# A Clebsch Method for Free-Surface Vortical Flow Simulation

SHIYING XIONG, Dartmouth College
ZHECHENG WANG, Dartmouth College
MENGDI WANG, Dartmouth College
BO ZHU, Dartmouth College

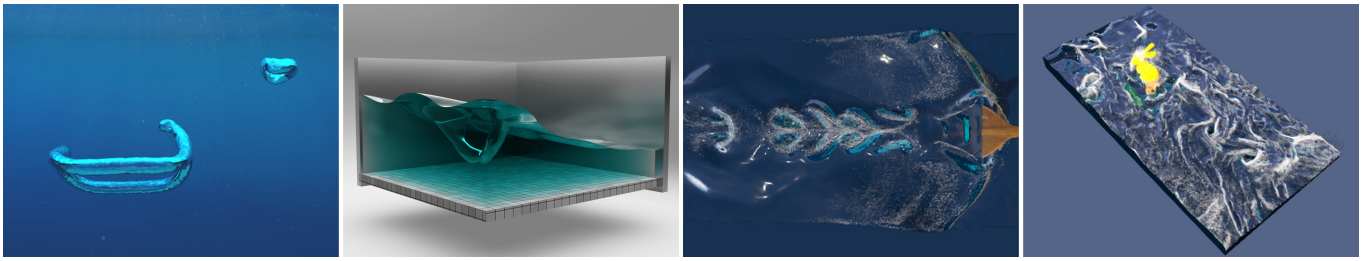

Fig. 1. Various free-surface vortical flow phenomena simulated using our Clebsch method (left to right): reconnection of vortex bubble rings, horseshoe vortex with free ends attached to the water surface, wake vortex formation in a paddling propulsion, turbulent free-surface vortices behind a bunny in tidal wave.

We propose a novel Clebsch method to simulate the free-surface vortical flow. At the center of our approach lies a level-set method enhanced by a wave-function correction scheme and a wave-function extrapolation algorithm to tackle the Clebsch method's numerical instabilities near a dynamic interface. By combining the Clebsch wave function's expressiveness in representing vortical structures and the level-set function's ability on tracking interfacial dynamics, we can model complex vortex-interface interaction problems that exhibit rich free-surface flow details on a Cartesian grid. We showcase the efficacy of our approach by simulating a wide range of new free-surface flow phenomena that were impractical for previous methods, including horseshoe vortex, sink vortex, bubble rings, and free-surface wake vortices.

CCS Concepts: • **Computing methodologies** → **Modeling and simulation**.

Additional Key Words and Phrases: Free-Surface Fluid, Level Set, Clebsch Representation, Bubble Ring, Horseshoe Vortex

**ACM Reference Format:**
Shiying Xiong, Zhecheng Wang, Mengdi Wang, and Bo Zhu. 2022. A Clebsch Method for Free-Surface Vortical Flow Simulation. *ACM Trans. Graph.* 41, 4, Article 116 (July 2022), 13 pages. https://doi.org/10.1145/3528223.3530150

Authors' addresses: Shiying Xiong, Computer Science Department, Dartmouth College, Hanover, NH, USA, shiying.xiong@dartmouth.edu; Zhecheng Wang, Computer Science Department, Dartmouth College, Hanover, NH, USA (Z. Wang was a research intern at Dartmouth College during this work), zhechengwang@protonmail.com; Mengdi Wang, Computer Science Department, Dartmouth College, Hanover, NH, USA, mengdi.wang.gr@dartmouth.edu; Bo Zhu, Computer Science Department, Dartmouth College, Hanover, NH, USA, bo.zhu@dartmouth.edu.


0730-0301/2022/7-ART116 $15.00
https://doi.org/10.1145/3528223.3530150

## 1 INTRODUCTION

Free-surface vortices are ubiquitous on different scales. Examples range from small-scale vortices such as bubble rings [White 1996] and horseshoe vortices behind a paddle [Mochizuki et al. 2008] to large-scale swirls and eddies on a turbulent ocean surface [Tings and Velotto 2018]. From the perspective of vortex dynamics, all these flow phenomena manifest interactions between the vortical structures in fluid bulks and the dynamic features on a fluid interface. The vortex tubes characterizing these vortical structures may intersect with the free surface by one end (e.g., sink swirls), two ends (e.g., horseshoe vortices), or form a closed ring surrounded by spinning air (e.g., toroidal bubbles). Developing numerical simulation algorithms to model these free-surface vortical flow phenomena will not only enhance the visual effects of the current fluid animation literature but also provide new possibilities to help scientists comprehend the fundamental physics underpinning these phenomena.

The numerical simulation of these free-surface vortical flows using a traditional approach by evolving the velocity [Osher and Fedkiw 2001] or vorticity [Coquerelle and Cottet 2008] with the fluid's level-set interface for example is challenging due to the inherent difficulties in characterizing the temporally coherent vortical evolution. Most of the existing free-surface flow solvers in computer graphics, be they purely Eulerian [Fedkiw and Osher 2002], Lagrangian [Müller et al. 2003], or particle-grid hybrid [Losasso et al. 2008; Zhu and Bridson 2005], are focused on capturing the small-scale splashing details such as spray, foam, and small bubbles near the fluid surface, while the studies on the generation and preservation of the highly structured free-surface vortices and their various visually-appealing interactions with the entrained air, moving solid, or other environmental objects are scarce.

The Clebsch method [Chern et al. 2016; Yang et al. 2021] has recently emerged as a new category of fluid simulation methods that have demonstrated their particular strength in preserving coherent vortical structures. Researchers have shown the method's

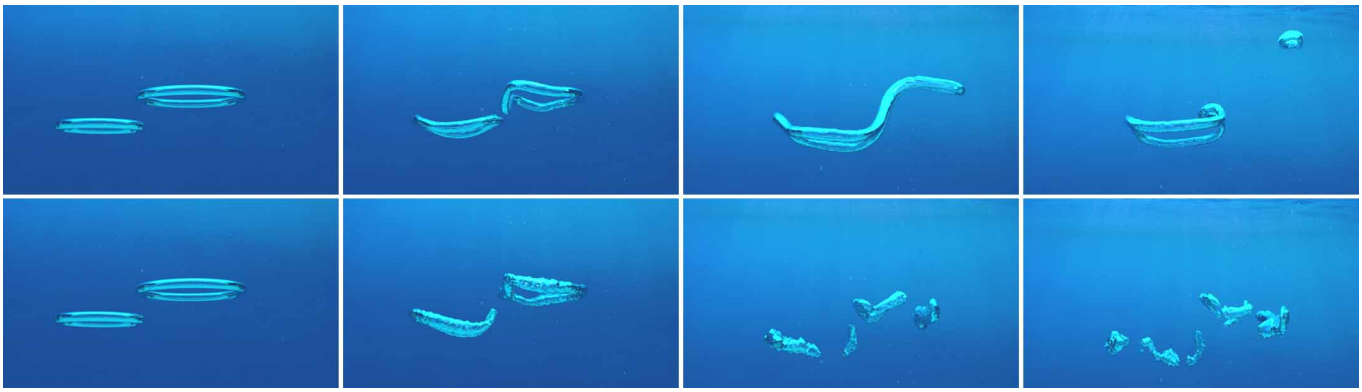

Fig. 2. Comparison of bubble ring evolution simulated with the free-surface Clebsch solver (top) and a velocity-based solver (bottom), which shows two bubble rings moving, deforming, connecting, and splitting, from left to right are frames 1, 120, 320, 400 of the two simulations.

effectiveness by simulating a broad range of vortical flow phenomena that were difficult or impractical for a traditional velocity-based or vorticity-based approach. The key idea of the Clebsch-based methods is to evolve a wave-function field and use it to reconstruct the velocity in each time step, rather than directly evolving the velocity. By leveraging the wave function's inherent expressiveness of vortical structures, the simulations can preserve geometrically clear and temporally coherent vortical structures in a turbulent flow.

Despite the success of Clebsch methods in modeling smoke flows, simulating highly turbulent free-surface flows remains a recalcitrant challenge due to the incompatibility issue between their wave function and velocity field near a fluid interface. A smooth wave-function field is needed to retrieve the flow velocity. However, complex interface evolution, such as near-surface turbulence, will break such smoothness due to various vortex interactions. The computational challenges manifest in the following three settings: (1) a turbulent wavy surface; (2) a free surface near the solid-air-liquid intersection; and (3) a free surface with drastic topological changes (e.g., a bulk of water merges into another, or interface self-collision). Yang et al. [2021] proposed a gauge Clebsch method to simulate free-surface fluid whose motion is dominated by surface tension. In this setting, most vortical structures remain underwater, and no strong vortices with topological evolution take place near the interface.

We propose a practical Clebsch method to simulate free-surface vortical flow. Our main technical contribution is the adaptation of the Clebsch representation into a standard liquid simulation pipeline to support a multitude of new free-surface vortical phenomena. To achieve this goal, we devised a novel wave-function correction scheme and a wave-function extrapolation algorithm to address the numerical instabilities near a free surface. Our free-surface Clebsch framework inherits most of its numerical components from the level-set-based water simulation, which allows its accessibility to the many parallelizable data structures for large-scale simulation. Our method provides an efficient and unified way to simulate a broad range of flow phenomena that exhibit complex vortical structural evolution, such as bubble rings, horseshoe vortices, sink swirls, and free-surface wake vortices, which were all challenging to solve using previous approaches.

We summarize the main contributions of our approach as:

- A novel wave-function correction scheme to address the inconsistency problem between the wave-function and the velocity field near a dynamic interface;
- A novel wave-function extrapolation scheme to enable consistent wave-function and velocity representation in the level-set narrow band;
- A unified Clebsch solver to support free-surface vortical flow simulation, enabling the simulation of many novel flow phenomena on a Cartesian grid.

## 2 RELATED WORKS

*Clebsch Map in Mathematics and Physics.* Clebsch [1859] proposed a representation of flow field, where the velocity–vorticity field is represented by a series of scalar functions called Clebsch potentials. Clebsch map has three advantages in featuring the flow's vorticity field: (1) Clebsch map is an Eulerian version of Hamilton's principle [Bateman 1929; Eckart 1938; Salmon 1988], motivating to design symplectic algorithms for fluid simulation [McLachlan et al. 2019]. (2) The vorticity Clebsch potentials are Lagrangian scalar fields in inviscid flow, which means the potentials can be treated as a Lagrangian coordinate that follows the flow to analyze fluid dynamics [Wu et al. 2015]. (3) The vorticity Clebsch potentials are the vortex surface fields [Hao et al. 2019; Yang and Pullin 2010, 2011]. Namely, the vorticity lines are tangent to the isosurface of vorticity Clebsch potentials, which facilitates the extraction of temporal coherent vortex structures such as vortex tubes and vortex sheets [Chern et al. 2017; He and Yang 2016]. Based on the Clebsch map, conservation laws for vorticity and related quantities, e.g. Kelvin's circulation theorem [Thomson 1869], Helmholtz's vorticity theorem [Helmholtz 1858], Ertel's theorem for potential vorticity [Ertel 1942], and the conservation of helicity [Moffatt 1969; Moreau 1961] become self-evident. Additionally, the Clebsch map has broad connections to topics in modern mathematics such as fiber bundle theory, gauge theory, and Lie theory. Chern [2017] covered an elaborated mathematical and physical statement for the Clebsch representation.

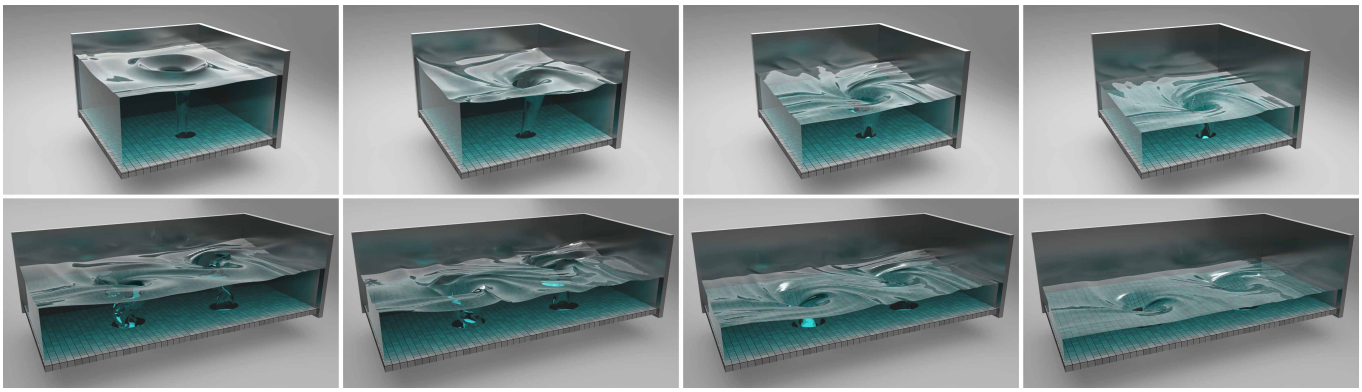

Fig. 3. Sink vortex with one (top) or two (bottom) holes. The top figures with one sink hole from left to right are frames 50, 150, 300, and 450, respectively. The bottom figures with two holes from left to right are frames 50, 100, 200, and 300, respectively.

*Clebsch Methods for Fluid Simulation.* Although the Clebsch map has multiple advantages in the theoretical study of fluid dynamics, the study of applying it to solve numerical simulations of complex flow phenomena is scarce in both computer graphics and computational physics. The main reasons are as follows: (1) The Clebsch map can only represent fields with vanishing helicity, meaning the Clebsch map can only be used to express flows with simple topology structures [Xiong and Yang 2020]; (2) Some portions of the numerical Clebsch potentials tend to evolve into nearly singular structures owing to the persistent straining motion driven by the chaotic flow field in fully developed turbulence during advection [Xiong and Yang 2019b]. Chern et al. [2016] first successfully applied Clebsch fluid to smoke simulation in graphics by introducing a spherical Clebsch map [Kuznetsov and Mikhailov 1980], a generalization of Clebsch map, and solving Schrödinger's equation with fluid properties [Madelung 1926, 1927; Pismen and Pismen 1999; Sorokin 2001]. The success of Chern et al. [2016] is attributed to two aspects: first, the original Clebsch map's inability to express a non-zero helicity flow field is solved by the spherical Clebsch Map with quantized helicity. Secondly, the incompressible Schrödinger flow only needs to solve the Poisson operator of the complex variable, thus avoiding the numerical difficulty in the advecting Lagrangian scalar field. Furthermore, combining with gauge fluid [Saye 2016], Yang et al. [2021] successfully extended the Clebsch-based simulator to handle surface-tension flow. However, current approaches still struggle to handle complex interface dynamics due to the phase discontinuity of the Clebsch function near the interface.

*Free-Surface Flow Simulation.* Free-surface flow simulation has been receiving recurring attention in computer graphics over the past decades for its visually appealing interface dynamics and various small-scale flow details. The level-set method [Fedkiw and Osher 2002; Gibou et al. 2018] along with its many-particle augmentations [Enright et al. 2002; Losasso et al. 2008], data-structure modifications [Houston et al. 2006; Losasso et al. 2006, 2004], and parallel implementations [Hughes et al. 2007; Mazhar et al. 2013], has been one of the most successful practices for high-resolution water animation. The swirling flow details on the water surface mainly stem from vortical enhancement forces [Fedkiw et al. 2001; Rasmussen et al. 2003], vortex particles [Selle et al. 2005], or less-dissipative advection schemes [MacCormack 2003] that are applied on the grid solver. On another front, particle-based approaches [Müller et al. 2003] and hybrid Eulerian-Lagrangian approaches [Jiang et al. 2015; Zhu and Bridson 2005] have shown their efficacy in simulating highly dynamic, large-scale water scenarios. The adaptive nature of particles, in conjunction with other enhancing numerical models [Huang and Michels 2020], enable impressively fine-scale details near the water surface that can capture the broad spectrum of flow features including spray, foam, bubble, and small-scale vortices. In particular, the combination of the level-set and advected spray particles [Chentanez and Müller 2011; Losasso et al. 2008] enabled some of the largest-scale simulations for turbulent ocean water. In comparison to the vast previous efforts on data-structure-level innovation, the study of enhancing the flow details by exploring novel vorticity representations is more focused on simulating incompressible flow without an interface (e.g., see [Chern et al. 2016; Xiong et al. 2021; Yang et al. 2021; Zhang and Bridson 2014]). Simultaneously adapting these approaches to a free-surface setting and leveraging their vorticity expressiveness to generate and preserve vortical structures near the free surface has remained an unexplored problem due to the complexities of tackling the various vortex-interface interactions.

## 3 PHYSICAL MODEL

### 3.1 Clebsch Wave Functions

For an incompressible fluid in the domain $\Omega_L$, the spherical Clebsch map [Chern et al. 2017, 2016; Kuznetsov and Mikhailov 1980] expresses a velocity field $\boldsymbol{u}(\boldsymbol{x}, t)$ as a two-component wave function $\boldsymbol{\Psi} = [\Psi_1, \Psi_2]^T$ with the normalization condition

$$\|\boldsymbol{\Psi}\|^2 = \langle \boldsymbol{\Psi}, \boldsymbol{\Psi} \rangle_{\mathbb{R}} = 1 \tag{1}$$

and the solenoidal condition

$$\langle \mathrm{i}\boldsymbol{\Psi}, \Delta\boldsymbol{\Psi} \rangle_{\mathbb{R}} = 0. \tag{2}$$

Here $\Psi_i = a_i + \mathrm{i}b_i,\ i = 1, 2$ are two complex-valued functions and the norm is defined as $\langle \boldsymbol{\Phi}, \boldsymbol{\Psi} \rangle_{\mathbb{R}} = \mathrm{Re}(\overline{\Phi}_1\Psi_1 + \overline{\Phi}_2\Psi_2)$. The mapping from a wave field $\boldsymbol{\Psi}$ to a velocity field $\boldsymbol{u}$ in $\Omega_L$ can be expressed as:

$$\boldsymbol{u} = \hbar \langle \boldsymbol{\nabla}\boldsymbol{\Psi}, \mathrm{i}\boldsymbol{\Psi} \rangle_{\mathbb{R}}, \tag{3}$$

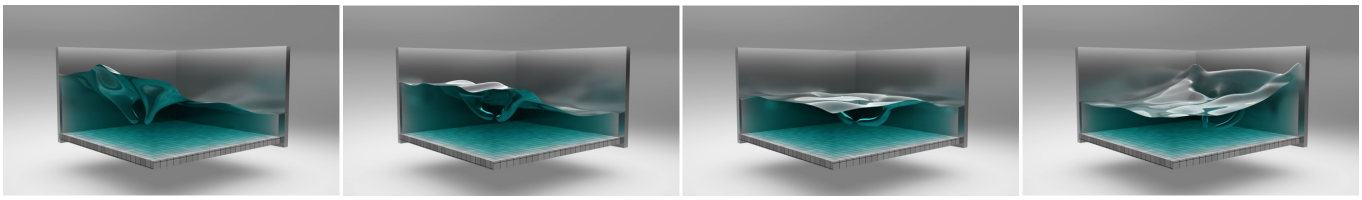

Fig. 4. The evolution of a horseshoe vortex. The figures from left to right are frames 150, 300, 450, and 600, respectively.

where $\hbar$ is a tunable parameter specifying the quantization of vorticity field $\boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{u}$.

The wave function can be transformed into vorticity Clebsch functions by Hopf [1931] fiberation: $s_1 = a_1^2 + b_1^2 - a_2^2 - b_2^2$, $s_2 = 2(b_1a_2 - a_1b_2)$, $s_3 = 2(a_1a_2 + b_1b_2)$. Same as the original Clebsch maps [Clebsch 1859], the vorticity Clebsch maps $\boldsymbol{s} = (s_1, s_2, s_3)$ are exact vortex surface fields owing to $\boldsymbol{\omega} \cdot \boldsymbol{\nabla} s_p = 0,\ p = 1, 2, 3$. We remark that this property contains important geometric information about the flow fields, which provides an appealing perspective for flow visualization [Chern et al. 2017; He and Yang 2016], analysis [Hao et al. 2019; Xiong and Yang 2017, 2019a,b; Yang and Pullin 2010], and simulation [Chern et al. 2016; Yang et al. 2021]. For demonstration, we describe an analytical method for constructing a wave function for knotted and linked vortex tubes in Appendix A.

### 3.2 Free-Surface Clebsch Fluid

For a free-surface flow in the domain $\Omega = \Omega_L \cup \Omega_A \cup \Gamma$, where $\Gamma$ is the free surface with zero thickness that separates the fluid into liquid in $\Omega_L$ and air with constant pressure $p_{\text{air}}$ in $\Omega_A$. We track the fluid interface with a level-set function $\phi$ [Fedkiw and Osher 2002; Osher and Sethian 1988]. In the case of no phase change, mass conservation implies that the interface advances at the speed of the flow $\boldsymbol{u} \cdot \boldsymbol{n}_\Gamma$, where $\boldsymbol{n}_\Gamma$ is the unit normal of $\Gamma$.

The inviscid incompressible flows with gravity, surface tension, and boundary conditions are governed by

$$\begin{aligned} &\begin{cases} \dfrac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = -\boldsymbol{\nabla} q, \\ \boldsymbol{\nabla} \cdot \boldsymbol{u} = 0, \end{cases} \quad \boldsymbol{x} \in \Omega_L, \\ &\boldsymbol{u} \cdot \boldsymbol{n} = \boldsymbol{u}_\partial \cdot \boldsymbol{n},\ \boldsymbol{x} \in \partial\Omega_b, \\ &q = p_{\text{air}} + \gamma\kappa - G,\ \boldsymbol{x} \in \Gamma. \end{aligned} \tag{4}$$

Here, $\mathrm{D}/\mathrm{D}t = \partial/\partial t + \boldsymbol{u} \cdot \boldsymbol{\nabla}$ is the material derivative, $q = p - G$ is an auxiliary gauge variable with the pressure $p$, and gravitational potential $G = G(\boldsymbol{x}) = \boldsymbol{g} \cdot \boldsymbol{x}$, $\boldsymbol{u}_\partial$ is the velocity of the solid boundary $\partial\Omega_b$, $\gamma$ is the surface tension coefficient on the interface $\Gamma$, and $\kappa$ is the interface curvature. The surface tension on the interface is modeled as the pressure jump due to the local curvature and the gravity force is caused by the density jump on the interface.

The momentum conservation in (4) can be transformed to

$$\frac{\mathrm{D}\Psi}{\mathrm{D}t} = \frac{-\mathrm{i}}{\hbar}\left(q - \frac{|\boldsymbol{u}|^2}{2}\right)\Psi, \tag{5}$$

with the normalization (1) and the solenoidal constraint (2). We refer readers to Appendix B for a more detailed deduction of (5).

**Algorithm 1** Time integration scheme.

**Input:** $\Psi^{(j)}$, $\boldsymbol{u}^{(j)}$, $\phi^{(j)}$, $\hbar$, $\Delta x$, $\Delta t$, $\beta$, $p_{\text{air}}$, $\gamma$, $G$, $l_{nb}$
**Output:** $\Psi^{(j+1)}$, $\boldsymbol{u}^{(j+1)}$, $\phi^{(j+1)}$

$\Psi^\star$, $\boldsymbol{u}^\sharp$, $\phi^{(j+1)}$ ←Advection ($\Psi^{(j)}$, $\boldsymbol{u}^{(j)}$, $\phi^{(j)}$, $\hbar$, $\beta$,$\Delta t$,$\Delta x$)
▷ Alg. 2
$\Psi^\sharp$ ←Correction ($\Psi^\star$, $\boldsymbol{u}^\sharp$, $\hbar$, $\Delta x$) ▷ Alg. 4
$\Psi^\flat$, $\boldsymbol{u}^\flat$ ←Projection ($\Psi^\sharp$, $\boldsymbol{u}^\sharp$, $\phi^{(j+1)}$, $p_{\text{air}}$, $\gamma$, $G$, $\hbar$, $\Delta x$, $\Delta t$)
▷ Alg. 3
$\Psi^{(j+1)}$, $\boldsymbol{u}^{(j+1)}$ ←Extrapolation ($\Psi^\flat$, $\boldsymbol{u}^\flat$, $\phi^{(j+1)}$, $\hbar$, $\Delta x$, $l_{nb}$)
▷ Alg. 5

## 4 NUMERICAL ALGORITHM

### 4.1 Method Overview

We discretize the fluid system on a staggered marker-and-cell grid. We store the wave function $\Psi$, the gauge variable $q$, and the level-set function $\phi$ at the center of each grid cell. The fluid velocity $\boldsymbol{u}$ is stored at the center of each grid face. We denote the grid cells and faces inside $\Omega_L$ as $\mathcal{V}$ and $\mathcal{E}$, respectively. We initialize the wave functions with reference to Chern et al. [2016] and Yang et al. [2021] for most of the examples. We also adopt the "rational map" in Appendix A to initialize vortex tubes in the background fluids.

Algorithm 1 summarizes our time integration scheme. In the pseudocode, $\Delta x$ is the spatial step, $\Delta t$ is the temporal step, $l_{nb}$ is a narrow-band width used to interpolate the velocity–wave fields near the interface. We usually take $p_{\text{air}} = 0$ for the air pressure, but for the bubble flow, we update $p_{\text{air}}$ by a state equation in Appendix C. The algorithm consists of four main steps, including physical quantity advection, wave-function correction, divergence-free projection, and wave-function extrapolation. The advection and projection follow Yang et al. [2021] with minor modifications, while the correction and extrapolation are devised to address the wave function's instability issue near a free surface. We introduce the numerical scheme of each step in this section. In the next section, we will discuss the interface instability problem along with its numerical solution.

### 4.2 Advection

At every time step, we advect both velocity $\boldsymbol{u}$ and wave functions $\Psi$ following the second-order semi-Lagrangian scheme, transform the wave function $\Psi$ to a velocity field with (3), and blend the advected velocity field $\boldsymbol{u}$ with the wave-function transformed velocity field. The advection of velocity $\mathrm{D}\boldsymbol{u}/\mathrm{D}t$ can be transformed into the wave

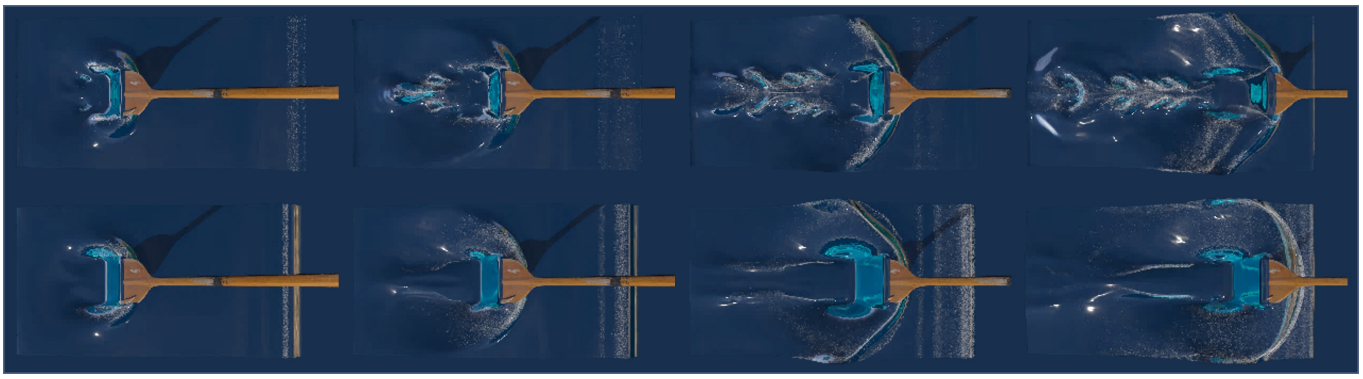

Fig. 5. Comparison of results simulated with the free-surface Clebsch solver (top) and that simulated with a velocity-based solver (bottom). This figure shows a paddling simulated on a sparse grid where the right end of the simulation domain attaches a wave generator outputs moderate waves. The figures from left to right are frames 50, 100, 150, and 200, respectively.

**Algorithm 2** Advection.

**Input:** $\Psi^{(j)}, \boldsymbol{u}^{(j)}, \phi^{(j)}, \hbar, \beta, \Delta t, \Delta x$ **Output:** $\Psi^{\star}, \boldsymbol{u}^{\sharp}, \phi^{(j+1)}$

1: $\boldsymbol{u}^{\star} \leftarrow$Advection$^{\boldsymbol{u}}(\boldsymbol{u}^{(j)}, \phi^{(j)}, \Delta t)$ ▷ SemiLagrangian
2: $\phi^{(j+1)} \leftarrow$Advection$^{\phi}(\boldsymbol{u}^{(j)}, \phi^{(j)}, \Delta t)$ ▷ SemiLagrangian
3: $\Psi^{\circ} \leftarrow$Advection$^{\Psi}(\Psi^{(j)}, \boldsymbol{u}^{(j)}, \phi^{(j)}, \Delta t)$ ▷ SemiLagrangian
4: $\Psi^{*} \leftarrow \exp[\mathrm{i}|\boldsymbol{u}^{(j)}|^2\Delta t/(2\hbar)]\Psi^{\circ}$ ▷ Right hand side of (6)
5: $\Psi^{\star} \leftarrow \Psi^{*}/\|\Psi\|$ ▷ Normalization
6: $\boldsymbol{u}^{\Psi}_{vw} \leftarrow \hbar \arg\left\langle \Psi^{\star}_{v}, \Psi^{\star}_{w} \right\rangle_{\mathbb{C}} / \Delta x$ ▷ Eq. (7)
7: $\boldsymbol{u}^{\sharp} \leftarrow \beta \boldsymbol{u}^{\Psi} + (1-\beta)\boldsymbol{u}^{\star}$ ▷ Blending

function form as

$$\frac{\mathrm{D}\Psi}{\mathrm{D}t} = \frac{\mathrm{i}|\boldsymbol{u}|^2}{2\hbar}\Psi \text{ with } \|\Psi\| = 1. \tag{6}$$

We convert the advected wave function $\Psi^{\star}$ to its velocity counterpart by calculating (3) on each grid face following

$$\boldsymbol{u}^{\Psi}_{vw} = \frac{\hbar}{\Delta x} \arg\left\langle \Psi^{\star}_{v}, \Psi^{\star}_{w} \right\rangle_{\mathbb{C}}, \tag{7}$$

with $vw \in \mathcal{E}$ and $\langle \Phi, \Psi \rangle_{\mathbb{C}} = \overline{\Phi}_1\Psi_1 + \overline{\Phi}_2\Psi_2$ [Chern et al. 2016].

The Clebsch advection in (6) may inherit many chaotic behaviors similar to what one would expect when directly evolving the flow map [Qu et al. 2019]. We use a similar way in Yang et al. [2021] to blend $\boldsymbol{u}^{\Psi}$ with the advected velocity $\boldsymbol{u}^{\star}$ as

$$\boldsymbol{u}^{\sharp} = \beta \boldsymbol{u}^{\Psi} + (1-\beta)\boldsymbol{u}^{\star}, \tag{8}$$

where $\beta \in [0, 1]$ is a blending coefficient to characterize the preference for the evolving "Clebsch flow" or "velocity flow". In details, for $\beta = 0$, the fluid is completely governed by velocity equations (4), for $\beta = 1$, the fluid is completely governed by wave equations (5), and for $0 < \beta < 1$, the fluid is governed by both equations. Algorithm 2 summarizes the advection and blending.

## 4.3 Wave-Function Correction

After advection, we employ a correction step to update the wave-function values that might cause any numerical instability near a free surface due to the large velocities $\boldsymbol{u}^{\Psi}$ calculated from the finite-difference of two neighboring wave functions with a drastic deviation. Such local wave-function deviations are caused by the fluid surface's topological changes. The interface collision brings wave functions that were geodesically far (and therefore manifested a large phase difference) next to each other on grid cells. Each local wave-function discrepancy may cause a large, unphysical velocity on a grid face that could unrealistically perturb the local fluid surface. To solve this problem, we propose a local correction algorithm that maps from $\Psi^{\star}$ to $\Psi^{\sharp}$ to smooth the deviating wave functions locally according to the advected velocities. We postpone the discussion of the mathematical and numerical details of the correction scheme to Section 5.2.

**Algorithm 3** Velocity and wave-function projection.

**Input:** $\Psi^{\sharp}, \boldsymbol{u}^{\sharp}, \phi^{(j+1)}, p_{\mathrm{air}}, \gamma, G, \hbar, \Delta x, \Delta t$ **Output:** $\Psi^{\flat}, \boldsymbol{u}^{\flat}$

1: $q_{\Gamma} \leftarrow$Calculating $(p_{\mathrm{air}} + \gamma\kappa - G)$ on $\Gamma$ ▷ Using $\phi^{(j+1)}$
2: $q \leftarrow$SolvingPoisson$(\boldsymbol{u}^{\sharp}, q_{\Gamma}, \Delta t)$ ▷ Eqs. (9) and (10)
3: $\boldsymbol{u}^{\flat}, \Psi^{\flat} \leftarrow$Projection$(\boldsymbol{u}^{\sharp}, \Psi^{\sharp}, q, \Delta x, \Delta t)$ ▷ Eq. (11)

## 4.4 Velocity and Wave-Function Projection

To enforce the divergence-free conditions on velocity and wave function, we solve the Poisson equation for $q$ in (4) and (5) as

$$\Delta t \nabla^2 q = \boldsymbol{\nabla} \cdot \boldsymbol{u}^{\sharp}, \ \boldsymbol{x} \in \Omega_L, \tag{9}$$

with Neumann boundary conditions on the solid boundary and Dirichlet boundary conditions with a jump condition specified by the net effects of the body forces and surface tensions on the free-surfaces:

$$\begin{cases} \partial_n q = 0, \ \boldsymbol{x} \in \partial\Omega_b, \\ q = p_{\mathrm{air}} + \gamma\kappa - G, \ \boldsymbol{x} \in \Gamma, \end{cases} \tag{10}$$

where the curvature $\kappa$ on $\Gamma$ can be calculated from the level-set function $\phi^{(j+1)}$. Then we use $q$ to project $\boldsymbol{u}^{\sharp}$ and $\Psi^{\sharp}$ and obtain the

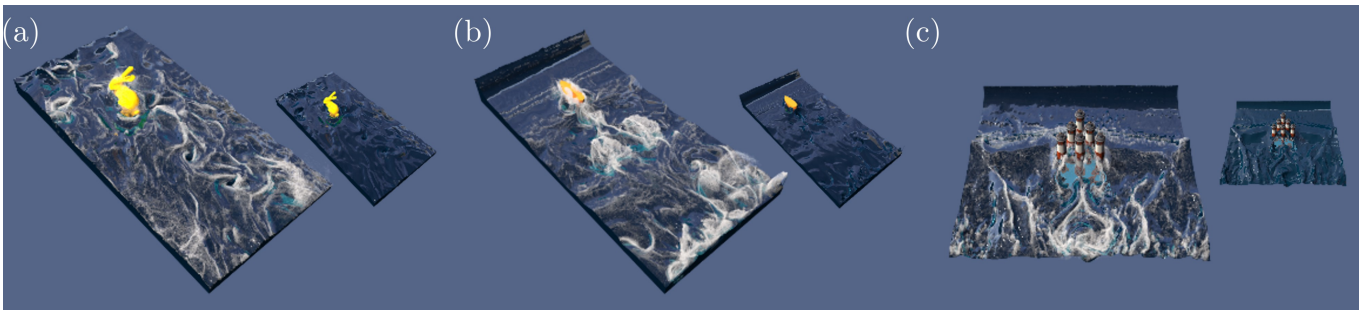


Fig. 6. Open water simulation with a wave generator attached to the top end of the simulation domain. This figure shows the results for (a) bunny example at frame 450, (b) boat example at frame 250, and (c) lighthouse example at frame 200 with and without the spray and foam particles separately.

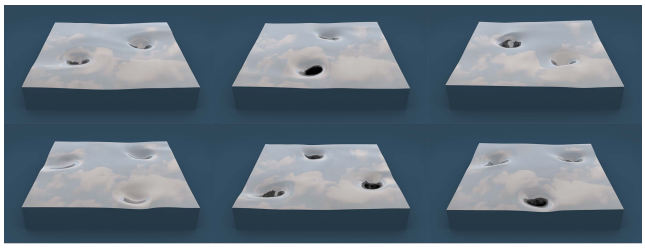

Fig. 7. Interactions of two (top) and three (bottom) parallel vortices on free-surface. The top figures from left to right are frame 200, 300 and 400 of the dual-vortex simulation. The bottom figures from left to right are frame 500, 600, and 700 of the tri-vortex simulation, respectively.

projected velocity and wave function for the next time step:

$$\begin{cases} \boldsymbol{u}^{\flat} = \boldsymbol{u}^{\sharp} - \Delta t \nabla q, \\ \boldsymbol{\Psi}^{\flat} = \boldsymbol{\Psi}^{\sharp} \exp\left(-\frac{iq\Delta t}{\hbar}\right), \end{cases} \tag{11}$$

where $\nabla q$ is computed by the ghost fluid method [Fedkiw et al. 1999] with given $\Delta x$. Algorithm 3 summarizes the projection from $\boldsymbol{u}^{\sharp}$ and $\boldsymbol{\Psi}^{\sharp}$ to the solenoidal fields $\boldsymbol{u}^{\flat}$ and $\boldsymbol{\Psi}^{\flat}$.

### 4.5 Velocity and Wave-Function Extrapolation

To advect fluid near the interface, we constantly extrapolate the fluid velocity field to a narrow band outside the interface. However, if we employ the same constant extrapolation scheme to the wave-function field, we will get a zero velocity field from (3) (the gradient of a constant is 0). The fact that the velocity is the wave-function gradient in (3) motivates us to extrapolate wave functions based on the information of the extrapolated velocities in the narrow band. Therefore, we propose a modified fast-marching method for wave-function extrapolation. Similar to the wave-function correction step, the purpose of this wave-function extrapolation is to ensure consistent velocity and wave-function fields in the level-set narrow band, *i.e.*, the difference between the transformed velocity field based on the wave-function field and the velocity field needs to be minimized. We postpone the discussion of mathematical and numerical details of the extrapolation algorithm to Section 5.3.

## 5 WAVE-FUNCTION CORRECTION AND EXTRAPOLATION ALGORITHMS

In this section, we describe the mathematical and algorithmic details of the proposed wave-function and extrapolation methods. We first give the mathematical definition of the wave-function discrepancy problem and a numerical example in Section 5.1. Then, we discuss the optimization-based wave-function correction and extrapolation algorithm in Section 5.2 and Section 5.3. Validation experiments for the two algorithms are presented in Section 6.1.

### 5.1 Mathematical Definition

A given wave function $\boldsymbol{\Psi}$ maps to a unique velocity field $\boldsymbol{u}$ based on (3). However, a given velocity field $\boldsymbol{u}$ corresponds to an infinite number of wave functions satisfying (3), with a constant phase offset. We describe such one-to-many relation as the following theorem.

Theorem 5.1. *The wave functions $\boldsymbol{\Psi}$ and $e^{i\theta}\boldsymbol{\Psi}$ correspond to the same velocity field in* (3) *with $\theta$ as a constant global phase.*

Proof. Since $\theta$ is a constant, we have

$$\nabla(e^{i\theta}\boldsymbol{\Psi}) = e^{i\theta}\nabla\boldsymbol{\Psi}. \tag{12}$$

Substituting (12) into (3) yields

$$\left\langle \nabla(e^{i\theta}\boldsymbol{\Psi}), \mathrm{i}(e^{i\theta}\boldsymbol{\Psi}) \right\rangle_{\mathbb{R}} = \langle \nabla\boldsymbol{\Psi}, \mathrm{i}\boldsymbol{\Psi} \rangle_{\mathbb{R}}, \tag{13}$$

*i.e.*, $\boldsymbol{\Psi}$ and $e^{i\theta}\boldsymbol{\Psi}$ correspond to the same velocity field in (3). □

The one-to-many mapping from velocity to wave function may cause instability issues near the interface. We show a simple example of this in Figures 8(a), 8(b), 8(d) by merging two fluid regions with the same velocity but different wave functions. The schematic fluid circled in the blue box on the right is the merging of the two parts circled in red in the blue box on the left. We divide the domain circled in the blue box into two parts, the left and the right. The wave function on the right domain is $\Psi_1 = \Psi_2 = \sqrt{2}\exp[\mathrm{i}(x + 0.6y)]/2$, and the wave function on the left is $\Psi_1 = \Psi_2 = \sqrt{2}\exp[\mathrm{i}(x + 0.6y + \theta)]/2$, where $\theta$ is a constant phase difference, taken as 0 in Figure 8(a) and $0.4\pi$ in the Figures 8(b) and 8(d). In the beginning, two topologically disconnected fluid regions have the same constant velocity field. This velocity field is transformed from two different sets of wave functions (as shown in the left column). Then the two regions get merged topologically. As shown in Figure 8(a),

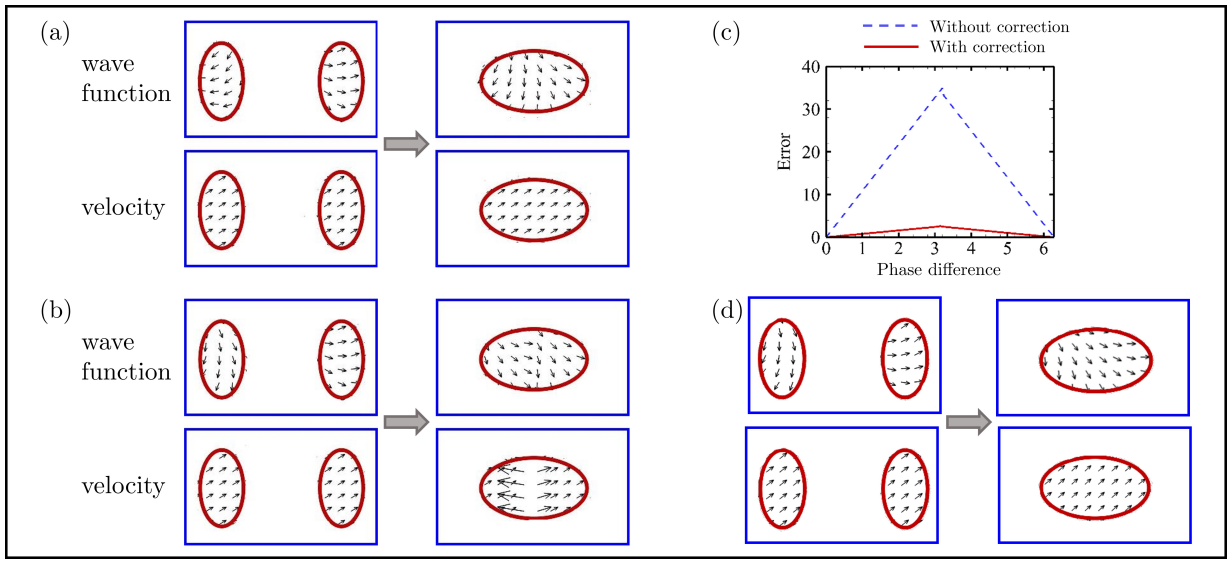


Fig. 8. The distribution of the velocity field of two topologically disconnected fluids close to each other. The wave functions in the two domains enclosed by the red lines have (a) the same global phase, (b) different global phases without wave function correction, and (d) different global phases with wave function correction. The arrow line in the wave function schematic denotes the first component of the two-component wave function. In (c), we show the relationship between the relative error of wave function transformed velocity and global phase difference.

the transformed velocity fields of the wave functions in the two regions with the same global phase still maintain smoothness after merging. However, as shown in Figure 8(b), the transformed velocity fields of the wave functions in the two regions with drastically different global phases may manifest a nonphysical discontinuity in the narrow band after merging, which will further cause velocity artifacts. The phase difference of the wave function in Figure 8(b) can be reduced by our correction algorithm (see Section 5.2) and therefore generate a smooth velocity field in Figure 8(d). We also plot the relationship between the relative error of the wave-function-transformed velocity and global phase difference in Figure 8(c). The curve in the figure varies periodically with the phase difference of the wave function and reaches the maximum error at the phase of $\pi$. We can see that the relative error is significantly reduced with the correction algorithm.

## 5.2 Wave-Function Correction Algorithm

Inspired by the discrete wave–velocity relation (7), we define an extrapolation scheme that backtraces the wave function with neighboring face velocity

$$\mathbf{\Phi}_{vw} = \mathbf{\Psi}_w \exp\left(\frac{\mathrm{i}\boldsymbol{u}^{\sharp}_{vw}\Delta x}{\hbar}\right). \tag{14}$$

Ideally, if the extrapolation step in (14) finds the precise wave function in cell $v$, then we would have

$$\|\mathbf{\Psi}_v - \mathbf{\Phi}_{vw}\| = 0. \tag{15}$$

Therefore, we optimize a constraint defined on the grid faces

$$\mathbf{\Psi}^{\sharp} = \underset{\mathbf{\Psi}\in\{\boldsymbol{\psi}\,|\,\|\boldsymbol{\psi}\|=1\}}{\arg\min} \sum_{vw\in\mathcal{E}} \|\mathbf{\Psi}_v - \mathbf{\Phi}_{vw}\|^2, \tag{16}$$

to achieve an averaged minimum of (15). We introduce $\Phi_v$ in the following theorem to reduce the energy of (16) until it converges to a stable value.

Theorem 5.2. *Suppose $s \in \mathcal{V}$ is a grid cell, we define $\Phi_v$ as*

$$\mathbf{\Phi}_v = \begin{cases} \dfrac{\sum_{w\in\{u|su\in\mathcal{E}\}} \mathbf{\Phi}_{sw}}{\left\|\sum_{w\in\{u|su\in\mathcal{E}\}} \mathbf{\Phi}_{sw}\right\|}, & v = s, \\ \mathbf{\Psi}_v, & v \neq s, \end{cases} \tag{17}$$

*then we have*

$$\sum_{vw\in\mathcal{E}} \|\mathbf{\Phi}_v - \mathbf{\Phi}_{vw}\|^2 \leq \sum_{vw\in\mathcal{E}} \|\mathbf{\Psi}_v - \mathbf{\Phi}_{vw}\|^2 \tag{18}$$

Proof. Applying $\|\mathbf{\Phi}_v\| = \|\mathbf{\Psi}_v\|$ to (18) yielding

$$0 \leq \sum_{v\in\mathcal{V}} \left\langle \mathbf{\Phi}_v - \mathbf{\Psi}_v, \sum_{w\in\{u|vu\in\mathcal{E}\}} \mathbf{\Phi}_{vw} \right\rangle_{\mathbb{R}}. \tag{19}$$

Using (17), (19) can be converted to

$$\langle \mathbf{\Psi}_s, \mathbf{\Phi}_s \rangle_{\mathbb{R}} \leq 1. \tag{20}$$

The equation (20) holds with the triangle inequality of $\langle\cdot\rangle_{\mathbb{R}}$ and the unitarity of the wave function. □

Theorem 5.2 indicates that the energy of (16) reduces each time when updating $\mathbf{\Psi}$ to $\mathbf{\Phi}$ in (17). Thus, we can minimize (16) iteratively by using (17) for a given initial wave function $\mathbf{\Psi}^{\star}$.

Algorithm 4 summarizes our wave-function correction algorithm. In our implementation, the total iteration step setting is $S_t = 4$ by default, and the cell set $\mathcal{V}_s^k$ for optimization is "non-adjacent" to ensure that adjacent cells do not appear in the same iteration step.

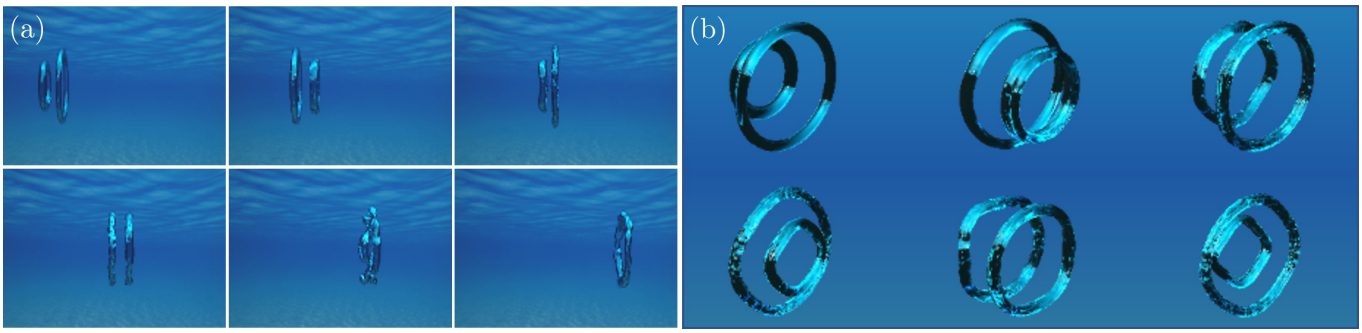


Fig. 9. Leapfrogging of two bubble rings simulated on (a) $128 \times 256 \times 128$ and (b) $256 \times 512 \times 256$ grids. The figures from left to right and top to bottom are frames 0, 66, 132, 198, 264, and 330, respectively.

**Algorithm 4** Correction of wave function.

**Input:** $\Psi^{\star}$, $\boldsymbol{u}^{\sharp}$, $\hbar$, $\Delta x$ **Output:** $\Psi^{\sharp}$

$\Psi^{\sharp,0} \leftarrow \Psi^{\star}$ ▷ Initialization
**for** $k = 1 : S_t$ **do**
  $\mathcal{V}_s^k \leftarrow$Determine the cell set for optimization
  **for** $v \in \mathcal{V}_s^k$ **do**
    $\Psi_w \leftarrow \Psi_w^{\sharp,k-1}$
    $\Phi_{vw} \leftarrow$Extrapolation$(\Psi_w, \boldsymbol{u}_{vw}^{\sharp}, \hbar, \Delta x)$ ▷ Eq. (14)
    $\Psi_v^{\sharp,k} \leftarrow$Optimization$(\Phi_{vw})$ ▷ Eq. (17)
  **end for**
**end for**
$\Psi^{\sharp} \leftarrow \Psi^{\sharp,S_t}$

The set $\mathcal{V}_s^k$ contains two subsets of cells to satisfy two optimization goals. When two wave functions with large phase differences create a high velocity on their shared face after blending in (8), we would prefer the system does not gain any artificial momentum due to the wave function transformed velocity field. Therefore, the first optimization goal is to avoid spikes in face velocity values. In addition, to match wave function in solid and fluid, the second optimization goal is to avoid discontinuity across the solid-fluid interface. Thus, we select the second subset of cells in the narrow band around the solid to pass the wave function values inside the solid to the surrounding fluid. We search the first correction subset with a red-black Gauss-Seidel algorithm and collect the second correction subset with a fast-marching like local search algorithm.

### 5.3 Wave-Function Extrapolation Algorithm

Algorithm 5 summarizes the extrapolation scheme for velocity and wave function. For the wave function extrapolation, we collect the cells within the narrow band of the interface in set

$$\mathcal{V}_I = \{v | l_{nb} > \phi_v^{(j+1)} > 0\}. \tag{21}$$

Next, we sort $\mathcal{V}_I$ with respect to $\phi^{(j+1)}$ in a descending order and store the results to an ordered set $\mathcal{O}$. Then, we extrapolate the wave function based on the extrapolated velocity field in the order specified by $\mathcal{O}$. Specifically, we calculate the wave function for the

**Algorithm 5** Extrapolation of velocity and wave function.

**Input:** $\Psi^{\flat}$, $\boldsymbol{u}^{\flat}$, $\phi^{(j+1)}$, $\hbar$, $\Delta x$, $l_{nb}$ **Output:** $\Psi^{(j+1)}$, $\boldsymbol{u}^{(j+1)}$

$\boldsymbol{u}^{(j+1)} \leftarrow$ConstantExtrapolation$(\boldsymbol{u}^{\flat}, \phi^{(j+1)}, l_{nb})$
$\mathcal{V}_I \leftarrow$Determine the cell set for extrapolation ▷ Eq. (21)
$\mathcal{O} \leftarrow$Sort $\mathcal{V}_I$ into an ordered set according to the $\phi^{(j+1)}$
**for** $k = 1 : |\mathcal{O}|$ **do**
  $\Phi_{\mathcal{O}(k)w}^{\flat} \leftarrow$Extrapolation$(\Psi_w^{\flat}, \boldsymbol{u}_{\mathcal{O}(k)w}^{(j+1)}, \hbar, \Delta x)$ ▷ Eq. (24)
  $\Psi_{\mathcal{O}(k)}^{\flat} \leftarrow$Optimization$(\Phi_{\mathcal{O}(k)w}^{\flat})$ ▷ Eq. (22)
**end for**
$\Psi^{(j+1)} \leftarrow$ Extrapolated$\Psi^{\flat}$

$k$-th cell as

$$\Psi_{\mathcal{O}(k)}^{\flat} = \frac{\sum_{w \in \{u | \mathcal{O}(k)u \in \mathcal{E}_k\}} \Phi_{\mathcal{O}(k)w}^{\flat}}{\left\| \sum_{w \in \{u | \mathcal{O}(k)u \in \mathcal{E}_k\}} \Phi_{\mathcal{O}(k)w}^{\flat} \right\|}. \tag{22}$$

in a similar way as (17). Here the edge set $\mathcal{E}_k$ is updated with

$$\{vw | v \text{ and } w \text{ are incident grid cells in} \mathcal{V} \cup \mathcal{O}(1) \cup \cdots \cup \mathcal{O}(k)\}, \tag{23}$$

and

$$\Phi_{\mathcal{O}(k)w}^{\flat} = \Psi_w^{\flat} \exp\left( \frac{\mathrm{i} \boldsymbol{u}_{\mathcal{O}(k)w}^{(j+1)} \Delta x}{\hbar} \right). \tag{24}$$

## 6 NUMERICAL RESULTS

### 6.1 Validation Examples

We validate our approach by comparing with [Yang et al. 2021] in several standard free-surface flow examples to demonstrate the efficacy of our wave-function correction and extrapolation algorithms. We remark that the structure of the wave function does not change with dimensionality. For all two-dimensional flows shown here, the wave function stored at the center of each computational cell is also a two-component complex variable.

*6.1.1 Validation of the Wave Function Correction.* In Figure 11, we test the correctness of our optimization algorithm by optimizing a randomly initialized wave function field to the target of a Taylor–Green velocity field [Taylor and Green 1937]. In this experiment, our algorithm converges under around 20 iterations.

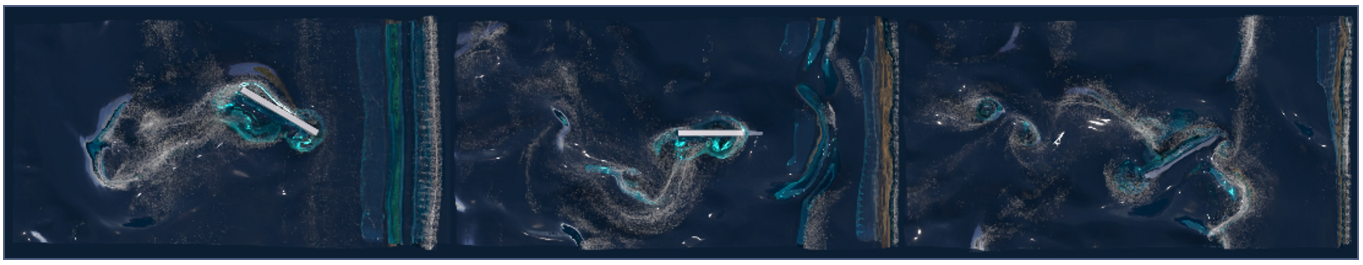

Fig. 10. A wiggling fishtail simulated on a sparse grid. The figures from left to right are frames 200, 550 and 900, respectively.

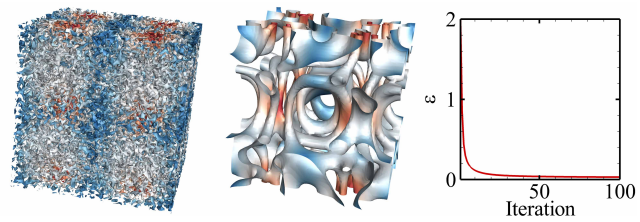


Fig. 11. Construction of wave function from Taylor–Green velocity using Algorithm 4 with the input of a random initial wave function. The figures from left to right are isosurface $s_1 = -0.5$ of the input and corrected wave function; and mean squared error of $\|\mathbf{\Psi}_v - \mathbf{\Phi}_{vw}\|^2$ in (16).

In Figure 12, we examine the necessity of the correction step through the interface extending and merging of a dam breaking example. As the top figures in Figure 12 show, when the correction step is disabled, we observe over-energetic unnatural movements near contacting interfaces like enlarging bubbles or growing wave peaks. The aforementioned visual artifacts are eliminated with the correction step. We can also conclude that a non-physical increment of kinetic energy may occur without the correction step, as shown in the second peak of the kinetic energy plot in Figure 12.

Figure 13 shows zoomed-in views of wave functions between two colliding waves in the dam break example, with and without the correction step. Without the correction step, the wave function is discontinuous near the interface after the merge, causing a local spiky velocity. In the visualized wave-function field, these discontinuities are observed as phase differences, and a face velocity is calculated as the wave-function gradient scaled by $\hbar$. The local wave-function correction step (shown as red dots in the figure) smooths these discontinuities and therefore avoids unnatural velocities.

*6.1.2 Validation Wave Function Extrapolation.* Here we examine the necessity of the fast-marching extrapolation step. Figure 14 showcases how three methods perform in three tests. From top to bottom, the three extrapolation methods are without extrapolation, with our fast-marching extrapolation, and with a constant extrapolation. The hydrostatic test shows that a static water surface can be obtained with both a constant extrapolation and our approach. However, the naive constant extrapolation scheme will cause a significant deviation between the wave function and the velocity in the narrow band, which will lead to an overly damped interface motion. Our approach addressed this issue by always maintaining a consistent pair of velocity and wave functions in the narrowband and therefore producing natural interface behaviors.

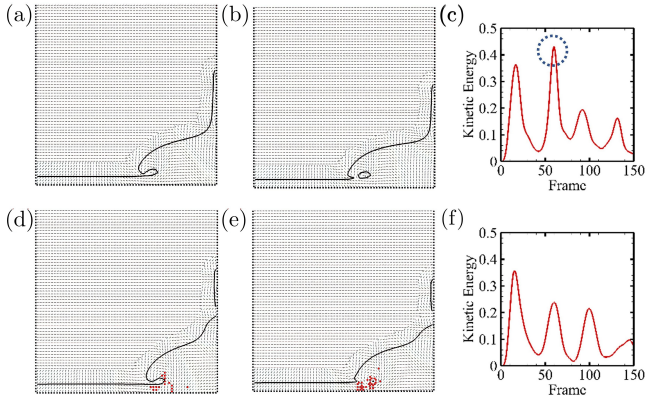


Fig. 12. Comparison of the wave-function distribution of a two-dimensional dam breaking (a), (b) without, and (d), (e) with wave-function correction. (a), (d) and (b), (e) are the two adjacent frames before and after the interface merge, respectively. (see the supplemented video for more details). We convert the first component of the two-component wave function to a vector in two-dimensional Euclidean space and plot it as an arrow line. Red points in (d) and (e) highlight the grid cells involved in the correction algorithm. (c) and (f) are the evolution of kinetic energy with and without wave-function correction, respectively. The dotted circle in (c) denotes a non-physical increment of kinetic energy without the correction step.

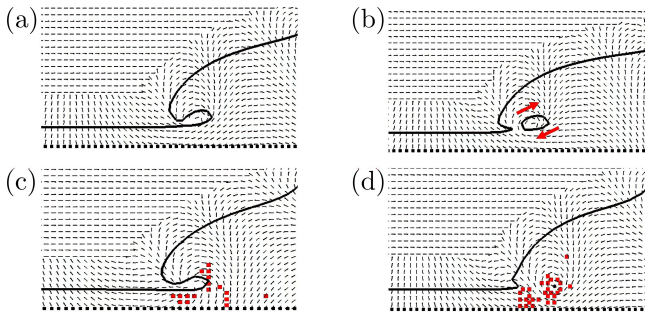


Fig. 13. Close-up views of wave functions of the two-dimensional dam example in Figure 12. The red arrow line in (b) is a schematic diagram of the wave function with opposite directions near the interface.

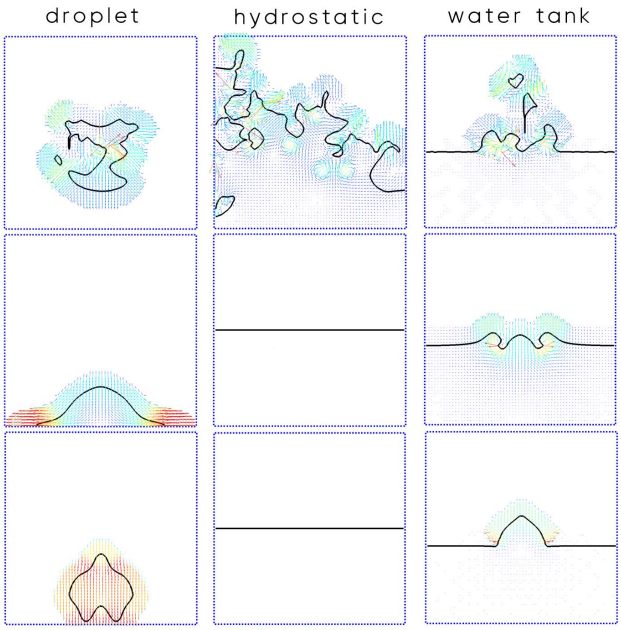


Fig. 14. Comparison of simulation results of a two-dimensional droplet (leftmost column) at frame 20, a two-dimensional hydrostatic test (middle column) at frame 100, and a two-dimensional droplet falling into a water tank (rightmost column) at frame 15 without (top figures) extrapolation, with a naive constant extrapolation (bottom figures) and with our fast-marching extrapolation (middle figures) (see the supplemented video for more details).

## 6.2 Three-Dimensional Free-Surface Vortex

We design several novel fluid simulation examples to evaluate the performance of our algorithm, including sink vortex, horseshoe vortex with their free ends attached to the water surface, the interaction of parallel vortices on the free surface, reconnection of two bubble rings, leapfrogging bubble rings, vortex shedding from solid obstacles. In addition, we compared our method with the Clebsch gauge method [Yang et al. 2021] and the standard velocity-based solver regarding their visual effects. In Table 1, we list the parameters of our numerical experiments and we refer the readers to the animations in our supplemental video for more visual details.

*6.2.1 Vortex on Free-Surface.* In this set of examples, we focus on the evolution and interaction of vortices on the free surface. All the examples in this subsection are initialized with an analytical wavefunction field on a steady background. The free surface deforms and forms an interface-vortex structure under the action of the self-induced velocity of the vortices. The surface vortex examples are great illustrations of how our algorithm has an advantage in vortices preservation.

*Sink vortex.* With a rotating background flow, we can easily make vortices in a sink at home. However, this common phenomenon in everyday life is yet nontrivial to simulate. The top row and the bottom row of Figure 3 show the formation of surface vortices during the draining process of a sink with one and two sink holes

Table 1. Details of the Simulation Examples. (a) reconnecting bubble rings, (b) vortex in a sink with one hole, (c) vortex in a sink with two holes, (d) horseshoe vortex, (e) paddling, (f) a bunny in waves, (g) multiple parallel surface vortex, (h) a boat in waves, (i) a lighthouse in waves, (j) reconnecting bubble rings, (j) leapfrogging bubble rings, and (k) fishtail wiggling. (d) was performed on a server with a 36-core CPU and an Nvidia RTX 2080 Ti GPU. (b) and (g) were performed on a server with a 64-core CPU and a Quadro RTX 8000 GPU. (c) was performed on a server with a 40-core CPU and an Nvidia Tesla V100 GPU. (a), (e), (j), and (k) were performed on personal computer with a 32-core CPU and an Nvidia RTX 2080 Ti. (f), (h), and (i) were performed on a personal computer with a 6-core CPU and an Nvidia RTX 2070. The time listed in the table is the average time per frame. Each frame may have several time steps, depending on the CFL number.

| Cases | Figures | $\beta$ | $\hbar$ | Resolution | CFL | Time(s) |
|---|---|---|---|---|---|---|
| (a) | Fig. 2 | 0.1 | 0.4 | $256 \times 256 \times 256$ | 5 | 14.4 |
| (b) | Fig. 3 | 0.1 | 0.05 | $256 \times 128 \times 256$ | 1 | 13.3 |
| (c) | Fig. 3 | 0.1 | 0.05 | $512 \times 128 \times 256$ | 1 | 23.85 |
| (d) | Fig. 4 | 0.5 | 0.2 | $256 \times 128 \times 128$ | 1 | 16.9 |
| (e) | Fig. 5 | 0.5 | 0.3 | $128 \times 32 \times 64$ | 4 | 2.3 |
| (f) | Fig. 6(a) | 0.5 | 0.25 | $256 \times 64 \times 128$ | 4 | 10.1 |
| (g) | Fig. 7 | 1 | 0.05 | $128 \times 64 \times 128$ | 1 | 1.7 |
| (h) | Fig. 6(b) | 0.5 | 0.25 | $256 \times 64 \times 128$ | 4 | 10.2 |
| (i) | Fig. 6(c) | 0.5 | 0.25 | $256 \times 64 \times 256$ | 4 | 18.2 |
| (j) | Fig. 9(a) | 0.1 | 0.4 | $256 \times 128 \times 128$ | 5 | 4.4 |
| (k) | Fig. 10 | 0.5 | 0.3 | $128 \times 32 \times 64$ | 4 | 2.3 |

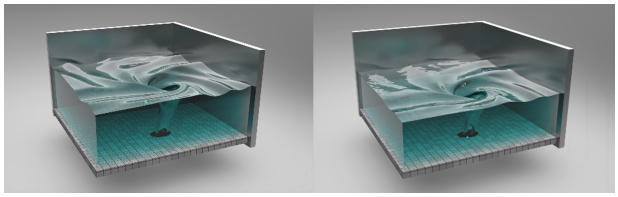

Fig. 15. Comparison of a single hole sink vortex simulated with our free-surface Clebsch solver (left) and Clebsch gauge solver [Yang et al. 2021] (right) at frame 200.

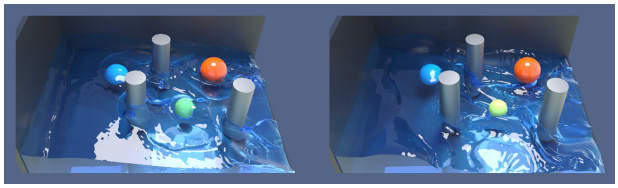

Fig. 16. A stream flowing over multiple obstacles at frames 50 and 250.

respectively. Our algorithm captures spiral folds created by surface vortices extending from the sink hole(s) to the water surface. Figure 15 shows that the sink vortex simulated with our free-surface Clebsch solver maintains more surface details compared to Yang et al. [2021].

*Horseshoe vortex.* Water-striders can move freely on the water surface, and horseshoe vortex with their free ends attached to the water surface is the key to understanding this phenomenon [Hu

et al. 2003]. In this experiment, we initialize the wave function field with a vortex ring on a steady background. Self-induced movements of the vortex ring energize the entire water body via surface waves. In Figure 4, the simulation shows that our algorithm can maintain the vortex moving through the entire simulation domain. Visually, we can observe a pair of air tubes "walking" from one end to the other, indicating that the vortex ring moved across the box.

*Interaction of parallel vortices on free-surface.* Figure 7 plots the interactions of two and three parallel vortices on the free surface. Different from the previous horseshoe vortex example in Figure 4, this example has multiple surface vortices that rotate in the same direction instead of a U-shape vortex tube as in the initial condition. This example is designed to show that our algorithm can preserve the interactions between multiple parallel vortices. An interesting observation is that every surface vortex tends to rotate around the center simulation domain while keeping a distance from each other.

*6.2.2 Vortex-Solid Interaction.* In this set of examples, we show that even on sparse grids, our algorithm can reproduce vortex shedding around the boundaries of stationary or dynamic solid objects. We admire the insightful quote from Prof. Shi-Jia Lu (1911–1986) made around 1980 [Wu et al. 2006]: "The essence of fluid is vortices. A fluid cannot stand rubbing; once you rub it there appear vortices." From a boat cruising in the ocean to a fish wiggling its tail in a water tank, our simulations show rich details of vortices around solid boundaries. We remark that the spray and foam particles in these scenarios were generated as a post-process, which has been a common practice for large-scale open-water simulation starting from Losasso et al. [2008].

*Stationary solids.* Figure 6(a) shows a bunny generating vortices under the tidal waves. In Figure 6(b), the boat scene has a similar setup of simulating a larger domain size on a lower resolution grid. Compared to the most recent work on large-scale open-water simulation [Huang et al. 2021] where the high-frequency details near the solid objects are achieved with fine FLIP simulation, we can observe a similar trail of spray and foam form around the rear of the boat. We show the evolution of the kinetic energy of the boat example in Figure 17, where the frequency of the kinetic oscillation is aligned with the water wave generator. We test a larger set of obstacles in Figure 6(c). To honor pioneers [Chentanez and Müller 2011] in real-time large-scale simulations, we place a triangular array of lighthouses against large waves and get the energetic surface. We also show a stream flowing over multiple obstacles in Figure 16. A similar example can be found in Yang et al. [2021], with less interesting surface details.

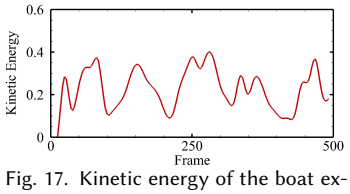

Fig. 17. Kinetic energy of the boat example.

*Paddling.* Figure 5 shows turbulent flow behaviors on the water surface when the paddle moves along the horizontal axis. The right end of the simulation domain attaches a wave generator that outputs medium height waves. We direct the reader's attention to the chain of whirlpools formed along with the paddle movement. Compared to our standard grid-based simulation where all the shallow vortex details dissipate quickly, our solver couples well with moving solid boundaries and highlights these cascaded vortex structures.

*Fishtail.* Figure 10 showcases the details of boundary-induced turbulence around a rotating fishtail. The incoming waves come from the right end of the simulation domain, get interrupted by the wiggling fishtail, then evolve into shredded wavelets. The solid object wiggles in a small margin along the vertical axis periodically and develops a zig-zag patterned vortex chain. This example emphasizes that our algorithm can correctly simulate boundary vortical flow around rotating solid objects.

*6.2.3 Bubble Rings.* Bubble rings of various thicknesses are particularly interesting to the computer graphics community when many research works focus on the air-liquid interface. Since bubble rings rely on the interaction of vortex filaments and surface tension to maintain their geometry, properly modeling underwater bubble rings has been a challenging task. Padilla et al. [2019] developed a specific geometric model to simulate complex bubble ring interactions in the computer graphics community for the first time. Our algorithm can achieve similar results with a volume-preserving term added to our free-surface simulator. We believe this is the first general grid-based fluid solver that is capable of capturing such fine details of bubble-vortex interactions.

*Reconnecting bubble rings.* Figure 2 shows two bubble rings of different radii move, deform, and merge into one ring. After merging, a smaller ring is separated off because of the vigorous subsequent fluctuations along the main bubble ring. Here we provide a comparison of results simulated with our solver and with a standard velocity-based solver. Note that the persistence of the vortex structure plays an essential role in this example. The bubble rings break into tiny bubbles as the vortices around them dissipate over evolution with the standard velocity-based fluid solver's inability to preserve vortex filaments around bubble rings.

*Leapfrogging bubble rings.* Figure 9(a) shows the leapfrog of two bubble rings. Starting with two aligned bubble rings, the front ring expands and moves forward with a smaller speed, and the rear ring shrinks and moves forward with a greater speed and further goes through the front ring, completing the first round of leapfrogging. After the third round, the two rings merge into one ring and continue moving forward. With higher resolution, Figure 9(b) shows the two rings can maintain the leapfrog longer in a more stable flow field.

## 7 CONCLUSION

We propose a Clebsch method to simulate incompressible free-surface flow. By introducing a correction scheme for wave function, and fast-marching extrapolation, we completely resolve the numerical instability near the interface that exists in Yang et al. [2021] and accurately resemble fine details of surface vortex–wave interactions. With a simplified volume control term added to our base free-surface Clebsch solver, our solver reproduces complex behaviors of bubble ring interactions. This method is capable of simulating a wide range of free-surface fluid phenomena including sink vortex, horseshoe

vortex, leapfrogging bubble rings, reconnecting bubble rings, and surface vortex-solid interactions with solid objects.

## 8 LIMITATION AND FUTURE WORK

*Limitation.* For the parameters $\hbar$ introduced in Chern et al. [2016] and $\beta$ in Yang et al. [2021], we have an intuitive understanding of their numerical results but have yet to understand their mathematical implications. We observe that the vortex structure of the Clebsch flow tends to be of a similar scale, which is speculated to be related to the constant parameter $\hbar$. Additionally, our current algorithm does not consider the effect of multi-phase flow due to the challenge of defining the jump condition on the wave functions across the interface. Lastly, we have not properly defined the viscosity term for the Clebsch method due to theoretical difficulties.

*Future work.* It would be exciting to see our free-surface Clebsch fluid simulation framework being applied to compressible flows and magnetohydrodynamics. In addition to our current one-way solid-fluid coupling scheme, we should explore the possibility of implicit two-way fluid-solid coupling and extend the Clebsch representation to rigid body dynamics. Note that the Clebsch map is originally defined with Lagrangian coordinates, so it would be natural for the Clebsch method to work with particle-based fluid simulations. The study of other Clebsch flows besides the spherical Clebsch flow is also a worthy topic of research.

## ACKNOWLEDGMENTS

We thank all the anonymous reviewers for their constructive feedback. We acknowledge NSF-1919647, 2106733, and 2144806 for funding support. We credit the Houdini Education license for producing the video animations.

## REFERENCES

M. Aanjaneya, S. Patkar, and R. Fedkiw. 2013. A monolithic mass tracking formulation for bubbles in incompressible flow. *J. Comput. Phys.* 247 (2013), 17–61.
H. Bateman. 1929. Notes on a differential equation that occurs in the two-dimensional motion of a compressible fluid and the associated variational problems. *Proc. R. Soc. London Ser. A* 125 (1929), 598–618.
N. Chentanez and M. Müller. 2011. Real-time Eulerian water simulation using a restricted tall cell grid. *ACM Trans. Graph.* 30 (2011), 10.
A. Chern. 2017. *Fluid dynamics with incompressible Schrödinger flow.* Ph.D. Dissertation. California Institute of Technology.
A. Chern, F. Knöppel, U. Pinkall, and P. Schröder. 2017. Inside fluids: Clebsch maps for visualization and processing. *ACM Trans. Graph.* 36 (2017), 142.
A. Chern, F. Knöppel, U. Pinkall, P. Schröder, and S. Weißmann. 2016. Schrödinger's smoke. *ACM Trans. Graph.* 35 (2016), 77.
A. Clebsch. 1859. Ueber die Integration der hydrodynamischen Gleichungen. *J. Reine Angew. Math.* 56 (1859), 1–10.
M. Coquerelle and G. H. Cottet. 2008. A vortex level set method for the two-way coupling of an incompressible fluid with colliding rigid bodies. *J. Comput. Phys.* 227 (2008), 9121–9137.
C. Eckart. 1938. The electrodynamics of material media. *Phys. Rev.* 54 (1938), 920–923.
D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. 2002. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.* 183 (2002), 83–116.
H. Ertel. 1942. Ein neuer hydrodynamischer Wirbelsatz. *Wirbelsatz. Meteorol. Z.* 59 (1942), 271—-281.
R. Fedkiw and S. Osher. 2002. Level set methods and dynamic implicit surfaces. *Surfaces* 44 (2002), 77.
R. Fedkiw, J. Stam, and H. W. Jensen. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques.* 15–22.
R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher. 1999. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.* 152 (1999), 457–492.
F. Gibou, R. Fedkiw, and S. Osher. 2018. A review of level-set methods and some recent applications. *J. Comput. Phys.* 353 (2018), 82–109.
J. Hao, S. Xiong, and Y. Yang. 2019. Tracking vortex surfaces frozen in the virtual velocity in non-ideal flows. *J. Fluid Mech.* 863 (2019), 513–544.
P. He and Y. Yang. 2016. Construction of initial vortex-surface fields and Clebsch potentials for flows with high-symmetry using first integrals. *Phys. Fluids* 28 (2016), 037101.
H. Helmholtz. 1858. Uber integrale der hydrodynamischen Gleichungen welche den Wirbel-bewegungen ensprechen. *J. Reine Angew. Math* 55 (1858), 25–55.
H. Hopf. 1931. Über die Abbildungen der Dreidimensionalen Sphäre auf die Kugelfläche. *Math. Ann.* 104 (1931), 637–665.
B. Houston, M. B. Nielsen, C. Batty, O. Nilsson, and K. Museth. 2006. Hierarchical RLE level set: A compact and versatile deformable surface representation. *ACM Trans. Graph.* 25 (2006), 151–175.
D. L. Hu, B. Chan, and J. W. M. Bush. 2003. The hydrodynamics of water strider locomotion. *Nature* 424 (2003), 663–666.
L. Huang and D. L. Michels. 2020. Surface-only ferrofluids. *ACM Trans. Graph.* 39 (2020), 6.
L. Huang, Z. Qu, X. Tan, X. Zhang, D. L. Michels, and C. Jiang. 2021. Ships, splashes, and waves on a vast ocean. *ACM Trans. Graph.* 40 (2021), 203.
C. J. Hughes, R. Grzeszczuk, E. Sifakis, D. Kim, S. Kumar, A. P. Selle, J. Chhugani, M. Holliman, and Y. Chen. 2007. Physical simulation for animation and visual effects: parallelization and characterization for chip multiprocessors. *SIGARCH Comput. Archit. News* 35 (2007), 220–231.
C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. 2015. The affine particle-in-cell method. *ACM Trans. Graph.* 34 (2015), 51.
H. Kedia, D. Foster, M. R. Dennis, and W. T. M. Irvine. 2016. Weaving knotted vector fields with tunable helicity. *Phys. Rev. Lett.* 117 (2016), 274501.
E. A. Kuznetsov and A. V. Mikhailov. 1980. On the topological meaning of canonical Clebsch variables. *Phys. Lett. A* 77 (1980), 37–38.
F. Losasso, R. Fedkiw, and S. Osher. 2006. Spatially adaptive techniques for level set methods and incompressible flow. *Comput. Fluids* 35 (2006), 995–1010.
F. Losasso, F. Gibou, and R. Fedkiw. 2004. Simulating water and smoke with an octree data structure. In *ACM Trans. Graph.* 457–462.
F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. 2008. Two-way coupled SPH and particle level set fluid simulation. *IEEE Trans. Vis. Comput. Graph.* 14 (2008), 797–804.
R. W. MacCormack. 2003. The effect of viscosity in hypervelocity impact cratering. *J. Spacecr. Rockets.* 40 (2003), 757–763.
E. Madelung. 1926. Eine anschauliche Deutung der Gleichung von Schrödinger. *Naturwissenschaften* 14 (1926), 1004–1004.
E. Madelung. 1927. Quantentheorie in hydrodynamischer Form. *Z. Phys.* 40 (1927), 322–326.
H. Mazhar, T. Heyn, A. Pazouki, D. Melanz, A. Seidl, A. Bartholomew, A. Tasora, and D. Negrut. 2013. Chrono: a parallel multi-physics library for rigid-body, flexible-body, and fluid dynamics. *Mech. Sci.* 4 (2013), 49–64.
R. I. McLachlan, C. Offen, and B. K. Tapley. 2019. Symplectic integration of PDEs using Clebsch variables. *J. Comput. Dyn.* 6 (2019), 111–130.
S. Mochizuki, K. Suzukawa, K. Saga, and H. Osaka. 2008. Vortex Structures around a Flat Paddle Impeller in a Stirred Vessel. *J. Fluid Sci.* 3 (2008), 241–249.
H. K. Moffatt. 1969. The degree of knottedness of tangled vortex lines. *J. Fluid Mech.* 35 (1969), 117–129.
J. J. Moreau. 1961. Constantes d'un îlot tourbillonnaire en fluide parfait barotrope. *C. R. Acad. Sci. Paris* 252 (1961), 2810–2812.
M. Müller, D. Charypar, and M. H. Gross. 2003. Particle-based fluid simulation for interactive applications.. In *Symposium on Computer Animation.* 154–159.
S. Osher and R. P. Fedkiw. 2001. Level set methods: an overview and some recent results. *J. Comput. Phys.* 169 (2001), 463–502.
S. Osher and J. A. Sethian. 1988. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations. *J. Comput. Phys.* 160 (1988), 151–178.
M. Padilla, A. Chern, F. Knöppel, U. Pinkall, and P. Schröder. 2019. On bubble rings and ink chandeliers. *ACM Trans. Graph.* 38 (2019), 1–14.
L. M. Pismen and L. M Pismen. 1999. *Vortices in nonlinear fields: from liquid crystals to superfluids, from non-equilibrium patterns to cosmic strings.* Vol. 100. Oxford University Press.
Z. Qu, X. Zhang, M. Gao, C. Jiang, and B. Chen. 2019. Efficient and conservative fluids using bidirectional mapping. *ACM Trans. Graph.* 38 (2019), 4.
N. Rasmussen, D. Q. Nguyen, W. Geiger, and R. Fedkiw. 2003. Smoke simulation for large scale phenomena. *ACM Trans. Graph.* 22 (2003), 703–707.
R. Salmon. 1988. Hamiltonian fluid mechanics. *Ann. Rev. Fluid Mech.* 20 (1988), 225–256.
R. Saye. 2016. Interfacial gauge methods for incompressible fluid dynamics. *Sci. Adv.* 2 (2016), e1501869.
A. Selle, N. Rasmussen, and R. Fedkiw. 2005. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph.* 24 (2005), 910–914.
A. L. Sorokin. 2001. Madelung transformation for vortex flows of a perfect liquid. *Dokl. Phys.* 46 (2001), 576–578.
M. Sussman. 2003. A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *J. Comput. Phys.* 187 (2003),

110–136.
R. Tao, H. Ren, Y. Tong, and S. Xiong. 2021. Construction and evolution of knotted vortex tubes in incompressible Schrödinger flow. *Phys. Fluids* 33 (2021), 077112.
G. I. Taylor and A. E. Green. 1937. Mechanism of the production of small eddies from large ones. *Proc. Roy. Soc. Lond.* A 158 (1937), 499–521.
W. Thomson. 1869. On vortex motion. *Trans. R. Soc. Edinburgh* 25 (1869), 217––260.
B. Tings and D. Velotto. 2018. Comparison of ship wake detectability on C-band and X-band SAR. *Int. J. Remote Sens* 39 (2018), 4451–4468.
K. Marten; K. Shariff; S. Psarakos; D. J. White. 1996. Ring bubbles of dolphins. *Sci. Am.* 275 (1996), 82.
J. Z. Wu, H. Y. Ma, and M. D. Zhou. 2006. *Vorticity and Vortex Dynamics.* Springer.
J. Z. Wu, H. Y. Ma, and M. D. Zhou. 2015. *Vortical Flows.* Springer.
S. Xiong, R. TAO, Y. Zhang, F. Feng, and B. ZHU. 2021. Incompressible Flow Simulation on Vortex Segment Clouds. *ACM Trans. Graph.* 40 (2021), 98.
S. Xiong and Y. Yang. 2017. The boundary-constraint method for constructing vortex-surface fields. *J. Comput. Phys.* 339 (2017), 31–45.
S. Xiong and Y. Yang. 2019a. Construction of knotted vortex tubes with the writhe-dependent helicity. *Phys. Fluids* 31 (2019), 047101.
S. Xiong and Y. Yang. 2019b. Identifying the tangle of vortex tubes in homogeneous isotropic turbulence. *J. Fluid Mech.* 874 (2019), 952–978.
S. Xiong and Y. Yang. 2020. Effects of twist on the evolution of knotted magnetic flux tubes. *J. Fluid Mech.* 895 (2020), A28.
S. Yang, S. Xiong, Y. Zhang, F. Feng, J. Liu, and B. Zhu. 2021. Clebsch gauge fluid. *ACM Trans. Graph.* 40 (2021), 99.
Y. Yang and D. I. Pullin. 2010. On Lagrangian and vortex-surface fields for flows with Taylor–Green and Kida–Pelz initial conditions. *J. Fluid Mech.* 661 (2010), 446–481.
Y. Yang and D. I. Pullin. 2011. Evolution of vortex-surface fields in viscous Taylor–Green and Kida–Pelz flows. *J. Fluid Mech.* 685 (2011), 146–164.
X. Zhang and R. Bridson. 2014. A PPPM fast summation method for fluids and beyond. *ACM Trans. Graph.* 33 (2014), 6.
Y. Zhu and R. Bridson. 2005. Animating sand as a fluid. *ACM Trans. Graph.* 24 (2005), 965–972.

## A CONSTRUCTION OF KNOTTED WAVE FUNCTIONS

We use the method from Tao et al. [2021] to construct an initial two-component wave function that can be transformed into a knotted velocity field with finite kinetic energy. First, two complex valued functions are constructed as $\alpha = 2(X + \mathrm{i}Y)f(R)/(1+R^2)$ and $\beta = [2(Z-\mathrm{i})f(R) + (1+R^2)\mathrm{i}]/(1+R^2)$, where $X = \lambda_x(x - x_c)$, $Y = \lambda_y(y-y_c)$, $Z = \lambda_z(z-z_c)$ are scaled and centralized coordinates with scaling numbers $(\lambda_x, \lambda_y, \lambda_z)$ and center coordinates $(x_c, y_c, z_c)$, $R = \sqrt{X^2+Y^2+Z^2}$ is the scaled distance from $(x_c, y_c, z_c)$; and the function $f(R)$ decays monotonically with $f(0) = 1$ and $\lim_{R\to\infty} f(R) = 0$. Then two polynomials $P$ and $Q$ are constructed to represent the twist of the vortex tube as $P = \alpha^k$ with an integer $k$ and the geometric shape of the vortex structures, such as $Q = \alpha^3 + \beta^2$ for trefoil knot, $Q = \alpha^2 - \beta^2$ for two linked rings, respectively [Kedia et al. 2016]. Finally, the wave function $\boldsymbol{\Psi}$ is obtained from the normlized and pressure projected $[P, Q]^T$. Figure 18 plots the isosurfaces of $s_p$, $p = 1, 2, 3$, with different contour values of three linked vortex rings with $P = \alpha^2$ and $Q = \alpha^3 - \beta^3$. The isosurfaces show that different $s_p$ identifies similar vortex structures. Some vortex lines integrated on the surfaces also agree with the constraints of the vortex surface.

## B DERIVATION OF DYNAMICS OF WAVE FUNCTION

From (3), we obtain

$$\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = \hbar\left(\left\langle\boldsymbol{\nabla}\frac{\mathrm{D}\boldsymbol{\Psi}}{\mathrm{D}t}, \mathrm{i}\boldsymbol{\Psi}\right\rangle_{\mathbb{R}} + \left\langle\boldsymbol{\nabla}\boldsymbol{\Psi}, \mathrm{i}\frac{\mathrm{D}\boldsymbol{\Psi}}{\mathrm{D}t}\right\rangle_{\mathbb{R}}\right) - \frac{\boldsymbol{\nabla}|\boldsymbol{u}|^2}{2}. \tag{25}$$

Substituting the first equation of (4) into (25) yields

$$\hbar\left(\left\langle\boldsymbol{\nabla}\frac{\mathrm{D}\boldsymbol{\Psi}}{\mathrm{D}t}, \mathrm{i}\boldsymbol{\Psi}\right\rangle_{\mathbb{R}} + \left\langle\boldsymbol{\nabla}\boldsymbol{\Psi}, \mathrm{i}\frac{\mathrm{D}\boldsymbol{\Psi}}{\mathrm{D}t}\right\rangle_{\mathbb{R}}\right) = -\boldsymbol{\nabla}\left(q - \frac{|\boldsymbol{u}|^2}{2}\right). \tag{26}$$

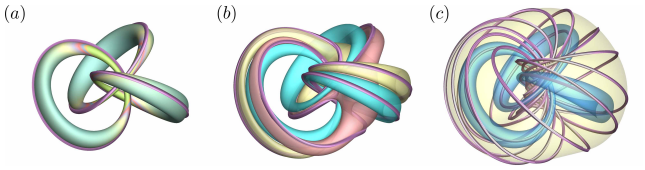


Fig. 18. Linked vortex rings encoded in the level sets of the Clebsch maps. Isosurfaces of (a) $s_1 = 0.6$; (b) $s_1 = 0.6$ (cyan), $s_2 = -0.9$ (salmon), and $s_3 = -0.85$ (yellow); (c) $s_1 = 0.6$ and $-0.7$. Some vortex lines are integrated on the isosurfaces.

We restrict that the moduli of the two components of the wave function remain constant during the evolution. In this case, we can assume that the evolution of the wave function is

$$\frac{\mathrm{D}\boldsymbol{\Psi}}{\mathrm{D}t} = \frac{-\mathrm{i}}{\hbar} f_\Psi \boldsymbol{\Psi} \tag{27}$$

to satisfy the normalization condition. Here $f_\Psi$ is a real valued function. Substituting (27) into (26) yields

$$\langle\boldsymbol{\nabla}(f_\Psi\boldsymbol{\Psi}), \boldsymbol{\Psi}\rangle_{\mathbb{R}} + \langle\boldsymbol{\nabla}\boldsymbol{\Psi}, (f_\Psi\boldsymbol{\Psi})\rangle_{\mathbb{R}} = \boldsymbol{\nabla}\left(q - \frac{|\boldsymbol{u}|^2}{2}\right). \tag{28}$$

Using (1), (28) can be rewritten as

$$\boldsymbol{\nabla} f_\Psi = \boldsymbol{\nabla}\left(q - \frac{|\boldsymbol{u}|^2}{2}\right). \tag{29}$$

Disregarding the global phase yields $f_\Psi = q - |\boldsymbol{u}|^2/2$, *i.e.*, (27) and (5) are equivalent. In particular, the pure advection of the velocity with $q = 0$ and (6) are equivalent.

## C BUBBLE FLOW ALGORITHM

We built our bubble volume tracking algorithm by following the previous practice in bubble flow simulations [Aanjaneya et al. 2013; Sussman 2003]. The core idea is to flood-fill each bubble on the grid cells and track its volume and topological changes with a semi-Lagrangian backtrack. Specifically, we solve the bubble flow algorithm in the following steps. First, we store a target volume for each bubble during initialization. In each time step, We mark each bubble as $i = 1, 2, \cdots$ by using breadth-first search method, then we update the target volume $V_{\mathrm{tar}}^{(i)}$ for each bubble using backtracking method. Combining the current volume $V^{(i)}$ and target volume $V_{\mathrm{tar}}^{(i)}$ of the $i$-th bubble, we define the state equation as

$$p_{\mathrm{air}}^{(i)} = \mathrm{atan}\left[\left(\frac{V_{\mathrm{tar}}^{(i)}}{V^{(i)}}\right)^2 - \left(\frac{V^{(i)}}{V_{\mathrm{tar}}^{(i)}}\right)^2\right]. \tag{30}$$

We remark that the equation (30) is designed to reduce and increase the gas pressure as the bubble volume becomes larger and smaller with respect to the target volume, respectively, and changing the specific form of (30) may yield similar results.