Soderstrom, John Johansson, Christoph Sprenger, Ken Museth, and Chris Wojtan. 2020. Wave Curves: Simulating Lagrangian Water Waves on Dynamically Deforming Surfaces. *ACM Trans. Graph.* 39, 4, Article 65 (2020), 12 pages.

G. S. Stelling and S. P. A. Duinmeijer. 2003. A staggered conservative scheme for every Froude number in rapidly varied shallow water flows. *International Journal for Numerical Methods in Fluids* 43, 12 (2003), 1329–1354.

Jerry Tessendorf. 2004a. Interactive water surfaces. *Game Programming Gems* 4 (2004), 265–274.

Jerry Tessendorf. 2004b. Simulating Ocean Water. In *ACM SIGGRAPH 2004 course notes*.

Jerry Tessendorf. 2014. *eWave: Using an Exponential Solver on the iWave Problem.* . Technical Note.

Jerry Tessendorf. 2017. *Gilligan: A Prototype Framework for Simulating and Rendering Maritime Environments.* . Technical Note.

William Thomson. 1887. On ship waves. *Proceedings of the institution of mechanical engineers* 38, 1 (1887), 409–434.

Nils Thürey, Ulrich Rüde, and Marc Stamminger. 2006. Animation of open water phenomena with coupled shallow water and free surface simulations. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 157–164.

Huamin Wang, Gavin Miller, and Greg Turk. 2007. Solving General Shallow Wave Equations on Surfaces. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 229–238.

Qizhi Yu, Fabrice Neyret, and Anthony Steed. 2011. Feature-based vector simulation of water waves. *Computer Animation and Virtual Worlds* 22, 2-3 (2011), 91–98.

## A SWE INTEGRATION

Here we list the integration rules from Section 5 of [Stelling and Duinmeijer 2003]. Note that we only implement their momentum-conserving time integration, but *not* the energy-head conservation algorithm. The wave heights are updated with

$$\frac{d\bar{h}_{i,j}}{dt} + \frac{\bar{h}_{i+1/2,j}\bar{u}^{\rightarrow}_{i+1/2,j} - \bar{h}_{i-1/2,j}\bar{u}^{\rightarrow}_{i-1/2,j}}{\Delta x} + \frac{\bar{h}_{i,j+1/2}\bar{u}^{\uparrow}_{i,j+1/2} - \bar{h}_{i,j-1/2}\bar{u}^{\uparrow}_{i,j-1/2}}{\Delta x} = 0 \tag{18}$$

where the time derivative $d/dt$ uses first-order finite differences, the arrows $\rightarrow$ and $\uparrow$ indicate the $x$- or $y$- component of velocity, and

$h$ values with non-integer indices are evaluated using first order up-winding. The velocity is updated with upwinding as well. For positive flow directions $\bar{u}^{\rightarrow}$ and $\bar{u}^{\uparrow}$, the update equations are:

$$\frac{d\bar{u}^{\rightarrow}_{i+1/2,j}}{dt} + \frac{\bar{q}^{\rightarrow}_{i,j}}{\bar{h}_{i+1/2,j}} \frac{\bar{u}^{\rightarrow}_{i+1/2,j} - \bar{u}^{\rightarrow}_{i-1/2,j}}{\Delta x} + \frac{\bar{q}^{\uparrow}_{i,j-1/2}}{\bar{h}_{i+1/2,j}} \frac{\bar{u}^{\rightarrow}_{i+1/2,j} - \bar{u}^{\rightarrow}_{i+1/2,j-1}}{\Delta x} + g\frac{\bar{h}_{i+1,j} - \bar{h}_{i,j}}{\Delta x} = 0 \tag{19}$$

$$\frac{d\bar{u}^{\uparrow}_{i,j+1/2}}{dt} + \frac{\bar{q}^{\rightarrow}_{i-1/2,j}}{\bar{h}_{i,j+1/2}} \frac{\bar{u}^{\uparrow}_{i,j+1/2} - \bar{u}^{\uparrow}_{i-1,j+1/2}}{\Delta x} + \frac{\bar{q}^{\uparrow}_{i,j}}{\bar{h}_{i,j+1/2}} \frac{\bar{u}^{\uparrow}_{i,j+1/2} - \bar{u}^{\uparrow}_{i,j-1/2}}{\Delta x} + g\frac{\bar{h}_{i,j+1} - \bar{h}_{i,j}}{\Delta x} = 0. \tag{20}$$

Again, arrows indicate the $x$- or $y$- component of the flow rate and velocity. Integer indices of $q$ (which are sampled on a staggered grid) indicate averaged values, e.g, $q^{\rightarrow}_{i,j} = (q^{\rightarrow}_{i-1/2,j} + q^{\rightarrow}_{i+1/2,j})/2$. Please see [Stelling and Duinmeijer 2003] for more thorough implementation details.

## B CORRECTING NUMERICAL DISPERSION

Following [Marshall et al. 2004], we analyze the effect of our numerical spatial derivative operator on the shallow water wave equation:

$$\begin{aligned} \partial_t q &= -g\, \partial_x h \\ \partial_t h &= -H\, \partial_x q \end{aligned} \tag{21}$$

where $g$ is gravity, $H$ is the nominal depth of the water, $q$ is the flow and $h$ the free-surface displacement. $\partial_t$ and $\partial_x$ are partial derivatives with respect to time and space, respectively. The theoretical dispersion relationship of this system is $\omega = \sqrt{gH}$.

Our implementation uses a spectral derivative (in Fourier space) to compute the first derivative, and a finite volume approximation on a staggered grid $\partial_x \approx \frac{1}{\Delta x}\partial_i$ when reconstructing the wave heights from the divergence of flow rates in the second:

$$\begin{aligned} \partial_t q &= -g\, \partial_x h \\ \partial_t h &= -\frac{H}{\Delta x}\partial_i q. \end{aligned} \tag{22}$$

The operator matrix is

$$\begin{bmatrix} \partial_t & g\partial_x \\ \frac{H}{\Delta x}\partial_i & \partial_t \end{bmatrix} \begin{bmatrix} q \\ h \end{bmatrix} = 0. \tag{23}$$

To discover the effective dispersion of this operation, we plug in the wave solution $ae^{i(kx-\omega t)}$ and compute the derivatives:

$$\begin{bmatrix} -i\omega & gik \\ \frac{2iH}{\Delta x}\sin\left(\frac{k\Delta x}{2}\right) & -i\omega \end{bmatrix} \begin{bmatrix} q \\ h \end{bmatrix} = 0. \tag{24}$$

The determinant of this matrix gives us the numerical dispersion relation

$$\omega^2 - gH\frac{2k}{\Delta x}\sin\left(\frac{k\Delta x}{2}\right) = 0 \tag{25}$$

or

$$\omega = \sqrt{gH}\sqrt{\frac{2k}{\Delta x}}\sqrt{\sin\left(\frac{k\Delta x}{2}\right)} = 0. \tag{26}$$

Thus, the numerical dispersion of our scheme is equal to the original dispersion $\sqrt{gH}$ multiplied by a factor of

$$\beta = \sqrt{\frac{2k}{\Delta x}}\sqrt{\sin\left(\frac{k\Delta x}{2}\right)}. \tag{27}$$

## C DERIVATION OF DIFFUSION COEFFICIENT $\alpha$

The diffusion equation (Equation 11) acts as a Gaussian filter with a cutoff wavelength $\lambda_{\text{cutoff}} = 2\pi h$. Rounding the filter's half-width at half-maximum to 1 for simplicity, this gives us a heat kernel with standard deviation equal to $\sigma = \frac{\lambda_{\text{cutoff}}}{2\pi} = \frac{2\pi h}{2\pi} = h$ and thus $\alpha = \frac{\sigma^2}{2t} = \frac{h^2}{2t}$. Fixing the diffusion timestep to the maximum stable size $\Delta T = 0.25$ and the number of diffusion iterations $n$ to a tolerable 128 iterations gives us a diffusion coefficient equal to $\frac{h^2}{2n\Delta T} = \frac{h^2}{64}$.