DOI: 10.1111/cgf.15023
EUROGRAPHICS 2024 / A. Bermano and E. Kalogerakis
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 2

Wavelet Potentials: An Efficient Potential Recovery Technique for Pointwise Incompressible Fluids

Luan Lyu*[1,2], Xiaohua Ren*[3], Wei Cao[4], Jian Zhu[5], Enhua Wu†[2,6] Zhi-Xin Yang†[1,2]

[1]SKL-IOTSC & ICI-CAR, Univ. of Macau, [2]FST, Univ. of Macau, [3]Tencent, [4]College of CST, China Univ. of Petroleum, [5]Guangdong Univ. of Technology, [6]State Key Lab. of CS, ISCAS & Univ. of CAS

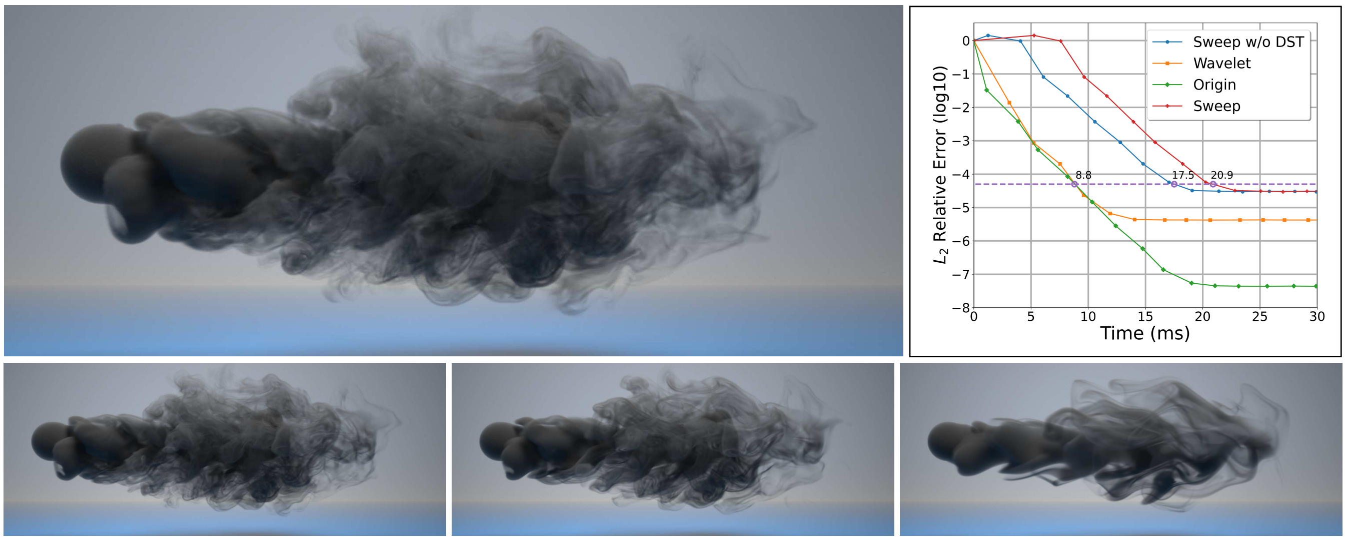


Figure 1: *A multiresolution smoke simulation of size $2^8 \times 2^7 \times 2^7$ using our wavelet potential recovery method: The top-left image shows the unfiltered smoke, while the bottom-left, bottom-center, and bottom-right images show the results of filtering the top 2, 3, and 4 levels of wavelet coeffs., respectively. The top-right image shows the convergence of different methods. Our method achieves comparable performance to the direct method when targeting an accuracy threshold of 5×10^{-5}, while also providing a* **2x** *speedup over the sweeping methods.*

Abstract

We introduce an efficient technique for recovering the vector potential in wavelet space to simulate pointwise incompressible fluids. This technique ensures that fluid velocities remain divergence-free at any point within the fluid domain and preserves local volume during the simulation. Divergence-free wavelets are utilized to calculate the wavelet coefficients of the vector potential, resulting in a smooth vector potential with enhanced accuracy, even when the input velocities exhibit some degree of divergence. This enhanced accuracy eliminates the need for additional computational time to achieve a specific accuracy threshold, as fewer iterations are required for the pressure Poisson solver. Additionally, in 3D, since the wavelet transform is taken in-place, only the memory for storing the vector potential is required. These two features make the method remarkably efficient for recovering vector potential for fluid simulation. Furthermore, the method can handle various boundary conditions during the wavelet transform, making it adaptable for simulating fluids with Neumann and Dirichlet boundary conditions. Our approach is highly parallelizable and features a time complexity of $O(n)$, allowing for seamless deployment on GPUs and yielding remarkable computational efficiency. Experiments demonstrate that, taking into account the time consumed by the pressure Poisson solver, the method achieves an approximate **2x** *speedup on GPUs compared to state-of-the-art vector potential recovery techniques while maintaining a precision level of 10^{-6} when single float precision is employed. The source code of 'Wavelet Potentials' can be found in* `https://github.com/yours321dog/WaveletPotentials`.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ *Physical simulation;*

1. Introduction

Incompressible fluid simulation has been widely studied in computer graphics. Recent years have brought about a growing inter-

* Co-first authors; equal contribution.
† Corresponding authors (ehwu@um.edu.mo & zxyang@um.edu.mo)

est in hybrid methodologies that blend Eulerian and Lagrangian simulation techniques. Such approaches, exemplified by works like [Sta99], [ZB05], and [JSS*15], take the advantages of both paradigms. This fusion enables the incorporation of Eulerian techniques, which excel at interior force computation, alongside Lagrangian methods that proficiently advect fluid properties. The synergy of these techniques results in fluid simulations that are not only robust and efficient but also exude vivid realism.

Nonetheless, the methods mentioned above do not assure incompressibility at arbitrary points, as their achievement of the divergence-free condition hinges on summing the differences in velocities across each direction within a grid cell. When computing velocities at a specific point, these approaches often resort to direct velocity interpolation, such as the basic polynomial interpolation method. These direct interpolation techniques yield velocities that do not adhere to the divergence-free constraint at specific points. To address the problem, some researchers, such as [BHN07] and [CPAB22], have adopted an approach where velocities are interpolated from the curl of a secondary vector field, which is a scalar stream function field in 2D and a vector potential field in 3D. Due to the inherent property that the divergence of the curl operator is always zero, this type of interpolation intrinsically ensures a divergence-free constraint at arbitrary points within the fluid domain.

To perform the curl operator of the vector potential, the generation of the vector potential becomes essential. Leveraging the principles of the Helmholtz decomposition, some researchers, such as [ATW15] and [BDG*17], solve a time-consuming vector Poisson equation to derive the vector potential. In 2D, the vector potential essentially becomes a scalar stream function. Thus, solving the Poisson equation to attain the stream function is relatively efficient, comparable to solving the pressure Poisson equation within the projection step of fluid simulation. However, the vector potential encompasses three components in 3D. Solving the vector Poisson equation in this case turns out to be extremely computationally intensive. To address this problem, Chang et al. [CPAB22] proposed a parallel sweeping method aimed at recovering the vector potential from divergence-free velocities, which are derived from a scalar Poisson solver. Compared to the resource-intensive vector Poisson solver, the parallel sweeping technique offers notable computational efficiency.

Although the parallel sweeping method is known for its computational efficiency, it still faces issues with non-smoothing and reduced accuracy. To address the first issue, Chang et al. [CPAB22] implemented an additional scalar Poisson solver for the non-smooth vector potential, which consequently adds extra computational overhead. As for the second challenge, Chang et al. [CPAB22] demonstrate the validity of the parallel sweeping method, assuming highly divergence-free input velocities. However, real-world simulations often require stopping the projection step once a specific accuracy level is reached to optimize simulation time. As a result, the velocities maintain a certain degree of divergence. In such cases, the parallel sweeping method undergoes a considerable decline in accuracy. Our experiments reveal that the accuracy decreases to 10^{-3} for a velocity with an accuracy of 10^{-5}.

To recover the smooth vector potential efficiently with high accuracy, we propose a wavelet-based method based on Lematrié Rieusset's proposition [Lem92]. Our method involves the transformation of divergence-free velocities using divergence-free wavelets, subsequently employing an approximate projection technique to derive the vector potential's wavelet coefficients. Finally, inverse wavelet transforms are applied to recover the vector potential. Wavelet-based techniques have been extensively explored in fluid simulation for many years, as evident in works like [DP06], [HP15], and more recently, [Les19]. Many of these wavelet-based methods like [HP15] involve solving a system of linear equations within the wavelet space to determine the coefficients of divergence-free wavelets. While these methods yield vector potentials with high accuracy, they conflict with our core requirement to eliminate resource-intensive Poisson solvers. Other approaches, exemplified by [DP06], employ an approximate projection to calculate wavelet coefficients, with exact results achieved through iterations. However, the approximate projection proposed by Deriaz and Perrier [DP06] fails to achieve the exact vector potential even when the input velocities are divergence-free in 3D cases. To address this limitation, we propose a novel projection method that can recover the exact vector potential for the divergence-free velocities.

In summary, we present a wavelet-based vector potential recovery method that can robustly and efficiently recover the smooth vector potential from divergence-free velocities. Notably, our approach excels even in scenarios where input velocities retain residual divergence, resulting in vector potentials with significantly reduced errors. Due to the improved accuracy, our method does not require any additional time for fluid simulation to achieve the same level of accuracy, as we can use fewer iterations for the pressure Poisson solver. The key contributions are summarized as follows:

- We establish a comprehensive wavelet framework for recovering a smooth vector potential from divergence-free velocities, leveraging the power of divergence-free wavelets;
- We introduce a novel and robust side-to-side approximate projection method, facilitating the derivation of vector potential coefficients within the wavelet space;
- Our method is efficient for recovering the vector potential for fluid simulation when targeting a specific accuracy threshold. Moreover, the multiresolution representation of the restored potential allows us to simulate smoke with varying levels of detail;
- Both the wavelet transformation and the projection method can be executed in parallel and optimized for GPU computation, thereby ensuring high-performance outcomes.

2. Related Work

In this section, the related works are described in three aspects: pointwise incompressible fluid simulation, vector potential recovery and wavelet theory.

2.1. Pointwise Incompressible Fields

Several notable research efforts have emerged over the last few decades to generate pointwise incompressible vector fields. DeWolf [DeW05] pioneered a method for generating divergence-free vector fields, accomplishing this by the cross product of the gradient of two scalar noise functions. Bridson et al.[BHN07] introduced a straightforward technique for creating turbulent divergence-free velocity fields through the curl of the vector potential, while also implementing a ramping method to address behaviors around internal solids. Drawing inspiration from their work, Kim et al.[KTJG08] leveraged divergence-free vector fields constructed by curling the enhanced wavelet noise function for detailed fluid animations. Similarly, Schechter and Bridson [SB08] incorporated a rudimentary turbulence model to amplify sub-grid details within smoke animations. The pivotal component of this turbulence model was the curl of the vector potential derived from designated noise functions.

More recently, Chang et al.[CPAB22] devised a pointwise incompressible interpolation strategy, which generates fluid particle velocities by leveraging the curl of the vector potential. They also introduced a constraint correction technique to refine fluid behaviors around interior obstacles in three dimensions. Building upon the concept of the LogSumExp distance function, Ding and Batty[DB23] pioneered a differentiable curl-noise method which further corrects the vector potential near obstacles, ultimately achieving strictly incompressible fluid flows in two dimensions.

2.2. Vector Potential Recovery

In contrast to the conventional velocity formulation of the Navier-Stokes equation, Elcott et al.[ETK*07] pursued a vorticity-based formulation to facilitate fluid simulation. They recovered the vector potential from the vorticity field by solving a computationally expensive vector Poisson equation. Ando et al. [ATW15] proposed a stream function approach for complicated liquid-solid and liquid-air coupling scenarios. Sato et al.[SDY*15] employed the vector potential to modulate the flow field, enabling the manipulation of the overall fluid behavior. In subsequent work, Sato et al.[SDK21] harnessed the vector potential to guide fluid simulation, thereby enhancing fluid effects. It's worth noting that all of these approaches necessitated the solution of a resource-intensive vector Poisson equation to attain the desired vector potential.

To efficiently obtain the vector potential, Chang et al.[CPAB22] introduced a parallel sweeping method for its recovery. Their approach draws inspiration from the work of Biswas et al.[BSW*16], who devised a technique for recovering the stream function from divergence-free velocities to construct streamlines in two dimensions.

2.3. Divergence-Free Wavelet

Lematrié-Rieusset's proposition [Lem92] has led to the proposal of several divergence-free wavelet construction methods for various complex situations in recent decades. These methods are based on the curl of the vector potential and imply a relationship between the vector potential and the divergence-free field in the wavelet space. Urban [Urb01] described how to construct the curl-free and divergence-free vector wavelets. Although curl-free and divergence-free vector wavelets are two different kinds of wavelets, they share a very similar construction process. Stevenson [Ste11; Ste16] constructed the divergence-free wavelets on the n-dimensional hypercube, addressing general homogeneous Dirichlet boundary conditions. Harouna and Perrier [HP13] proposed an efficient construction of divergence-free wavelets on the square with the help of fast divergence-free wavelet transforms. This approach was recently extended for homogeneous Dirichlet wavelets in [HP22].

Urban [Urb96] employed divergence-free wavelets to address the Stokes problem. When the input velocities are not strictly divergence-free, the relationship between the coefficients of velocities and vector potentials in wavelet space becomes intricate. A system of linear equations is required, constructed through the inversion of divergence-free and curl-free wavelet Gram matrices, to obtain the coefficients of these wavelets. Zhou et al.[ZH05] extended Urban's approach for the numerical solution of the 2-D stationary Navier–Stokes equations. Harouna and Perrier[HP12] harnessed optimal preconditioning techniques to enhance the efficiency of solving the systems. Subsequently, they [HP21; HP15] further extended their methods to simulate incompressible viscous flows encompassing more intricate boundary conditions.

Instead of tackling a system of linear equations, Deriaz and Perrier [DP06] carried out the Helmholtz-Hodge decomposition by repeatedly approximating the coefficients of curl-free and divergence-free wavelets using simple approximate projection methods. In a subsequent endeavor [DP09], they extended this method to arbitrary dimensions, presenting a more precise approximate projection technique to determine the coefficients of divergence-free wavelets in higher dimensions.

In computer graphics, Ren et al. [RLH*17] introduced an approximate curl-free wavelet projection inspired by the work of Deriaz and Perrier [DP06]. This approach achieved state-of-the-art results in gradient-domain compositing compared to multigrid methods on both CPUs and GPUs. Later, Ren et al. [RLH*18] extended this method to three dimensions and constructed a new family of wavelets utilizing Lematrié Rieusset's proposition [Lem92] for 3D surface reconstruction. Their method addressed issues with previous wavelet techniques that were not resilient to missing or nonuniformly sampled data, while also inheriting the feature of a friendly streaming implementation to process very large datasets.

3. Background

3.1. Pointwise Incompressible Fluids

The pointwise incompressible fluid simulation is based on the incompressible Euler equations:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla p + \mathbf{f}, \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{2}$$

where $\mathbf{u}$ and p are the velocity and pressure of the fluid, respectively, and $\mathbf{f}$ accounts for external forces. We have arbitrarily set the constant density of the fluid to one.

Based on the governing equations, Alg. 1 presents the pointwise incompressible fluid simulation algorithm. The key distinction between pointwise incompressible fluid simulation and traditional hybrid fluid simulation, such as [Sta99; ZB05; JSS*15], lies in the

Algorithm 1: Pointwise incompressible fluid simulation

```
  Input: $\mathbf{f}$                      // external forces
  Input: $n_{sim}$                         // step number of simulation
1 for $i = 0 : (n_{sim} - 1)$ do
2     Add external forces $\mathbf{f} \Rightarrow \bar{\mathbf{u}}^i$
3     Project $\bar{\mathbf{u}}^i$ via the pressure Poisson solver $\Rightarrow \mathbf{u}^i$
4     Recover the vector potential from $\mathbf{u}^i \Rightarrow \mathbf{q}$
5     Calculate the vector potential of particles at position $\mathbf{x}_p$
        using linear or cubic interpolation of $\mathbf{q} \Rightarrow \mathbf{q}(\mathbf{x}_p)$
6     Advect particles via $\nabla \times \mathbf{q}(\mathbf{x}_p)$
7     Advect the velocity on the grid $\mathbf{u}^i \Rightarrow \mathbf{u}^{i+1}$
```

velocity interpolation method from the grid to particles. The traditional method for determining particle velocity involves linear or cubic interpolation of the velocity $\mathbf{u}$. In contrast, pointwise incompressible interpolation consists of three steps (lines 4 $\sim$ 6 in Alg. 1): first, recovering the vector potential from $\mathbf{u}$; second, performing linear or cubic interpolation of the potential to obtain the potential of particles; third, deriving the velocity of particles by taking the curl of the particle potential.

Among the three steps of the pointwise incompressible interpolation, the first step - recovering the vector potential from the divergence-free velocity - is considered the most computationally expensive. A common solution is to solve a vector Poisson equation, as demonstrated in [ATW15; BDG*17], but this approach is costly. Chang et al.[CPAB22] propose a more efficient method that initially employs the parallel sweeping method to recover an approximate vector potential and then solves an additional scalar Poisson equation to smooth the potential. Although the fast discrete sine transform (DST) is utilized for the additional scalar Poisson equation, the DST has a time complexity of $O(n\log n)$. Another limitation is that the parallel sweeping method relies on the assumption that the input velocity is highly divergence-free. Otherwise, the accuracy would significantly decrease (refer to Sec. 5.1 for convergence analysis).

Drawing inspiration from the curl-free wavelet projection methods introduced by Ren et al. [RLH*17; RLH*18] for applications in gradient-domain compositing and 3D surface reconstruction, we propose a robust and efficient divergence-free wavelet projection method for vector potential recovery. Our method has a time complexity of $O(n)$ and can improve the accuracy when the input velocity exhibits a certain degree of divergence. Before delving into the method, let's introduce some concepts in wavelet theory.

3.2. Wavelet Theory

In this section, a brief overview of wavelet theory is provided. Readers seeking a more in-depth understanding are encouraged to consult the comprehensive literature provided in [Mal08].

Multiresolution Analysis (MRA). A sequence of closed subspaces $\{V_j = \text{span}\{\phi(2^j t - k) : k \in \mathbb{Z}\} : j \in \mathbb{Z}\}$ is called a multiresolution analysis of $L^2(\mathbb{R})$ if they satisfy the following requirement:

$$\{0\} \subset \cdots \subset V_{j-1} \subset V_j \subset V_{j+1} \subset \cdots \subset L^2(\mathbb{R}),$$

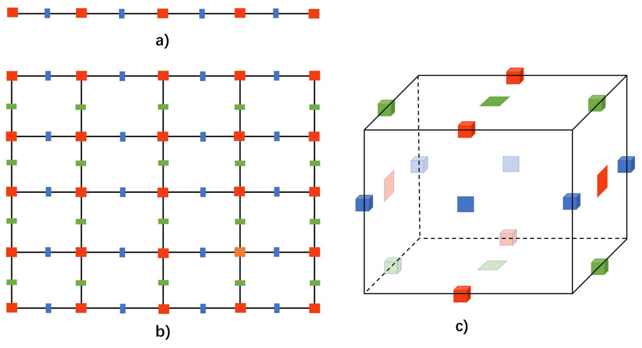


Figure 2: *Discretization in a) 1D: the potential $q(t)$ is sampled on segment vertices (red squares) and the velocity $u(t)$ is sampled on segment centers (blue rectangles); b) 2D: the potential is sampled on the cell vertices (red squares) and the velocity is sampled on cell edges (rectangles). Green and blue rectangles represent the x and y components of the velocity, respectively; c) 3D: the vector potential is sampled on the cell edges (boxes) and the velocity is sampled on the cell faces (squares). Red, green and blue represent the x, y and z components, respectively.*

where $L^2(\mathbb{R})$ is the vector space of all functions $f(t) : \mathbb{R} \to \mathbb{R}$ of finite energy and $\phi(t) : \mathbb{R} \to \mathbb{R}$ is referred to as a *scaling* function.

Wavelet Space. Wavelet spaces $W_j = \text{span}\{\psi(2^j t - k) : k \in \mathbb{Z}\}$ are constructed to be the complements such that $V_{j+1} = V_j \oplus W_j$. Here, $\psi(t) : \mathbb{R} \to \mathbb{R}$ is referred to as a *wavelet*.

We now present Lemarié-Rieusset's proposition [Lem92], which connects two MRAs through differentiation and serves as the foundation for our method. For more details, please refer to [DP09; RLH*17; RLH*18].

Lemarié-Rieusset's Proposition. Let $\{V_j^1\}$ be a MRA of $L^2(\mathbb{R})$ with associated scaling function and wavelet $(\phi^1(t), \psi^1(t))$. Then, there exists another MRA $\{V_j^0\}$ of $L^2(\mathbb{R})$ with associated scaling function and wavelet $(\phi^0(t), \psi^0(t))$, satisfying

$$(\phi^1)'(t) = \phi^0(t+1) - \phi^0(t), \tag{3}$$

$$(\psi^1)'(t) = 4 \cdot \psi^0(t), \tag{4}$$

4. Our Approach

This section introduces our wavelet approach for recovering the vector potential from divergence-free velocities. To make it easier to understand, we begin with one dimension before extending to two dimensions and three dimensions.

4.1. 1D Wavelet Potential Recovery

Problem Setting. A one-dimensional potential recovery problem can be viewed as a one-dimensional integration problem. Consider the following one-dimensional integration problem defined on the unit interval $[0,1]$ with periodic boundary conditions:

$$q'(t) = u(t), \quad t \in [0,1], \qquad (5)$$
$$q(0) = q(1).$$

Discretization. To solve the problem numerically, we discretize Eq. 5 using the staggered grid scheme. As illustrated in Fig. 2a), we evenly partition the domain $[0,1]$ into 2^n parts with the size of step $h = 1/2^n$, and then sample $q_k^n = q(kh)$ of $q(t)$ and $u_k^n = u(kh+h/2)$ of $f(t)$, respectively, for each $k \in \{0,1,\cdots,2^n-1\}$. Here, n is a positive integer number. The superscript n in q_k^n and u_k^n denotes the values are sampled at level n.

The goal of a numerical integration method is to solve q_k^n when provided with the values of u_k^n. In one dimension, the parallel sweeping method [CPAB22] reduces to the Euler method. Therefore, in order to establish a link between our wavelet method and the parallel sweeping method [CPAB22], we revisit the derivation of the Euler method using Eq. 3 from Lematrié-Rieusset's Proposition.

Euler Method. Let $\phi^1(t)$ and $\phi^0(t)$ be two functions that satisfy Eq.3. We can approximate the function $q(t)$ and $u(t)$ as follows:

$$q^n(t) = \sum_k q_k^n \phi^1(2^n t - k), \qquad (6)$$

$$u^n(t) = \sum_k u_k^n \phi^0(2^n t - k). \qquad (7)$$

Taking the derivative of $q^n(t)$ and using Eqs. 3 and 5, we obtain

$$\sum_k 2^n \cdot (q_{k+1}^n - q_k^n)\phi^0(2^n t - k) = \sum_k u_k^n \phi^0(2^n t - k)$$
$$\Rightarrow q_{k+1}^n = q_k^n + h u_k^n. \qquad (8)$$

Eq. 8 is nothing more than the Euler method, which also happens to be the parallel sweeping method [CPAB22] for one dimension.

Wavelet Method. Unlike the Euler method, which performs integration in the spatial domain, the proposed method operates in the wavelet domain. More specifically, the method calculates the wavelet coefficients of $q^n(t)$ from those of $u^n(t)$. To simplify the explanation, we will illustrate the relationship after a one-level wavelet transform.

First, we apply a one-level wavelet transform to $q^n(t)$ and $u^n(t)$ individually, using appropriate wavelets $\psi^1(t)$ and $\psi^0(t)$ that satisfy Eq. 4 and resulting in

$$q^n(t) = q^{n-1}(t) + \sum_k \tilde{q}_k^{n-1} \psi^1(2^{n-1}t - k), \qquad (9)$$

$$u^n(t) = u^{n-1}(t) + \sum_k \tilde{u}_k^{n-1} \psi^0(2^{n-1}t - k), \qquad (10)$$

$$q^{n-1}(t) = \sum_k q_k^{n-1} \phi^1(2^{n-1}t - k),$$

$$u^{n-1}(t) = \sum_k u_k^{n-1} \phi^0(2^{n-1}t - k)$$

where $k \in \{0,1,\cdots,2^{n-1}-1\}$. q_k^{n-1} and u_k^{n-1} are referred to the approximate coefficients at level $n-1$. $\tilde{q}_k^{n-1}$ and $\tilde{u}_k^{n-1}$ are known as the wavelet (or detail) coefficients at level $n-1$.

Second, we take the derivative of $q^n(t)$ and use Eqs. 3, 4 and 5

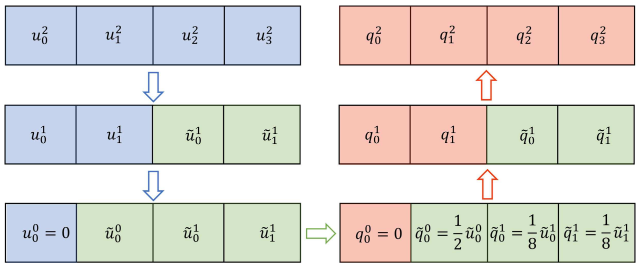


Figure 3: *1D wavelet potential recovery on a grid of size 2^2: First, a full-level wavelet transform is applied to the velocity u^2 to obtain its wavelet coefficients. Second, the wavelet coefficients $\tilde{q}^j$ of q^2 are extracted using Eq.15. Finally, q^2 is recovered by applying a full-level inverse wavelet transform to $\tilde{q}^j$.*

to obtain the following results:

$$\sum_k 2^{n-1} \cdot (q_{k+1}^{n-1} - q_k^{n-1})\phi^0(2^{n-1}t - k) = \sum_k u_k^{n-1}\phi^0(2^{n-1}t - k)$$
$$\Rightarrow q_{k+1}^{n-1} = q_k^{n-1} + 2h u_k^{n-1}, \qquad (11)$$

$$\sum_k 4 \cdot 2^{n-1} \cdot \tilde{y}_k^{n-1} \psi^0(2^{n-1}t - k) = \sum_k \tilde{f}_k^{n-1} \psi^0(2^{n-1}t - k)$$
$$\Rightarrow \tilde{q}_k^{n-1} = \frac{1}{4} \cdot 2h\tilde{u}_k^{n-1}. \qquad (12)$$

Eq. 11 presents the outcome of the Euler method, but this time on the coarser grid of level $n-1$. Eq. 12 is the crucial finding, revealing that the wavelet coefficients of the unknown $q^n(t)$ at level $n-1$ are equal to those of $u^n(t)$, with the only difference being a constant factor.

We can now perform a full-level wavelet transform to $q^n(t)$ and $u^n(t)$, yielding

$$q^n(t) = q_0^0 \phi_c^1(x) + \sum_j \sum_k \tilde{q}_k^j \psi^1(2^j t - k), \qquad (13)$$

$$u^n(t) = u_0^0 \phi_c^0(x) + \sum_j \sum_k \tilde{u}_k^j \psi^0(2^j t - k), \qquad (14)$$

where $k \in \{0,1,\cdots,2^j-1\}$ for each $j \in \{n-1,\cdots,0\}$. Taking the derivative of $q^n(t)$, we can obtain

$$\tilde{q}_k^j = \frac{1}{4} \cdot 2^{-j} \cdot \tilde{u}_k^j, \qquad (15)$$

As Eq. 5 has an unconstrained mean value for the periodic boundary conditions, ϕ_c^1 is a constant function and $\phi_c^0 = (\phi_c^1)' = 0$. We set the coarsest undetermined approximate coefficient q_0^0 to zero. Finally, q^n is recovered by applying a full-level inverse wavelet transform to the wavelet coefficients $\tilde{q}^j$. Fig. 3 illustrates the entire process of the 1D wavelet potential recovery on a grid of size 2^2.

Although the wavelet method may appear more complicated than the Euler method, it has the advantage of being able to perform integration for higher dimensions in a robust manner. In the following sections, we extend it to two and three dimensions, demonstrating its effectiveness for recovering vector potentials in these higher-dimensional spaces.

4.2. 2D Wavelet Potential Recovery

Problem Setting. The 2D vector potential recovery problem aims to find a vector potential that reduces to a scalar stream function $q(x,y)$ in 2D and satisfies the perpendicular gradient equation, given a divergence-free vector field $\mathbf{u}(x,y) = [u(x,y), v(x,y)]^T$. Specifically, the problem on the unit square $[0,1]^2$ with periodic boundary conditions is defined as follows:

$$\begin{aligned} \nabla^{\perp} q(x,y) &= \mathbf{u}(x,y), \quad [x,y]^T \in [0,1]^2, \\ q(0,y) &= q(1,y), \\ q(x,0) &= q(x,1). \end{aligned} \tag{16}$$

Here, the perpendicular operator $\nabla^{\perp} = [\frac{\partial}{\partial y}, -\frac{\partial}{\partial x}]^T$ is a 90° rotation of the gradient operator.

Parallel Sweeping Method [CPAB22]. Eq. 16 can be expressed as

$$\frac{\partial q}{\partial y}(x,y) = u(x,y), \quad -\frac{\partial q}{\partial x}(x,y) = v(x,y). \tag{17}$$

The fundamental concept behind the parallel sweeping method involves using the one-dimensional Euler method as a building block and separately evaluating the two integration problems in Eq. 17. There are two approaches to implement the method.

The first approach involves calculating the one-dimensional integration $\frac{\partial q}{\partial y}(0,y) = u(0,y)$ to obtain $q(0,y)$. Then, for each fixed y, the one-dimensional integration $\frac{\partial q}{\partial x}(x,y) = -v(x,y)$ is performed using $q(0,y)$ as the initial value. Both steps utilize the Euler method described in Sec.4.1, and the second step is executed in parallel.

Alternatively, the method can begin by computing $\frac{\partial q}{\partial x}(x,0) = -v(x,0)$ to obtain $q(x,0)$, and then integrate $\frac{\partial q}{\partial y}(x,y) = u(x,y)$ for each fixed x using $q(x,0)$ as the initial value.

The parallel sweeping method is highly efficient, as it requires only n^2 floating-point operations on a grid of size n^2. However, if the input velocity $\mathbf{u}$ is not highly divergence-free, the accuracy of the method is significantly compromised. Moreover, in such cases, the two approaches of the method are not equivalent, resulting in ambiguity.

Next, we will introduce a wavelet method that uniquely computes $q(x,y)$ using the wavelet coefficients of both $u(x,y)$ and $v(x,y)$.

Discretization. Similar to the one-dimensional case, we employ the staggered grid scheme to discretize Eq.16. As shown in Fig. 2b), the computational domain $[0,1]^2$ is divided into identical cells with a cell size of $h = 1/2^n$. We then sample $q^{\mathbf{n}}_{\mathbf{k}} = q(k_1 h, k_2 h)$, $u^{\mathbf{n}}_{\mathbf{k}} = u(k_1 h, k_2 h + h/2)$ and $v^{\mathbf{n}}_{\mathbf{k}} = v(k_1 h + h/2, k_2 h)$ for each $\mathbf{k} = [k_1, k_2]^T \in \{0, 1, \cdots, 2^n - 1\}^2$. The superscript $\mathbf{n}$ in $q^{\mathbf{n}}_{\mathbf{k}}$, $u^{\mathbf{n}}_{\mathbf{k}}$ and $v^{\mathbf{n}}_{\mathbf{k}}$ denotes the values are sampled at level $\mathbf{n} = [n,n]$.

Wavelet Method. We approximate $q^{\mathbf{n}}(x,y)$ and $\mathbf{u}^{\mathbf{n}}(x,y)$ as follows:

$$\begin{aligned} q^{\mathbf{n}}(x,y) &= \sum_{\mathbf{k}} q^{\mathbf{n}}_{\mathbf{k}} \phi^1(2^n x - k_1)\phi^1(2^n y - k_2), \\ u^{\mathbf{n}}(x,y) &= \sum_{\mathbf{k}} u^{\mathbf{n}}_{\mathbf{k}} \phi^1(2^n x - k_1)\phi^0(2^n y - k_2), \\ v^{\mathbf{n}}(x,y) &= \sum_{\mathbf{k}} v^{\mathbf{n}}_{\mathbf{k}} \phi^0(2^n x - k_1)\phi^1(2^n y - k_2). \end{aligned}$$

We perform a full-level wavelet transform to $q^{\mathbf{n}}(x,y)$, $u^{\mathbf{n}}(x,y)$ and $v^{\mathbf{n}}(x,y)$ individually, resulting in

$$\begin{aligned} q^{\mathbf{n}}(x,y) = {} & q^{\mathbf{0}}_{\mathbf{0}} \phi^1_c(x)\phi^1_c(y) \\ & + \sum_{j_1} \tilde{q}^{j_1}_{k_1} \psi^1(2^{j_1}x - k_1)\phi^1_c(y) \\ & + \sum_{j_2} \tilde{q}^{j_2}_{k_2} \phi^1_c(x)\psi^1(2^{j_2}y - k_2) \\ & + \sum_{\mathbf{j}}\sum_{\mathbf{k}} \tilde{q}^{\mathbf{j}}_{\mathbf{k}} \psi^1(2^{j_1}x - k_1)\psi^1(2^{j_2}y - k_2), \end{aligned} \tag{18}$$

$$\begin{aligned} u^{\mathbf{n}}(x,y) = {} & u^{\mathbf{0}}_{\mathbf{0}} \phi^1_c(x)\phi^0_c(y) \\ & + \sum_{k_1} \tilde{u}^{j_1}_{k_1} \psi^1(2^{j_1}x - k_1)\phi^0_c(y) \\ & + \sum_{k_2} \tilde{u}^{j_2}_{k_2} \phi^1_c(x)\psi^0(2^{j_2}y - k_2) \\ & + \sum_{\mathbf{j}}\sum_{\mathbf{k}} \tilde{u}^{\mathbf{j}}_{\mathbf{k}} \psi^1(2^{j_1}x - k_1)\psi^0(2^{j_2}y - k_2), \end{aligned} \tag{19}$$

$$\begin{aligned} v^{\mathbf{n}}(x,y) = {} & v^{\mathbf{0}}_{\mathbf{0}} \phi^0_c(x)\phi^1_c(y) \\ & + \sum_{k_1} \tilde{v}^{j_1}_{k_1} \psi^0(2^{j_1}x - k_1)\phi^1_c(y) \\ & + \sum_{k_2} \tilde{v}^{j_2}_{k_2} \phi^0_c(x)\psi^1(2^{j_2}y - k_2) \\ & + \sum_{\mathbf{j}}\sum_{\mathbf{k}} \tilde{v}^{\mathbf{j}}_{\mathbf{k}} \psi^0(2^{j_1}x - k_1)\psi^1(2^{j_2}y - k_2), \end{aligned} \tag{20}$$

where $\mathbf{k} = [k_1, k_2]^T \in \{0, \cdots, 2^{j_1} - 1\} \times \{0, \cdots, 2^{j_2} - 1\}$ for $\mathbf{j} = [j_1, j_2] \in \{n-1, \cdots, 0\}^2$. Our goal is to compute the wavelet coefficients $\tilde{q}^{j_1}$, $\tilde{q}^{j_2}$ and $\tilde{q}^{\mathbf{j}}$ using the wavelet coefficients of $\mathbf{u}^{\mathbf{n}}$ and $\mathbf{v}^{\mathbf{n}}$. By applying the perpendicular gradient operator to $q^{\mathbf{n}}(x,y)$ and considering that $\phi^0_c(x) = (\phi^1_c(x))' = 0$ and $\phi^0_c(y) = (\phi^1_c(y))' = 0$ for the periodic boundary conditions, we obtain the following results:

$$\tilde{q}^{j_2}_{k_2} = +\frac{1}{4} \cdot 2^{-j_2} \cdot \tilde{u}^{j_2}_{k_2}, \quad \tilde{q}^{j_1}_{k_1} = -\frac{1}{4} \cdot 2^{-j_1} \cdot \tilde{v}^{j_1}_{k_1} \tag{21}$$

$$\tilde{q}^{\mathbf{j}}_{\mathbf{k}} \begin{bmatrix} +4 \cdot 2^{j_2} \\ -4 \cdot 2^{j_1} \end{bmatrix} = \begin{bmatrix} u^{\mathbf{j}}_{\mathbf{k}} \\ v^{\mathbf{j}}_{\mathbf{k}} \end{bmatrix} \tag{22}$$

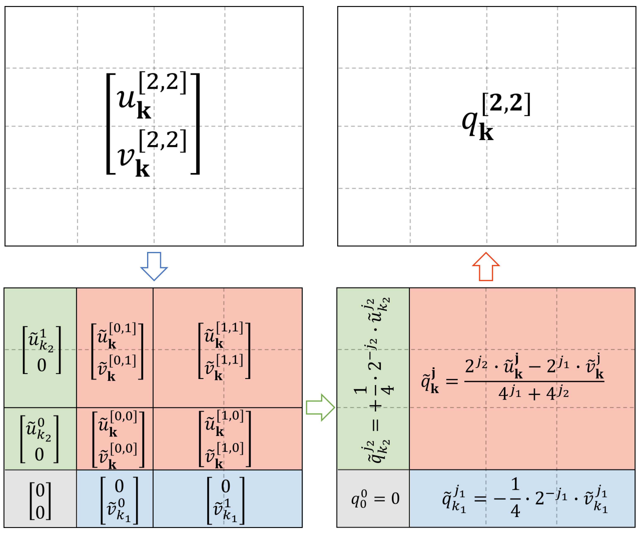
Remark 1: In Eq. 22, if the input velocity $\mathbf{u}^n$ is divergence-free, then we can compute $\tilde{q}^{\mathbf{j}}_{\mathbf{k}} = +\frac{1}{4} \cdot u^{\mathbf{j}}_{\mathbf{k}} = -\frac{1}{4} \cdot v^{\mathbf{j}}_{\mathbf{k}}$. However, if the input velocity is not divergence-free, it is unclear how to determine $\tilde{q}^{\mathbf{j}}_{\mathbf{k}}$. To address this, we use the least-squares method to compute it as follows:

$$\tilde{q}^{\mathbf{j}}_{\mathbf{k}} = \frac{2^{j_2} \cdot u^{\mathbf{j}}_{\mathbf{k}} - 2^{j_1} \cdot v^{\mathbf{j}}_{\mathbf{k}}}{4^{j_1} + 4^{j_2}}. \tag{23}$$

Remark 2: Eq. 23 can be interpreted as the orthogonal projection of the 2D vector wavelet $[\tilde{u}^{\mathbf{j}}_{\mathbf{k}} \psi^1(2^{j_1}x - k_1)\psi^0(2^{j_2}y - k_2), \tilde{v}^{\mathbf{j}}_{\mathbf{k}} \psi^0(2^{j_1}x - k_1)\psi^1(2^{j_2}y - k_2)]^T$ in Eqs. 19 and 20 onto the *2D divergence-free wavelet* defined as follows:

$$\begin{aligned} & \nabla^{\perp}\psi^1(2^{j_1}x - k_1)\psi^1(2^{j_2}y - k_2) \\ & = \begin{bmatrix} +4 \cdot 2^{j_2}\psi^1(2^{j_1}x - k_1)\psi^0(2^{j_2}y - k_2) \\ -4 \cdot 2^{j_1}\psi^0(2^{j_1}x - k_1)\psi^1(2^{j_2}y - k_2) \end{bmatrix}. \end{aligned}$$

Finally, $q^{\mathbf{n}}$ can be recovered by applying a full-level inverse



$\begin{bmatrix} u_{\mathbf{k}}^{[2,2]} \\ v_{\mathbf{k}}^{[2,2]} \end{bmatrix}$
$q_{\mathbf{k}}^{[2,2]}$
$\begin{bmatrix} \tilde{u}_{k_2}^{1} \\ 0 \end{bmatrix}$
$\begin{bmatrix} \tilde{u}_{\mathbf{k}}^{[0,1]} \\ \tilde{v}_{\mathbf{k}}^{[0,1]} \end{bmatrix}$
$\begin{bmatrix} \tilde{u}_{\mathbf{k}}^{[1,1]} \\ \tilde{v}_{\mathbf{k}}^{[1,1]} \end{bmatrix}$
$\begin{bmatrix} \tilde{u}_{k_2}^{0} \\ 0 \end{bmatrix}$
$\begin{bmatrix} \tilde{u}_{\mathbf{k}}^{[0,0]} \\ \tilde{v}_{\mathbf{k}}^{[0,0]} \end{bmatrix}$
$\begin{bmatrix} \tilde{u}_{\mathbf{k}}^{[1,0]} \\ \tilde{v}_{\mathbf{k}}^{[1,0]} \end{bmatrix}$
$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$
$\begin{bmatrix} 0 \\ \tilde{v}_{k_1}^{0} \end{bmatrix}$
$\begin{bmatrix} 0 \\ \tilde{v}_{k_1}^{1} \end{bmatrix}$
$\tilde{q}_{k_2}^{j_2} = +\frac{1}{4} \cdot 2^{-j_2} \cdot \tilde{u}_{k_2}^{j_2}$
$\tilde{q}_{\mathbf{k}}^{\mathbf{j}} = \frac{2^{j_2} \cdot \tilde{u}_{\mathbf{k}}^{\mathbf{j}} - 2^{j_1} \cdot \tilde{v}_{\mathbf{k}}^{\mathbf{j}}}{4^{j_1} + 4^{j_2}}$
$q_0^0 = 0$
$\tilde{q}_{k_1}^{j_1} = -\frac{1}{4} \cdot 2^{-j_1} \cdot \tilde{v}_{k_1}^{j_1}$

Figure 4: *2D wavelet potential recovery on a grid of size $2^2 \times 2^2$: First, a full-level wavelet transform is applied to the velocity $\mathbf{u}^{[2,2]}$ to obtain its wavelet coefficients. Second, the wavelet coefficients $\tilde{q}^{\mathbf{j}}$ of $q^{[2,2]}$ are extracted using Eqs. 21 and 23. Finally, $q^{[2,2]}$ is recovered by applying a full-level inverse wavelet transform to $\tilde{q}^{\mathbf{j}}$.*

wavelet transform to $\tilde{q}^{\mathbf{j}}$. Fig. 4 illustrates the entire process of our 2D wavelet potential recovery on a grid of size $2^2 \times 2^2$.

4.3. 3D Wavelet Potential Recovery

Problem Setting. The 3D vector potential recovery problem on the unit cube $[0,1]^3$ with periodic boundary conditions is defined similarly to the 2D case. It can be expressed as follows:

$$\begin{aligned} \nabla \times \mathbf{q}(\mathbf{x}) &= \mathbf{u}(\mathbf{x}), \quad \mathbf{x} = [x,y,z]^T \in [0,1]^3, \qquad (24) \\ \mathbf{q}(0,y,z) &= \mathbf{q}(1,y,z), \\ \mathbf{q}(x,0,z) &= \mathbf{q}(x,1,z), \\ \mathbf{q}(x,y,0) &= \mathbf{q}(x,y,1). \end{aligned}$$

Here, the vector potential function $\mathbf{q} = [q_x(\mathbf{x}), q_y(\mathbf{x}), q_z(\mathbf{x})]^T$ needs to be recovered and $\mathbf{u}(\mathbf{x}) = [u(\mathbf{x}), v(\mathbf{x}), w(\mathbf{x})]^T$ represents a divergence free vector field.

Discretization. Similar to the two-dimension case, we divide the unit cube $[0,1]^3$ into identical cells with a cell size of $h = 1/2^n$, as illustrated Fig. 2c). We then sample $\mathbf{q}(\mathbf{x})$ on cell edges and $\mathbf{u}(\mathbf{x})$ on cell faces as follows:

$$\begin{aligned} q_{x,\mathbf{k}}^{\mathbf{n}} &= q_x(k_1 h + h/2, k_2 h, k_3 h), \\ q_{y,\mathbf{k}}^{\mathbf{n}} &= q_y(k_1 h, k_2 h + h/2, k_3 h), \\ q_{z,\mathbf{k}}^{\mathbf{n}} &= q_z(k_1 h, k_2 h, k_3 h + h/2), \\ u_{\mathbf{k}}^{\mathbf{n}} &= u(k_1 h, k_2 h + h/2, k_3 h + h/2), \\ v_{\mathbf{k}}^{\mathbf{n}} &= v(k_1 h + h/2, k_2 h, k_3 h + h/2), \\ w_{\mathbf{k}}^{\mathbf{n}} &= w(k_1 h + h/2, k_2 h + h/2, k_3 h), \end{aligned}$$

for each $\mathbf{k} = [k_1, k_2, k_3]^T \in \{0, 1, \cdots, 2^n - 1\}^3$. The superscript $\mathbf{n}$ in $q_{x,\mathbf{k}}^{\mathbf{n}}$, $q_{y,\mathbf{k}}^{\mathbf{n}}$, $q_{z,\mathbf{k}}^{\mathbf{n}}$, $u_{\mathbf{k}}^{\mathbf{n}}$, $v_{\mathbf{k}}^{\mathbf{n}}$ and $w_{\mathbf{k}}^{\mathbf{n}}$ denotes the values are sampled at level $\mathbf{n} = [n,n,n]$.

Wavelet Method. With the sampled values, we approximate $\mathbf{q}^{\mathbf{n}}(\mathbf{x})$, $\mathbf{u}^{\mathbf{n}}(\mathbf{x})$ as follows:

$$\begin{aligned} q_x^{\mathbf{n}}(\mathbf{x}) &= \sum_{\mathbf{k}} q_{x,\mathbf{k}}^{\mathbf{n}} \phi^0(2^n x - k_1) \phi^1(2^n y - k_2) \phi^1(2^n z - k_3), \\ q_y^{\mathbf{n}}(\mathbf{x}) &= \sum_{\mathbf{k}} q_{y,\mathbf{k}}^{\mathbf{n}} \phi^1(2^n x - k_1) \phi^0(2^n y - k_2) \phi^1(2^n z - k_3), \\ q_z^{\mathbf{n}}(\mathbf{x}) &= \sum_{\mathbf{k}} q_{z,\mathbf{k}}^{\mathbf{n}} \phi^1(2^n x - k_1) \phi^1(2^n y - k_2) \phi^0(2^n z - k_3), \\ u^{\mathbf{n}}(\mathbf{x}) &= \sum_{\mathbf{k}} u_{\mathbf{k}}^{\mathbf{n}} \phi^1(2^n x - k_1) \phi^0(2^n y - k_2) \phi^0(2^n z - k_3), \\ v^{\mathbf{n}}(\mathbf{x}) &= \sum_{\mathbf{k}} v_{\mathbf{k}}^{\mathbf{n}} \phi^0(2^n x - k_1) \phi^1(2^n y - k_2) \phi^0(2^n z - k_3), \\ w^{\mathbf{n}}(\mathbf{x}) &= \sum_{\mathbf{k}} w_{\mathbf{k}}^{\mathbf{n}} \phi^0(2^n x - k_1) \phi^0(2^n y - k_2) \phi^1(2^n z - k_3), \end{aligned}$$

We can utilize the 2D wavelet potential recovery method described in Sec. 4.2 as a fundamental component to derive the wavelet coefficients of $\mathbf{q}^{\mathbf{n}}(\mathbf{x})$ from those of $\mathbf{u}^{\mathbf{n}}(\mathbf{x})$. Since the curl operator is linear, Eq. 24 can be reformulated as follows:

$$\nabla \times \mathbf{q}(\mathbf{x}) = \nabla \times \begin{bmatrix} q_x(\mathbf{x}) \\ 0 \\ 0 \end{bmatrix} + \nabla \times \begin{bmatrix} 0 \\ q_y(\mathbf{x}) \\ 0 \end{bmatrix} + \nabla \times \begin{bmatrix} 0 \\ 0 \\ q_z(\mathbf{x}). \end{bmatrix}$$

We now have three coupled 2D wavelet potential recovery problems. Considering the wavelet transform is also linear and utilizing Eqs. 21 and 22, we can compute the wavelet coefficients of $\mathbf{q}^{\mathbf{n}}(\mathbf{x})$ from those of $\mathbf{u}^{\mathbf{n}}(\mathbf{x})$ as follows:

$$\left\{ \tilde{q}_{x,k_3}^{j_3} = +\frac{1}{4} \cdot 2^{-j_3} \cdot \tilde{v}_{k_3}^{j_3}, \quad \tilde{q}_{x,k_2}^{j_2} = -\frac{1}{4} \cdot 2^{-j_2} \cdot \tilde{w}_{k_2}^{j_2} \right\}_{k_1}^{j_1}, \qquad (25)$$

$$\left\{ \tilde{q}_{y,k_1}^{j_1} = +\frac{1}{4} \cdot 2^{-j_1} \cdot \tilde{w}_{k_1}^{j_1}, \quad \tilde{q}_{y,k_3}^{j_3} = -\frac{1}{4} \cdot 2^{-j_3} \cdot \tilde{u}_{k_3}^{j_3} \right\}_{k_2}^{j_2}, \qquad (26)$$

$$\left\{ \tilde{q}_{z,k_2}^{j_2} = +\frac{1}{4} \cdot 2^{-j_2} \cdot \tilde{u}_{k_2}^{j_2}, \quad \tilde{q}_{z,k_1}^{j_1} = -\frac{1}{4} \cdot 2^{-j_1} \cdot \tilde{v}_{k_1}^{j_1} \right\}_{k_3}^{j_3}, \qquad (27)$$

$$\tilde{q}_{x,\mathbf{k}}^{\mathbf{j}} \begin{bmatrix} 0 \\ +4 \cdot 2^{j_3} \\ -4 \cdot 2^{j_2} \end{bmatrix} + \tilde{q}_{y,\mathbf{k}}^{\mathbf{j}} \begin{bmatrix} -4 \cdot 2^{j_3} \\ 0 \\ +4 \cdot 2^{j_1} \end{bmatrix} + \tilde{q}_{z,\mathbf{k}}^{\mathbf{j}} \begin{bmatrix} +4 \cdot 2^{j_2} \\ -4 \cdot 2^{j_1} \\ 0 \end{bmatrix} = \begin{bmatrix} \tilde{u}_{\mathbf{k}}^{\mathbf{j}} \\ \tilde{v}_{\mathbf{k}}^{\mathbf{j}} \\ \tilde{w}_{\mathbf{k}}^{\mathbf{j}} \end{bmatrix}, \qquad (28)$$

where $\mathbf{k} = [k_1, k_2, k_3]^T \in \{0, \cdots, 2^{j_1} - 1\} \times \{0, \cdots, 2^{j_2} - 1\} \times \{0, \cdots, 2^{j_3} - 1\}$ for each $\mathbf{j} = [j_1, j_2, j_3] \in \{0, \cdots, n-1\}^3$. The curly braces $\left\{ \textbf{Equations} \right\}_{k_1}^{j_1}$ in Eq. 25 indicate that **Equations** holds for each $k_1 \in \{0, \cdots, 2^{j_1} - 1\}$ and for each $j_1 \in \{0, \cdots, n-1\}$. The same applies to Eq. 26 and Eq. 27.

Remark 1: The linear equation Eq. 28 is under-determined and has a rank of two, due to the presence of one degree of freedom in Eq. 24. This can be seen by noting that the gradient of any scalar field p can be added to $\mathbf{q}$ to obtain a new solution $\mathbf{q}_{new} = \mathbf{q} + \nabla p$ that satisfies Eq. 24. The parallel sweeping method [CPAB22] selects a smooth solution by enforcing $\nabla \cdot \mathbf{q} = 0$, which results in solving an expensive scalar Poisson equation. Instead, we select a solution that satisfies the constraint:

$$2^{j_1} \cdot \tilde{q}_{x,\mathbf{k}}^{\mathbf{j}} + 2^{j_2} \cdot \tilde{q}_{y,\mathbf{k}}^{\mathbf{j}} + 2^{j_3} \cdot \tilde{q}_{z,\mathbf{k}}^{\mathbf{j}} = 0. \qquad (29)$$

Experiments demonstrate that our solution is sufficiently smooth for vector potential interpolation. Please refer to Sec. 5.4 for visual results.

Remark 2: Solving Eq. 28 can be interpreted as performing a projection of the 3D vector wavelet:

$$\mathbf{u}^{\mathbf{j}}_{\mathbf{k}} = \tilde{u}^{\mathbf{j}}_{\mathbf{k}} \begin{bmatrix} 0 \\ \psi^0(2^{j_1}x-k_1)\psi^1(2^{j_2}y-k_2)\psi^0(2^{j_3}z-k_3) \\ \psi^0(2^{j_1}x-k_1)\psi^0(2^{j_2}y-k_2)\psi^1(2^{j_3}z-k_3) \end{bmatrix} + \tilde{v}^{\mathbf{j}}_{\mathbf{k}} \begin{bmatrix} \psi^0(2^{j_1}x-k_1)\psi^1(2^{j_2}y-k_2)\psi^1(2^{j_3}z-k_3) \\ 0 \\ \psi^1(2^{j_1}x-k_1)\psi^1(2^{j_2}y-k_2)\psi^0(2^{j_3}z-k_3) \end{bmatrix} + \tilde{w}^{\mathbf{j}}_{\mathbf{k}} \begin{bmatrix} \psi^1(2^{j_1}x-k_1)\psi^0(2^{j_2}y-k_2)\psi^0(2^{j_3}z-k_3) \\ \psi^0(2^{j_1}x-k_1)\psi^1(2^{j_2}y-k_2)\psi^0(2^{j_3}z-k_3) \\ 0 \end{bmatrix}$$

onto the *3D divergence-free wavelet* defined as follows:

$$\nabla\times \begin{bmatrix} \psi^0(2^{j_1}x-k_1)\psi^1(2^{j_2}y-k_2)\psi^1(2^{j_3}z-k_3) \\ \psi^1(2^{j_1}x-k_1)\psi^0(2^{j_2}y-k_2)\psi^1(2^{j_3}z-k_3) \\ \psi^1(2^{j_1}x-k_1)\psi^1(2^{j_2}y-k_2)\psi^0(2^{j_3}z-k_3) \end{bmatrix} = \begin{bmatrix} 0 \\ +4\cdot 2^{j_3}\cdot\psi^0(2^{j_1}x-k_1)\psi^1(2^{j_2}y-k_2)\psi^0(2^{j_3}z-k_3) \\ -4\cdot 2^{j_2}\cdot\psi^0(2^{j_1}x-k_1)\psi^0(2^{j_2}y-k_2)\psi^1(2^{j_3}z-k_3) \end{bmatrix} + \begin{bmatrix} -4\cdot 2^{j_3}\cdot\psi^1(2^{j_1}x-k_1)\psi^0(2^{j_2}y-k_2)\psi^0(2^{j_3}z-k_3) \\ 0 \\ +4\cdot 2^{j_1}\cdot\psi^0(2^{j_1}x-k_1)\psi^0(2^{j_2}y-k_2)\psi^1(2^{j_3}z-k_3) \end{bmatrix} + \begin{bmatrix} +4\cdot 2^{j_2}\cdot\psi^1(2^{j_1}x-k_1)\psi^0(2^{j_2}y-k_2)\psi^0(2^{j_3}z-k_3) \\ -4\cdot 2^{j_1}\cdot\psi^0(2^{j_1}x-k_1)\psi^1(2^{j_2}y-k_2)\psi^0(2^{j_3}z-k_3) \\ 0 \end{bmatrix}$$

Finally, with the wavelet coefficients $\tilde{\mathbf{q}}^{\mathbf{j}} = [\tilde{q}^{\mathbf{j}}_x, \tilde{q}^{\mathbf{j}}_y, \tilde{q}^{\mathbf{j}}_z]^T$ at hand, $\mathbf{q}^{\mathbf{n}}$ can be recovered by performing a full-level inverse wavelet transform to $\tilde{\mathbf{q}}^{\mathbf{j}}$.

4.4. Boundary Conditions

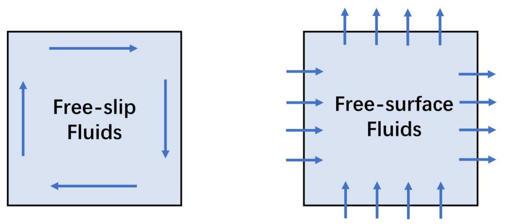
Up to this point, we have only considered potential recovery problems based on periodic boundary conditions. However, in real-world applications, Neumann and Dirichlet boundary conditions are frequently employed. When simulating free-slip fluids, the potential recovery problem uses Dirichlet boundary conditions, while the pressure Poisson equation uses Neumann boundary conditions. Conversely, when simulating free-surface fluids, the potential recovery problem uses Neumann boundary conditions, and the pressure Poisson equation uses Dirichlet boundary conditions. Fig. 5 depicts the boundary condition configurations for these two types of fluids. In this section, we will expand our wavelet potential recovery to include the Neumann and Dirichlet boundary conditions.

The Neumann and Dirichlet boundary conditions for the 1D potential recovery problem 5 are defined as follows, respectively:

$$q'(0) = q'(1) = 0, \tag{30}$$

$$q(0) = q(1) = 0. \tag{31}$$

To extend the 1D wavelet potential recovery described in Sec.4.1



	Free-slip Fluids	Free-surface Fluids
Potential Recovery Problem	Dirichlet	Neumann
Pressure Poisson's Equation	Neumann	Dirichlet

Figure 5: *Boundary condition configurations for simulating free-slip fluids and free-surface fluids are presented. In the top row, the normal component of velocity for free-slip fluids is zero, resulting in fluid flow along the tangent of the boundaries. Conversely, for free-surface fluids, the tangent component of velocity is zero, causing the fluid to flow along the normal of boundaries. It is crucial to apply the appropriate boundary conditions accurately when simulating these two types of fluids. The table below lists the boundary condition configurations for each type of fluid.*

to support these two boundary conditions, we need to implement 1D wavelet transforms satisfying the boundary conditions. Here, we adopt a practical approach. During each one-level wavelet transform, when we require the unknown approximate coefficients outside the boundaries, we extrapolate the coefficients using mirror reflection for Neumann boundary conditions and skew reflection for Dirichlet boundary conditions. Fig. 6 illustrates our approach for enforcing the Neumann and Dirichlet boundary conditions.

With the availability of 1D wavelet transforms that satisfy Neumann or Dirichlet boundary conditions, the wavelet potential recovery for problem 5 with Neumann boundary conditions requires only two modifications. Firstly, the wavelet transform of u^n needs to be changed from periodic to Dirichlet. Secondly, the inverse wavelet transform of $\tilde{u}^n$ needs to be changed from periodic to Neumann. The changes required for the wavelet potential recovery for problem 5 with Dirichlet boundary conditions are similar.

Since our 2D wavelet transform are separable for each dimension, we can perform the 2D wavelet transform by first applying the 1D wavelet transform to each row of the input signal (row transform) and then to each column of the row-transformed signal (column transform). The same applies to the 3D wavelet transform. Therefore, the modifications for the wavelet potential recovery to support the two boundary conditions in 2D and 3D are similar to the 1D case.

4.5. Implementation

The one-dimensional wavelet transform serves as the key component of our wavelet potential recovery methods in all dimensions (1D, 2D, and 3D). To implement this transform, we utilize the lifting scheme [SS96], which has lower complexity than the conventional convolution-based methods and can be performed in-place without requiring additional memory. In this section, we present algorithms for the wavelet potential recovery methods described

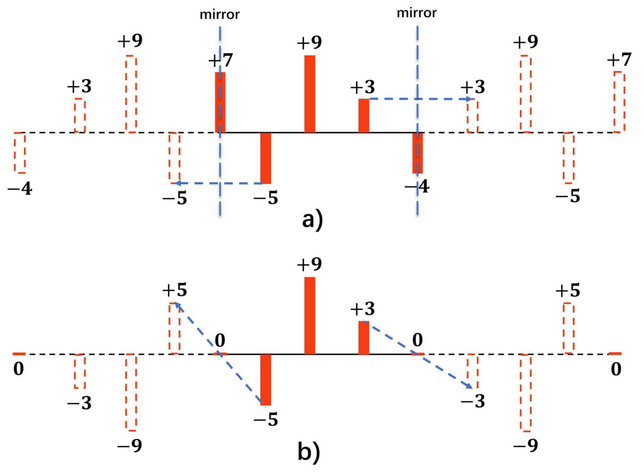


Figure 6: *Enforcing Neumann and Dirichlet boundary conditions using reflection and skew-reflection: a) To enforce Neumann boundary conditions, the unknown approximate coefficients outside the boundaries are extrapolated by mirror reflection along the boundaries during a one-level wavelet transform; b) To enforce Dirichlet boundary conditions, the unknown approximate coefficients outside the boundaries are extrapolated by skew reflection along the boundaries during a one-level wavelet transform.*

in Secs. 4.1∼ 4.3, using the one-dimensional lifting wavelet transform (referred to as LWT) and its inverse (referred to as iLWT) as building blocks.

We first implement the full-level one-dimensional lifting wavelet transform and its inverse as shown in Algs. 5 and 6, respectively. Using these transforms as building blocks, we then implement the full-level 2D lifting wavelet transform and its inverse as shown in Algs. 7 and 8, respectively. Finally, we implement the full-level 3D transform and its inverse based on the 1D and 2D transforms, as shown in Algs. 9 and 10, respectively.

With these transforms, we can implement algorithms for wavelet potential recovery methods for all dimensions (1D, 2D and 3D). These algorithms are listed in Algs. 2, 3 and 4 in Appendix A.

Memory Usage. We define N as the memory block size needed for values equal to the grid's size. Using a MAC grid for fluid properties, each velocity component's memory allocation slightly exceeds N. However, for simplicity, we approximate this as N. We also use the same treatments for the vector potentials. To compare the memory usage of the sweeping method and the proposed method, we use direct velocity interpolation as a baseline and count the extra memory blocks needed for pointwise incompressible interpolation.

In 2D, our method needs $2N$ memory to recover the vector potential. Initially, we transform the input velocities (with two components) into wavelet space, storing the coefficients in the $2N$ allocated memory. After obtaining the vector potential coefficients using Eq.22, we overwrite the velocity coefficients with them since the coefficients of velocities are no longer needed. Subsequently, an inverse wavelet transform is then performed on these coefficients, and due to the in-place wavelet transform, it can be done in the same memory block, making $2N$ sufficient for our method. In 3D, the process is similar, requiring $3N$ memory for vector potential recovery.

The sweeping method efficiently computes the vector potential in 2D using Eq. 8, requiring only N memory for vector potential storage. In 3D, the total memory usage for the sweeping method increases to $5N$, including $3N$ for storing the output vector potentials and two extra N-sized blocks for DST. One of the extra memory blocks is used to store the divergence of vector potentials obtained via Eq. 8. To enhance DST performance, we employ NVIDIA's Fast Fourier Transform, which requires another memory block.

5. Experiments

All of our experiments were conducted on a PC equipped with a 3.6-GHz Intel Core i7-6850K CPU and an NVIDIA GeForce RTX 3080 GPU. Our proposed method is mainly compared to the state-of-the-art parallel sweeping method [CPAB22]. The sweeping method requires an additional Poisson solver after obtaining the vector potential to guarantee its smoothness. Therefore, we compare both the sweeping methods with and without a smoothness solver in this article. For the proposed wavelet potential recovery method, we utilize the symmetry biorthogonal wavelets constructed by Cohen-Daubechies-Feauveau (cdf) in [CDF92]. In all of our experiments, we employ cdf4.6 and cdf3.7 for ψ^1 and ψ^0, respectively. All of the experiments use the cut-cell Multigrid [WMSF15] as the pressure Poisson solver.

5.1. Convergence Comparison of Various Methods

We begin by comparing the convergence of the proposed method to the parallel sweeping method in both 2D and 3D contexts. The results of these comparisons in 2D, under free-slip and free-surface boundary conditions for exterior boundaries, are clearly displayed in Fig. 7. The convergence of 3D scenes with varying boundary conditions is illustrated in Figs. 8 and 9.

After obtaining the resulting velocity from the Multigrid solver, we recover the vector potential using both the parallel sweeping method and our method. We then use the finite difference method to evaluate the velocity, denoted as $\mathbf{u}_r$, on the computational grids. Finally, we compare the velocity to the ideal divergence-free velocity $\mathbf{u}_g$. The relative error is calculated using the L_∞ norm as follows:

$$\epsilon = \frac{\|\mathbf{u}_g(\mathbf{x}_i) - \mathbf{u}_r(\mathbf{x}_i))\|_\infty}{\|\mathbf{u}_g(\mathbf{x}_i)\|_\infty}, \tag{32}$$

where $\mathbf{x}_i$ represents position i within the grid.

In the following convergence comparison figures, the curve labeled "Sweep" represents the results obtained from the parallel sweeping method. The "Wavelet" label indicates the outcomes derived from the proposed wavelet-based method. Meanwhile, the curve marked "Origin" displays the comparative results between the output velocities generated by the Multigrid solver and the ideal divergence-free velocities. A key difference between the 2D and 3D sweeping methods is the requirement for an additional Poisson solver to ensure the smoothness of vector potentials in 3D. In this article, we use the discrete sine transform (DST) as the additional

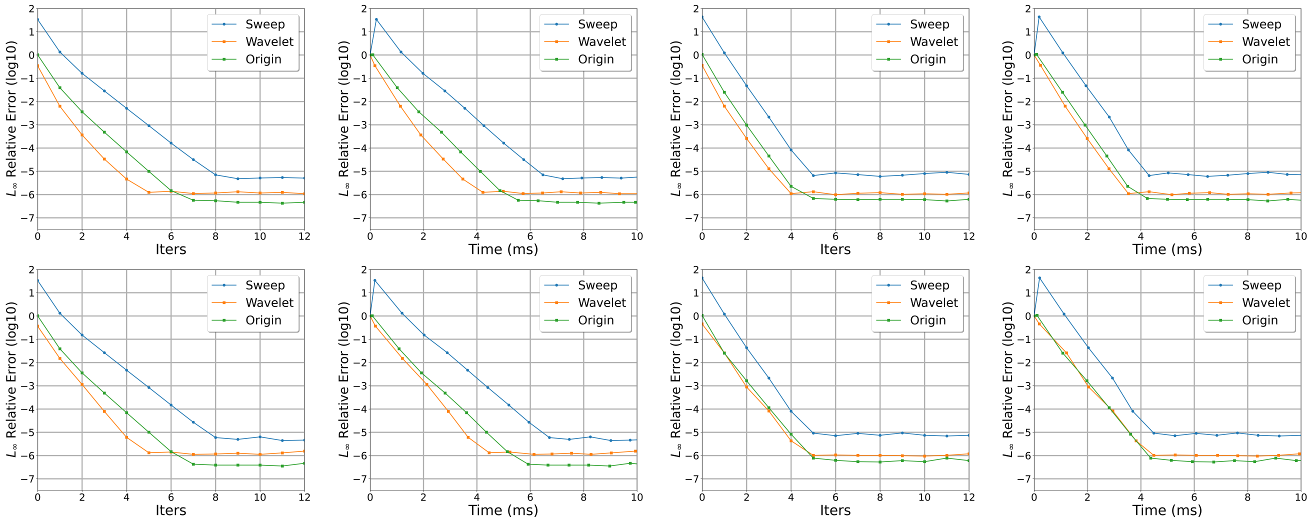


Figure 7: *Convergence comparison of different methods on a* 512^2 *grid without internal obstacles (top row) and with a central circular obstacle (bottom row), using randomly generated divergence-free velocity as the ground truth. Center-left: free-surface boundary conditions; Center-right: free-slip boundary conditions.*

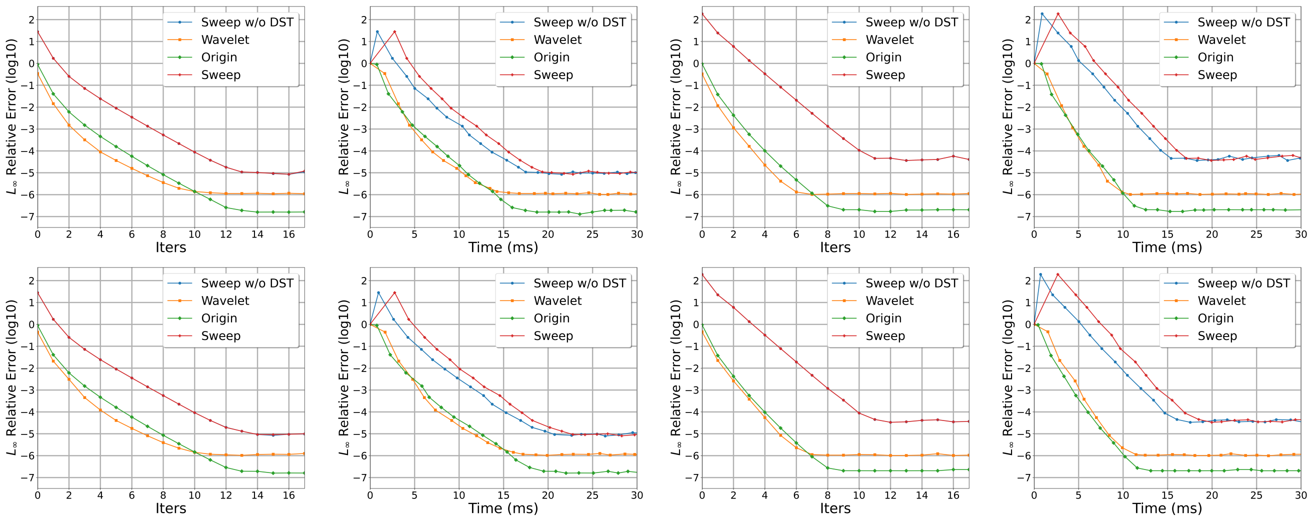


Figure 8: *Convergence comparison of different methods on a* 128^3 *grid without internal obstacles (top row) and with a central sphere (bottom row), using randomly generated divergence-free velocity as the ground truth. Center-left: free-surface boundary conditions; Center-right: free-slip boundary conditions.*

Poisson solver, a technique employed in [CPAB22]. In the three-dimensional figures, the sweeping method with the DST solver is denoted as "Sweep", while the version without the DST solver is referred to as "Sweep w/o DST".

2D Case. The top row of Fig. 7 presents results without interior obstacles, while the bottom row of Fig. 7 includes a central circular obstacle within the domain, both using randomly generated divergence-free velocities as ground truth. It can be observed that the proposed method requires about 1.5ms less time to achieve the same level of accuracy compared to the parallel sweeping method. Taking into account the time consumed by the pressure Poisson solver, our approach demonstrates an impressive speedup, approximately **1.5x**, compared to the parallel sweeping method, while maintaining a precision threshold of 10^{-5}. When the output velocities have residual divergence after a set number of Multigrid iterations, the advocated wavelet method proves effective in reducing this divergence and enhancing the overall accuracy. In contrast, the parallel sweeping method, instead of alleviating, introduces further divergence, resulting in a negative impact on accuracy. Consequently, the parallel sweeping method requires a higher number of Poisson solver iterations to match the accuracy levels achieved by our proposed method. From the provided figures, one can observe that before reaching full convergence, the curves representing both the proposed method and the Multigrid solver are almost overlapping, indicating that the proposed method ***does not need any additional time*** in deriving the vector potential when targeting a specific accuracy threshold.

3D Case. The 3D experiments yield results analogous to those observed in 2D. Fig. 8 presents comparisons conducted on 128^3 grids under both free-slip and free-surface boundary conditions, utilizing randomized input velocities. Specifically, the top row of Fig. 8 shows results without any internal obstacles, while the bottom row of Fig. 8 displays results with a central spherical obstacle. Compared to the results in two dimensions, the acceleration benefits of the proposed method in three dimensions become distinctly evident. Observations reveal that the proposed method requires roughly 8ms less to recover the vector potentials than the sweeping method, maintaining equivalent accuracy levels. Moreover, the proposed method has an approximate **2x** speedup over the sweeping method when targeting a precision of 10^{-5}.

Beyond the scope of random velocities, we further evaluate performance using velocities derived from simulation scenes, as illustrated in Figs. 14 and 15. We use the output velocities of the Multigrid solver with a tolerance factor of 10^{-13} as the ground truth. As can be discerned from Fig. 9, the outcomes in simulation scenes bear a close resemblance to those observed with random velocities. Moreover, Fig. 9 shows the advantage of the proposed method over the sweeping method is robust and not affected by initial guesses.

5.2. Accuracy Analysis

In this section, we evaluate the order of accuracy of various interpolation methods in both 2D and 3D. We use analytical divergence-free velocities as ground truth.

For both dimensions, we begin by generating a substantial number of random particles within the domain. Using the defined analytical velocities, we can then calculate the analytical velocity of each particle at its location. Subsequently, we sample the velocities on the staggered grid with various resolutions and recover the vector potentials using different methods. After that, we compute the interpolated velocities of particles using the bilinear direct interpolation method and vector potential interpolation method, respectively. Notably, we employ the quadratic B spline kernel [JST*16] for the vector potential interpolation. Finally, we compute the relative L2-norm error between the interpolated velocities and the analytical velocities.

Beyond the labels previously introduced, the curves labeled "Direct" in Figs. 10 and 11 represent the outcomes obtained through direct velocity interpolation.

In addition to the error measurement, Figures 10 and 11 display the streamlines corresponding to the defined analytical velocities. The streamlines are colored based on the magnitude of velocities, with the maximum magnitude colored yellow and the minimum magnitude colored red. Intermediate values are colored using a linear interpolation between these extremes.

2D Case. For 2D, we use the following analytical divergence-free velocity as the input for the compared methods:

$$\mathbf{u}_a(x,y) = \begin{bmatrix} \sin(3\pi x)\cos(3\pi y) \\ -\cos(3\pi x)\sin(3\pi y) \end{bmatrix}. \tag{33}$$

Table 1: *Computational time for vector potential recovery.*

Cases	Recovery Time (ms)	
	Sweeping Method	Proposed Method
Fig. 7 (Top-Left)	0.101	0.100
Fig. 7 (Top-Right)	0.078	0.100
Fig. 8 (Top-Left)	1.884	1.008
Fig. 8 (Top-Right)	1.867	1.019
Fig. 9 (Top)	1.854	1.044
Fig. 9 (Bottom)	1.852	1.042

The rate of convergence of the compared methods employing the velocities is depicted in Fig. 10.

In Fig. 10, the curve associated with the proposed method is parallel to the curve of $O(h^2)$, where h denotes the cell spacing. This indicates that the proposed method achieves second-order accuracy, similar to the direct velocity interpolation and the parallel sweeping methods.

3D Case. The analytical divergence-free velocity in 3D is given by:

$$\mathbf{u}_a(x,y,z) = \begin{bmatrix} \sin(3\pi x)\cos(3\pi y) - \sin(3\pi x)\cos(3\pi z) \\ \sin(3\pi y)\cos(3\pi z) - \cos(3\pi x)\sin(3\pi y) \\ \cos(3\pi x)\sin(3\pi z) - \cos(3\pi y)\sin(3\pi z) \end{bmatrix}. \tag{34}$$

The convergence rate of the compared methods using the velocity is presented in Fig. 11.

It can be observed from Fig. 11 that the parallel sweeping method without DST leads to significant numerical error. In contrast, the proposed method achieves second-order accuracy, similar to the parallel sweeping method and the direct interpolation method, without an additional Poisson solver after recovery.

In these experiments, by employing a double-precision floating-point format, all methods achieve convergence with an accuracy of less than 10^{-13} compared to the results of Multigrid via Eq. 32. Although the sweeping method and the proposed method do not converge to identical results as shown in Figs 7, 8, and 9, the curves for both methods in Figs. 10 and 11 nearly overlap since the accuracy of the quadratic spline kernel dominates the error.

5.3. Time Comparison of Potential Recovery

We compare the computational time for vector potential recovery of both the sweeping method and the proposed wavelet-based method in Table 1. It can be seen that the proposed method is faster than the sweeping method in most of the cases except for the 2D case with the free-flip boundary conditions. Given that the free-flip boundary has a prior condition where the vector potential on the boundary is zero, in the 2D case, unlike free-surface, the free-flip boundary does not require handling of boundary values. Therefore, the sweeping method with a free-flip boundary is faster compared to the free-surface boundary. In 3D scenarios, no matter what kinds of boundaries are employed, the proposed method is significantly faster than the sweeping method, as the latter incorporates an additional DST to ensure the smoothness of the vector potential.

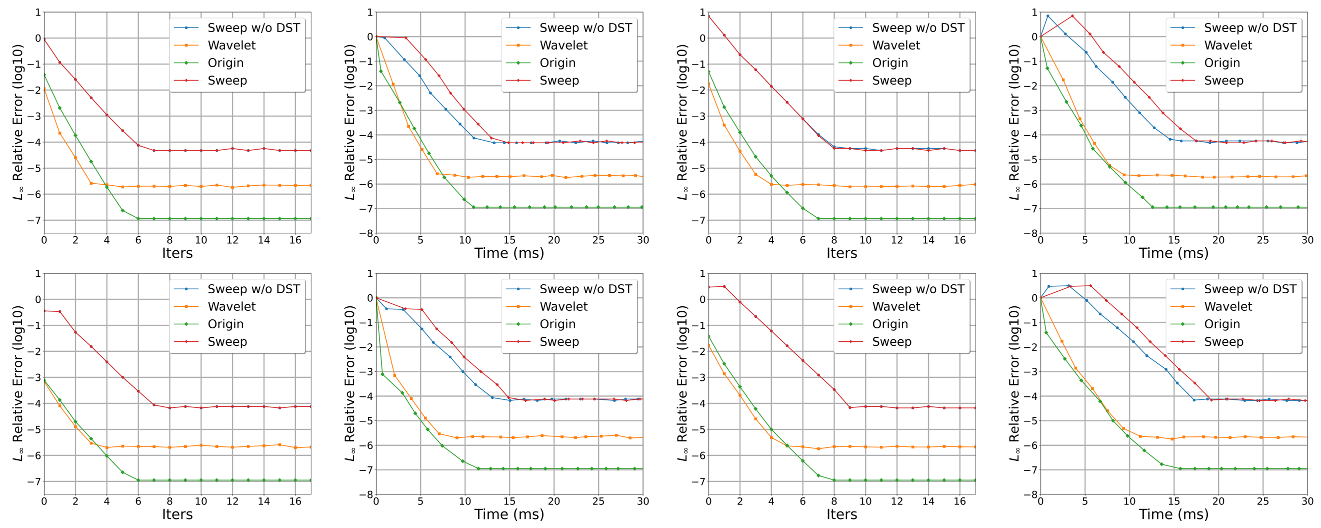


Figure 9: *Convergence comparison of different methods at a specific frame in a* 128^3 *smoke simulation without internal obstacles (top row, the simulation result is shown in Fig. 14) and with a central sphere (bottom row, the simulation result is shown in Fig. 15). Center-left: the results using the predicted pressure that is the output of the projection Poisson solver in the previous frame; Center-right: the results without predicted pressures.*

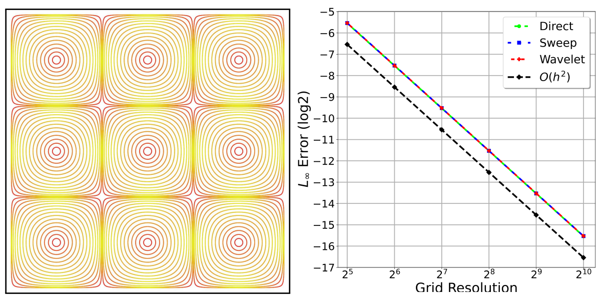


Figure 10: *Convergence rate of velocity interpolation in 2D. Left: the streamlines of the divergence-free velocity given by Eq.33. Right: a comparison of the L2-norm relative error between direct velocity interpolation, vector potential interpolation using the parallel sweeping method, and vector potential interpolation using the proposed method.*

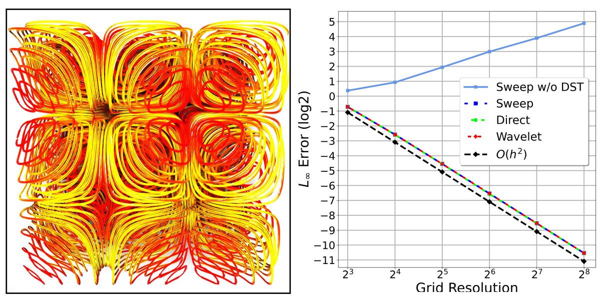


Figure 11: *Convergence rate of velocity interpolation in 3D. Left: the streamlines of the divergence-fee velocity given by Eq. 34. Right: a comparison of the L2-norm relative error between direct velocity interpolation, vector potential interpolation using the sweeping method without DST, vector potential interpolation using the sweeping method, and vector potential interpolation using the proposed method.*

5.4. Simulation Results

In this section, smoke simulations generated by the proposed method are compared to those simulated by the sweeping method.

Robust Test Cases. We begin our experiments by evaluating the robustness of the proposed method and the sweeping method. To conduct these evaluations, we simulate fluid scenes in both 2D and 3D, where smoke rises from a constant source using various Multigrid iterations. The smoke source is strategically placed at the center along the x and z axes. As the smoke rises from the source, it should display symmetry along the y-axis, resulting in a symmetrical pattern of smoke along this axis. It is worth noting that the velocity field in these experiments is advected by the velocity field interpolated from the recovered vector potential. Furthermore, to eliminate the influence of errors caused by single-float precision, we employ double-float precision.

Figs. 12 and 13 reveal that the proposed wavelet potential recovery method converges more rapidly than the sweeping method. These simulation results are consistent with the numerical comparison findings detailed in Section 5.1. Furthermore, the proposed wavelet potential recovery method produces smoke simulations without noticeable artifacts using only one Multigrid iteration, whereas the sweeping method exhibits evident asymmetry even after four iterations.

Convergence Comparison Cases. Fig. 14 shows the smoke rising from the constant source, whereas Fig. 15 depicts the smoke rising from the constant source and passing a spherical solid located in

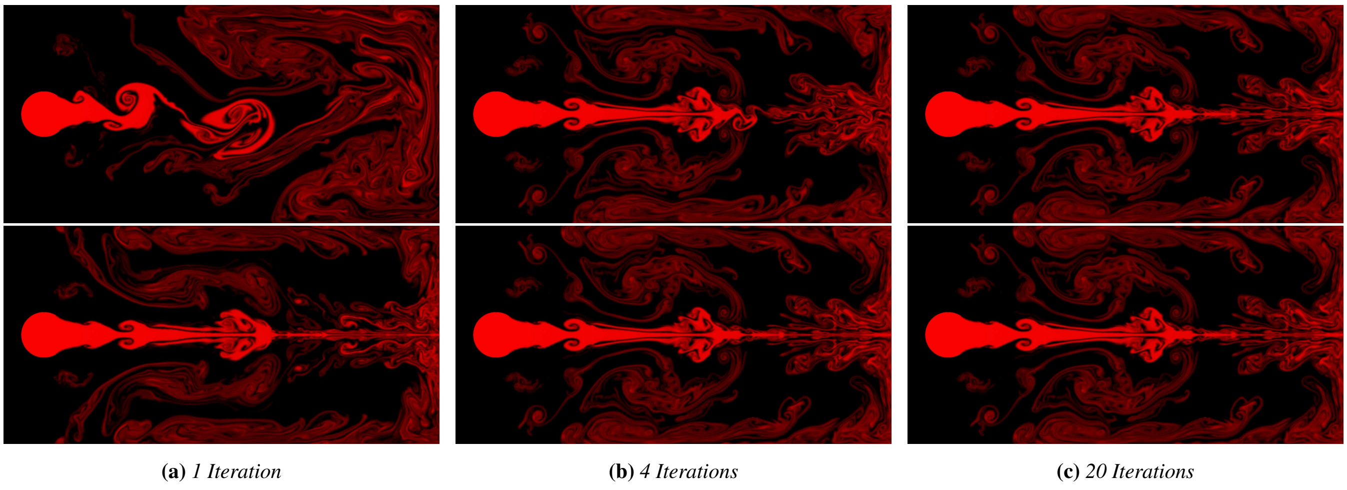


Figure 12: *Robust comparison of the proposed method and parallel sweeping method at frame 130 of a 256x512 simulation. Top: the sweeping method; Bottom: the proposed method.*

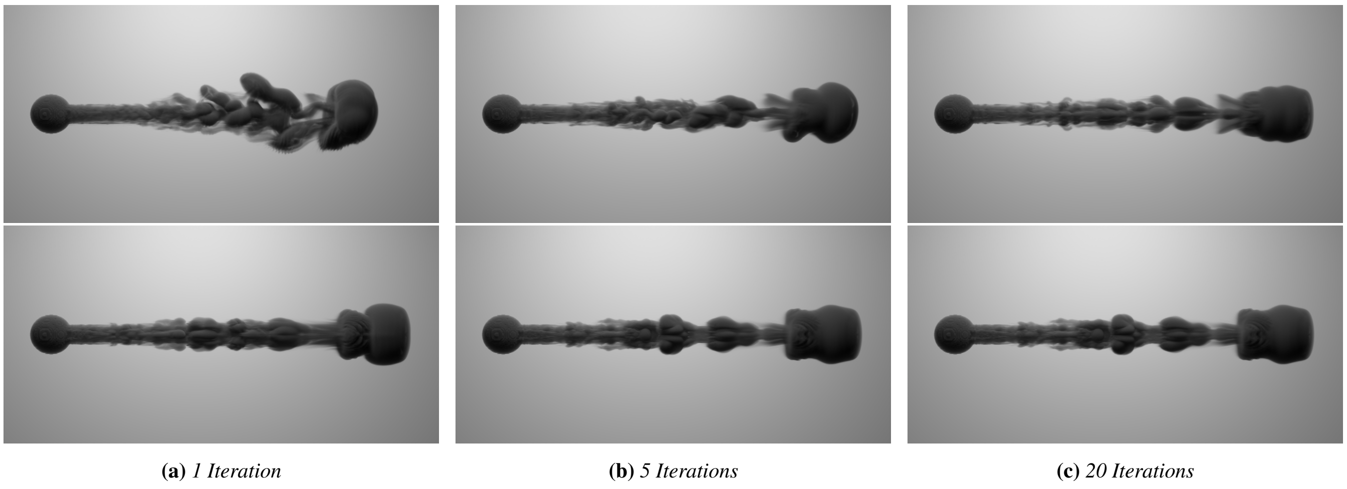


Figure 13: *Robust comparison of the proposed method and parallel sweeping method at frame 98 of a 128x256x128 simulation. Top: the sweeping method; Bottom: the proposed method.*

the center of the simulation domain. It is evident that the outcomes from the proposed method closely resemble the results obtained from the sweeping method. The proposed method does not require an additional Poisson solver to generate a smooth smoke simulation. In contrast, the sweeping method without DST may produce noticeable artifacts. It is worth noting that as the divergence-free wavelet projection improves accuracy, the proposed method may reach this threshold with fewer iterations and less time than the baseline simulation without vector potential recovery in some frames.

Accuracy Analysis Cases. Fig. 16 presents simulation cases where fluid particles are advected by Eq. 34 using various velocity interpolation methods. At the initiation of the simulation, smoke particles are sampled in the subdomain $(\frac{1}{3},\frac{2}{3})\times(\frac{1}{3},\frac{2}{3})\times(\frac{2}{3},1)$. According to Eq. 34, the velocities are equal to 0 at locations where the x, y, or z components are 0, $\frac{1}{3}$, $\frac{2}{3}$, or 1. Consequently, the smoke particles initialized within the subdomain should remain constrained within its boundaries, without crossing the subdomain.

In Fig. 16, the subdomain is represented by a white wireframe cube. The proposed wavelet potential recovery method can produce results similar to those simulated by the sweeping method. Some particles escape the subdomain when using the sweeping method without DST.

Pointwise Incompressible Interpolation Cases. We demonstrate the superiority of pointwise incompressible interpolation through five experiments. In order to distinguish between the pointwise incompressible interpolation using vector potentials recovered by the sweeping method and the proposed method, we refer to the former as "Sweeping Curl-Flow interpolation" and the latter as "Wavelet-based Curl-Flow interpolation".

The first one is the comparison of the particle trajectories flowing under a static 2D random velocity field on a 4x4 grid as shown

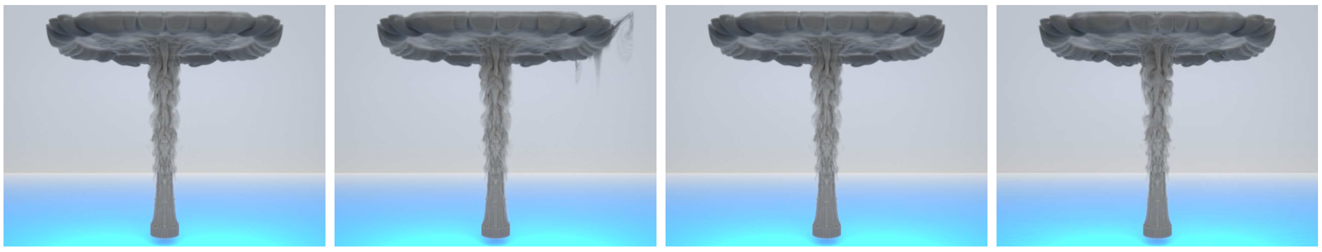

(a) *Direct Velocity Interpolation* **(b)** *Sweeping Method without DST* **(c)** *Sweeping Method* **(d)** *The Proposed Method*

Figure 14: *Comparisons of the proposed method and the parallel sweeping method at frame 160 of a* 128^3 *smoke simulation without internal obstacles. When targeting an accuracy threshold of* 5×10^{-5}*, the average time cost per frame for the various methods are as follows: a)* 10.25*ms; b)* 17.04*ms; c)* 18.77*ms; d)* 9.96*ms. Our method achieves comparable performance to the direct method and provides a* **2x** *speedup over the sweeping methods.*

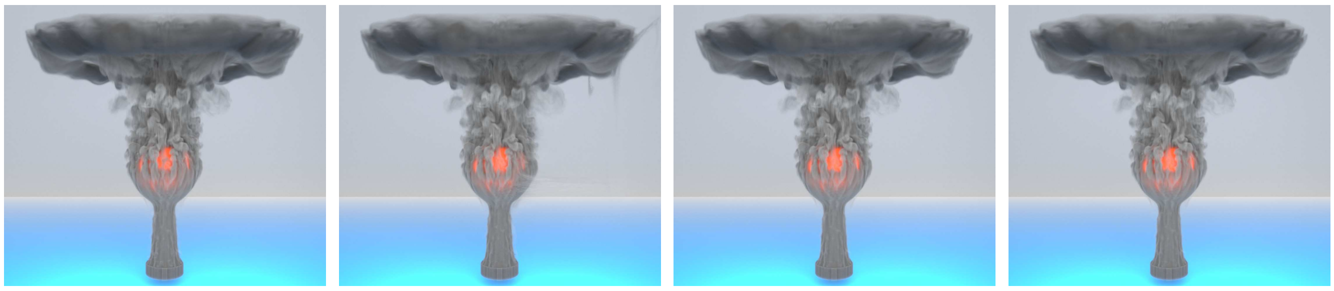

(a) *Direct Velocity Interpolation* **(b)** *Sweeping Method without DST* **(c)** *Sweeping Method* **(d)** *The Proposed Method*

Figure 15: *Comparisons of the proposed method and the parallel sweeping method at frame 192 of a* 128^3 *smoke simulation with a sphere. When targeting an accuracy threshold of* 5×10^{-5}*, the average time cost per frame for the various methods are as follows: a)* 9.13*ms; b)* 15.72*ms; c)* 17.92*ms; d)* 8.96*ms. Our method achieves comparable performance to the direct method and provides a* **2x** *speedup over the sweeping methods.*

in Fig. 17. Observations from Fig. 17 reveal that the direct bilinear interpolation method exhibits several spurious sinks which means particles may cluster at such places. Regardless of whether linear or quadratic interpolation kernels are employed, the Wavelet-Based Curl-Flow interpolation does not result in such spurious sinks. Moreover, the use of a quadratic interpolation kernel, due to its higher order, yields smoother particle trajectories.

Figs. 18 and 19 show the position of particles advected from uniformly sampled particles under static random velocity fields in 2D and 3D, respectively. After a period of time, particles using direct interpolation will cluster together while some areas become sparse or completely devoid of particles, reflecting an uneven distribution from their initially uniform state. Particles using Wavelet-based Curl-Flow interpolation will continue to be uniformly distributed in the domain.

The primary reason for these experimental results is that direct velocity interpolation cannot guarantee divergence at arbitrary points. In contrast, interpolation using the curl of vector potential can provide pointwise incompressible interpolation, thereby ensuring that fluid particles, which were originally uniformly distributed, remain uniformly distributed over time. Moreover, it can be observed that the sweeping method and the proposed method are capable of producing very similar results.

As mentioned in [CPAB22], Curl-Flow uses ramping methods to correct the behavior near the obstacles. This serves as a post-processing technique and does not affect the vector potential recovery process. Therefore we use a ramping method to correct the motion near the obstacles for both the proposed method and Curl-Flow in the following experiments.

Fig. 20 shows a slice of fluid particles passing a central spherical obstacle. In this experiment, fluid particles are frozen if they penetrate the interior of the spherical obstacle. The second experiment is smoke rising from the bottom of the domain and passing a cow-shaped obstacle, as illustrated in Fig. 21. In this experiment, smoke is represented by numerous smoke particles during the simulation. In the rendering process, the smoke particles are converted into smoke volume, and subsequently rendered in Houdini.

Figs. 20 and 21 show that some fluid particles are halted in the interior of the obstacles when using the direct velocity interpolation method. Conversely, fluid particles using the vector potential interpolation method traverse the obstacles smoothly regardless of whether vector potential is recovered by the sweeping method or the proposed wavelet potential recovery method.

Multiresolution Smoke. Thanks to our wavelet-based approach, we can obtain a natural multiresolution representation of the restored potential. Leveraging signal processing tools within the

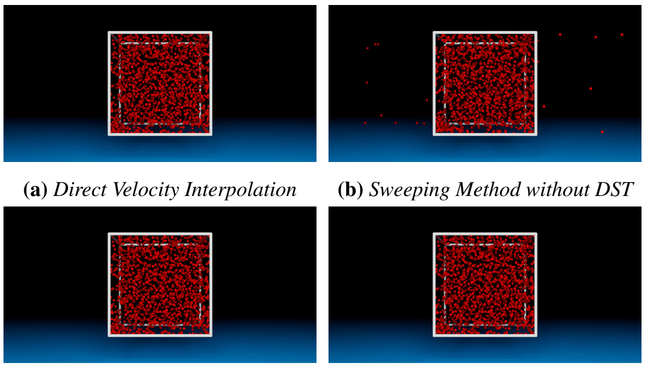

(a) *Direct Velocity Interpolation* (b) *Sweeping Method without DST* (c) *Sweeping Method* (d) *The Proposed Method*

Figure 16: *Comparisons of the proposed method and parallel sweeping method at frame 250 within a 3D computational domain* $[0,1]^3$*, where the fluid particles are advected by Eq. 34. The red points symbolize fluid particles, which were initially situated within the subdomain* $(\frac{1}{3},\frac{2}{3})\times(\frac{1}{3},\frac{2}{3})\times(\frac{2}{3},1)$*. The boundary of this subdomain is demarcated by the white wireframe cube.*

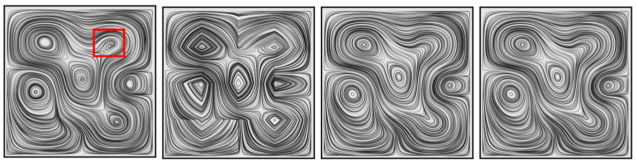


Figure 17: *The particle trajectories flowing under a static 2D random velocity field on a 4x4 grid. The velocity interpolation methods from left to right: Direct bilinear velocity interpolation; Linear Wavelet-based Curl-Flow interpolation; Quadratic Sweeping Curl-Flow interpolation; Quadratic Wavelet-based Curl-Flow interpolation. The red box in the left figure contains a spurious sink.*

wavelet community, we can simulate smoke with varying levels of detail.

To achieve this, we first eliminate the high frequencies of the potential by zeroing out the wavelet coefficients at the top levels. Next, we advect the smoke density using the vector potential that has been filtered for high frequencies. Finally, we generate multiresolution smokes. Figure 1 depicts a multiresolution smoke simulation.

6. Conclusion and Future Work

In this article, we have proposed a robust and efficient wavelet potential recovery method for simulating pointwise incompressible fluid. The key technique of our proposed method is to derive the wavelet coefficients of the vector potential from the wavelet coefficients of the divergence-free velocity using a fast approximate projection. Although previous research, such as [CPAB22], has explored vector potential recovery, we believe our work is the first in computer graphics to employ divergence-free wavelets to recover vector potential with little overhead when targeting a specific accuracy threshold. Experimental results support this claim. Furthermore, our wavelet-based approach can propose smooth results, eliminating the need for an extra Poisson solver to enhance

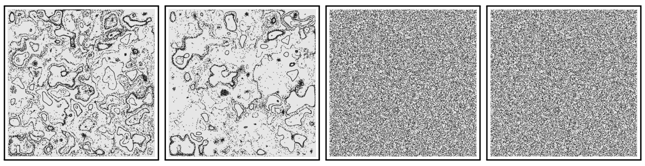


Figure 18: *Uniform sampled particles advected through a static 2D random velocity field within a 32x32 grid at frame 100. The velocity interpolation methods from left to right: Direct bilinear velocity interpolation; Direct monotonic cubic velocity interpolation; Quadratic Sweeping Curl-Flow interpolation; Quadratic Wavelet-based Curl-Flow interpolation.*

the smoothness of the vector potentials. Additionally, we propose specialized handling for wavelet transforms, allowing our approach to cater not only to periodic boundary conditions but also to Neumann and Dirichlet boundary conditions, thereby facilitating more intricate fluid simulations.

Although our method in this paper is tailored for pointwise incompressible fluid simulation, we believe that wavelet potential recovery can be applied to other fields due to the wide range of applications of vector potential in computer graphics.

In terms of future directions, there are two ways to extend our work. Firstly, wavelets naturally allow us to capture velocity fields at different frequencies. By incorporating higher-frequency wavelet coefficients during the potential recovery process, the fluid detail can be further enriched. Second, given that our wavelet potential recovery hinges on ideal divergence-free velocities, it can not recover the exact vector potential when the input velocities have residual divergence. Therefore, devising a more precise projection methodology becomes imperative to achieve higher accuracy during the potential recovery process.

Acknowledge

This work was funded in part by the Science and Technology Development Fund, Macau SAR (Grant no. 0075/2023/AMJ, 0003/2023/RIB1, 001/2024/SKL), in part by NSFC (62332015, 62072449, 62237001), in part by the Guangdong Science and Technology Department (Grant no. 2020B1515130001), in part by the Zhuhai Science and Technology Innovation Bureau (Grant no. ZH2220004002524), in part by the International Science and Technology project of Guangzhou Development District (Grant no. 2022GH09), in part by Zhuhai UM Research Institute (Grant no. HF-011-2021), in part by the University of Macau (Grant No.: MYRG2022-00059-FST, MYRG-GRG2023-00237-FST-UMDF), and in part by the Fund of China University of Ptroleum (Grant No. 21CX06042A).

References

[ATW15] Ando, Ryoichi, Thuerey, Nils, and Wojtan, Chris. "A stream function solver for liquid simulations". *ACM Transactions on Graphics (TOG)* 34.4 (2015), 1–9 2–4.

[BDG*17] Bao, Yuanxun, Donev, Aleksandar, Griffith, Boyce E, et al. "An immersed boundary method with divergence-free velocity interpolation and force spreading". *Journal of computational physics* 347 (2017), 183–206 2, 4.

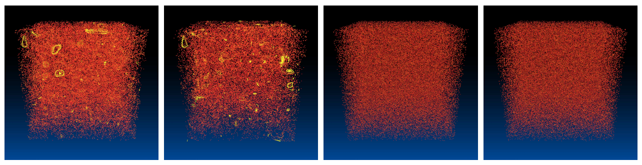
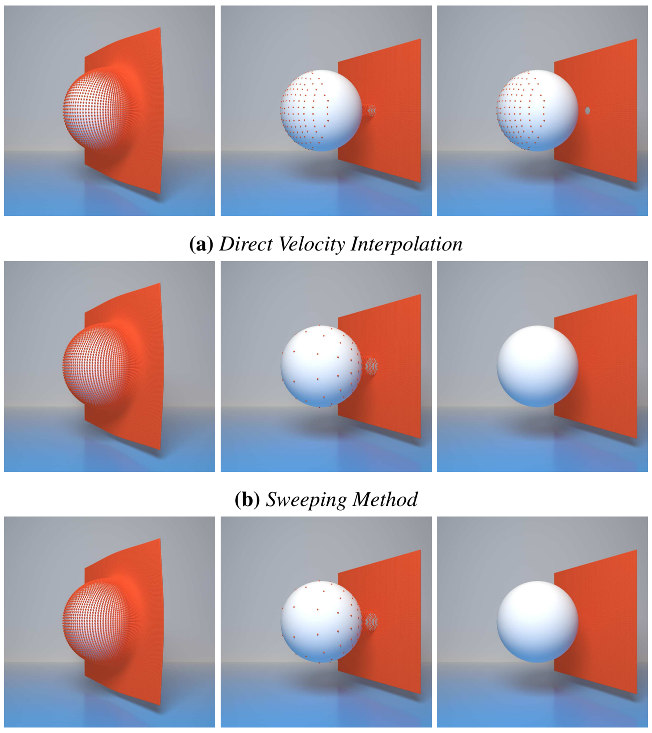


Figure 19: *Uniform sampled particles advected through a static 3D random velocity field within a 16x16x16 grid at frame 300. The velocity interpolation methods from left to right: Direct bilinear velocity interpolation; Direct monotonic cubic velocity interpolation; Quadratic Sweeping Curl-Flow interpolation; Quadratic Wavelet-based Curl-Flow interpolation. The color of each particle is determined by the number of particles within a spherical domain centered on that particle. The more particles there are, the closer the color of the central particle is to yellow; the fewer the particles, the closer it is to red.*

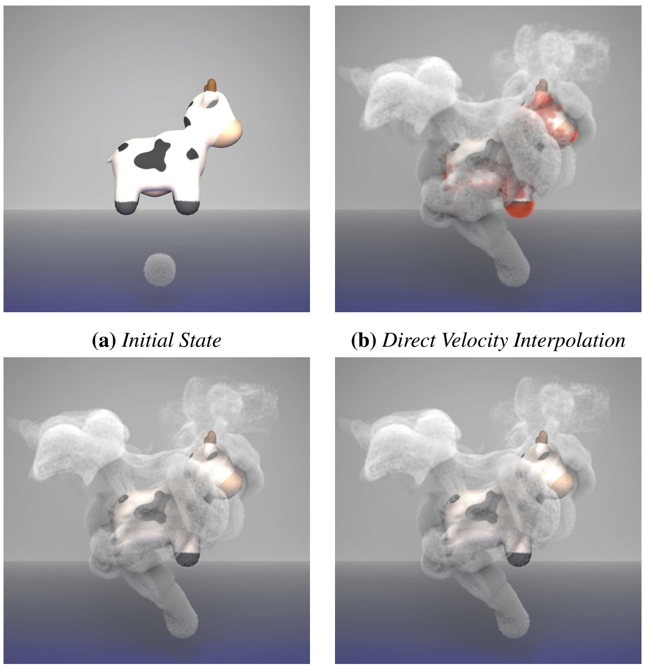


(a) *Direct Velocity Interpolation*

(b) *Sweeping Method*

(c) *The Proposed Method*

Figure 20: *Simulation results of fluid particles passing a spherical obstacle. In this case, fluid particles that run into the interior of the solid are halted.*



(a) *Initial State* **(b)** *Direct Velocity Interpolation*

(c) *Sweeping Method* **(d)** *The Proposed Method*

Figure 21: *Simulation results of smoke passing a spotted-cow-shaped obstacle. In this case, smoke that runs into the interior of the obstacle is halted and colored red.*

[BHN07] Bridson, Robert, Houriham, Jim, and Nordenstam, Marcus. "Curl-noise for procedural fluid flow". *ACM Transactions on Graphics (ToG)* 26.3 (2007), 46–es 2, 3.

[BSW*16] Biswas, Ayan, Strelitz, Richard, Woodring, Jonathan, et al. "A Scalable Streamline Generation Algorithm Via Flux-Based Isocontour Extraction." *EGPGV@ EuroVis*. 2016, 69–78 3.

[CDF92] Cohen, Albert, Daubechies, Ingrid, and Feauveau, J. C. "Biorthogonal bases of compactly supported wavelets". *Communications on Pure and Applied Mathematics* 45 (1992), 485–560 9.

[CPAB22] Chang, Jumyung, Partono, Ruben, Azevedo, Vinicius C, and Batty, Christopher. "Curl-Flow: Boundary-Respecting Pointwise Incompressible Velocity Interpolation for Grid-Based Fluids". *ACM Transactions on Graphics (TOG)* 41.6 (2022), 1–21 2–7, 9, 10, 14, 15.

[DB23] Ding, Xinwen and Batty, Christopher. "Differentiable Curl-Noise: Boundary-Respecting Procedural Incompressible Flows Without Discontinuities". *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6.1 (2023), 1–16 3.

[DeW05] DeWolf, Ivan. "Divergence-free noise". *Martian Labs* (2005) 2.

[DP06] Deriaz, Erwan and Perrier, Valérie. "Divergence-free and curl-free wavelets in two dimensions and three dimensions: application to turbulent flows". *Journal of Turbulence* 7 (2006), N3 2, 3.

[DP09] Deriaz, Erwan and Perrier, Valérie. "Orthogonal Helmholtz decomposition in arbitrary dimension using divergence-free and curl-free wavelets". *Applied and Computational Harmonic Analysis* 26.2 (2009), 249–269 3, 4.

[ETK*07] Elcott, Sharif, Tong, Yiying, Kanso, Eva, et al. "Stable, circulation-preserving, simplicial fluids". *ACM Transactions on Graphics (TOG)* 26.1 (2007), 4–es 3.

[HP12] Harouna, Souleymane Kadri and Perrier, Valérie. "Helmholtz-Hodge Decomposition on [0, 1] d by Divergence-free and Curl-free Wavelets". *Curves and Surfaces: 7th International Conference, Avignon, France, June 24-30, 2010, Revised Selected Papers 7*. Springer. 2012, 311–329 3.

[HP13] Harouna, S Kadri and Perrier, Valérie. "Effective construction of divergence-free wavelets on the square". *Journal of Computational and Applied Mathematics* 240 (2013), 74–86 3.

[HP15] Harouna, S Kadri and Perrier, Valérie. "Divergence-free wavelet projection method for incompressible viscous flow on the square". *Multiscale Modeling & Simulation* 13.1 (2015), 399–422 2, 3.

[HP21] HAROUNA, SOULEYMANE KADRI and PERRIER, VALÉRIE. "No-Slip and Free-Slip Divergence-Free Wavelets for the Simulation of Incompressible Viscous Flows". *Cartesian CFD Methods for Complex Applications*. Springer. 2021, 37–65 3.

[HP22] HAROUNA, S KADRI and PERRIER, VALÉRIE. "Homogeneous Dirichlet wavelets on the interval diagonalizing the derivative operator, and application to free-slip divergence-free wavelets". *Journal of Mathematical Analysis and Applications* 505.2 (2022), 125479 3.

[JSS*15] JIANG, CHENFANFU, SCHROEDER, CRAIG, SELLE, ANDREW, et al. "The affine particle-in-cell method". *ACM Transactions on Graphics (TOG)* 34.4 (2015), 1–10 2, 3.

[JST*16] JIANG, CHENFANFU, SCHROEDER, CRAIG, TERAN, JOSEPH, et al. "The material point method for simulating continuum materials". *Acm siggraph 2016 courses*. 2016, 1–52 11.

[KTJG08] KIM, THEODORE, THÜREY, NILS, JAMES, DOUG, and GROSS, MARKUS. "Wavelet turbulence for fluid simulation". *ACM Transactions on Graphics (TOG)* 27.3 (2008), 1–6 3.

[Lem92] LEMARIE-RIEUSSET, PIERRE GILLES. "Analyses multi-résolutions non orthogonales, commutation entre projecteurs et derivation et ondelettes vecteurs à divergence nulle". *Revista Matemática Iberoamericana* 8.2 (1992), 221–237 2–4.

[Les19] LESSIG, CHRISTIAN. "Divergence free polar wavelets for the analysis and representation of fluid flows". *Journal of Mathematical Fluid Mechanics* 21.1 (2019), 18 2.

[Mal08] MALLAT, STPHANE. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. 3rd. USA: Academic Press, Inc., 2008. ISBN: 0123743702 4.

[RLH*17] REN, XIAOHUA, LYU, LUAN, HE, XIAOWEI, et al. "Efficient Gradient-Domain Compositing Using an Approximate Curl-free Wavelet Projection". *Computer Graphics Forum*. Vol. 36. 7. Wiley Online Library. 2017, 207–215 3, 4.

[RLH*18] REN, XIAOHUA, LYU, LUAN, HE, XIAOWEI, et al. "Biorthogonal wavelet surface reconstruction using partial integrations". *Computer Graphics Forum*. Vol. 37. 7. Wiley Online Library. 2018, 13–24 3, 4.

[SB08] SCHECHTER, HAGIT and BRIDSON, ROBERT. "Evolving sub-grid turbulence for smoke animation". *Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 2008, 1–7 3.

[SDK21] SATO, SYUHEI, DOBASHI, YOSHINORI, and KIM, THEODORE. "Stream-guided smoke simulations". *ACM Transactions on Graphics (TOG)* 40.4 (2021), 1–7 3.

[SDY*15] SATO, SYUHEI, DOBASHI, YOSHINORI, YUE, YONGHAO, et al. "Incompressibility-preserving deformation for fluid flows using vector potentials". *The Visual Computer* 31 (2015), 959–965 3.

[SS96] SWELDENS, WIM and SCHRÖDER, PETER. "Building your own wavelets at home". *Wavelets in Computer Graphics*. ACM SIGGRAPH Course notes. 1996, 15–87 8.

[Sta99] STAM, JOS. "Stable Fluids". *In Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. USA: ACM Press/Addison-Wesley Publishing Co., 1999 2, 3.

[Ste11] STEVENSON, ROB. "Divergence-free wavelet bases on the hypercube: Free-slip boundary conditions, and applications for solving the instationary Stokes equations". *Mathematics of computation* 80.275 (2011), 1499–1523 3.

[Ste16] STEVENSON, ROB. "Divergence-free wavelets on the hypercube: General boundary conditions". *Constructive Approximation* 44 (2016), 233–267 3.

[Urb01] URBAN, KARSTEN. "Wavelet Bases in H(div) and H(curl)". *Mathematics of Computation* 70.234 (2001), 739–766. ISSN: 00255718, 10886842. (Visited on 08/31/2023) 3.

[Urb96] URBAN, KARSTEN. *Using divergence free wavelets for the numerical solution of the Stokes problem*. Citeseer, 1996 3.

[WMSF15] WEBER, DANIEL, MUELLER-ROEMER, JOHANNES, STORK, ANDRÉ, and FELLNER, DIETER. "A Cut-Cell Geometric Multigrid Poisson Solver for Fluid Simulation". *Computer Graphics Forum*. Vol. 34. 2. Wiley Online Library. 2015, 481–491 9.

[ZB05] ZHU, YONGNING and BRIDSON, ROBERT. "Animating sand as a fluid". *ACM Transactions on Graphics (TOG)* 24.3 (2005), 965–972 2, 3.

[ZH05] ZHOU, XIAOLIN and HE, YINNIAN. "Using divergence free wavelets for the numerical solution of the 2-D stationary Navier–Stokes equations". *Applied mathematics and computation* 163.2 (2005), 593–607 3.

Appendix A: Algorithms for Wavelet Potential Recovery

Algs. 2, 3, and 4 describe wavelet potential recovery algorithms for all dimensions (1D, 2D, and 3D). These algorithms are based on full-level lifting wavelet transforms and their inverses for all dimensions (1D, 2D and 3D) as shown in Algs. 5∼10 described in Appendix B.

Algorithm 2: 1D Wavelet Potential Recovery

```
Input: $u^n$                                  // a 1D $n$-level velocity
Output: $q^n$                                 // output potential
  // Compute $\tilde{u}^j$
1 $\tilde{u}^n$ = full-LWT($u^n$)
  // Compute $\tilde{q}^n$
2 for $j = 0 : n-1$ do
3     Compute $\tilde{q}^j$ by Eq. 15
  // Compute $q^n$
4 $q^n$ = full-iLWT($\tilde{q}^n$)
```

Algorithm 3: 2D Wavelet Potential Recovery

```
Input: $\mathbf{u^n} = [u^\mathbf{n}, v^\mathbf{n}]^T$          // a $\mathbf{n}$-level velocity
Output: $q^\mathbf{n}$                        // output potential
  // Compute $\tilde{u}^\mathbf{n}$ and $\tilde{v}^\mathbf{n}$
1 $\tilde{u}^\mathbf{n}$ = full-LWT2($u^\mathbf{n}$)
2 $\tilde{v}^\mathbf{n}$ = full-LWT2($v^\mathbf{n}$)
  // Compute $\tilde{q}^\mathbf{n}$
3 for $j_2 = 0 : n-1$ do
4     Compute $\tilde{q}^{[0,j_2]}$ by the first Eq. in Eq. 21
5 for $j_1 = 0 : n-1$ do
6     Compute $\tilde{q}^{[j_1,0]}$ by the second Eq. in Eq. 21
7 for $\mathbf{j} = [j_1, j_2] \in \{0, \cdots, n-1\}^2$ do
8     Compute $\tilde{q}^\mathbf{j}$ by Eq. 23
  // Recover $q^\mathbf{n}$
9 $q^\mathbf{n}$ = full-iLWT2($\tilde{q}^\mathbf{n}$)
```

Algorithm 4: 3D Wavelet Potential Recovery

```
Input: $\mathbf{u^n} = [u^n, v^n, w^n]^T$          // a n-level velocity
Output: $\mathbf{q^n} = [q_x^n, q_y^n, q_z^n]^T$    // output potential
// Compute $\tilde{u}^n$, $\tilde{v}^n$ and $\tilde{w}^n$
1  $\tilde{u}^n$ = full-LWT3($u^n$)
2  $\tilde{v}^n$ = full-LWT3($v^n$)
3  $\tilde{w}^n$ = full-LWT3($w^n$)
// Compute $\tilde{q}_x^n$, $\tilde{q}_y^n$ and $\tilde{q}_z^n$
4  for $j_1 = 0 : n-1$ do
5      for $j_3 = 0 : n-1$ do
6          Compute $\tilde{q}_x^{[j_1,0,j_3]}$ by the first Eq. in Eq. 25
7      for $j_2 = 0 : n-1$ do
8          Compute $\tilde{q}_x^{[j_1,j_2,0]}$ by the second Eq. in Eq. 25
9  for $j_2 = 0 : n-1$ do
10     for $j_1 = 0 : n-1$ do
11         Compute $\tilde{q}_y^{[j_1,j_2,0]}$ by the first Eq. in Eq. 26
12     for $j_3 = 0 : n-1$ do
13         Compute $\tilde{q}_y^{[0,j_2,j_3]}$ by the second Eq. in Eq. 26
14 for $j_3 = 0 : n-1$ do
15     for $j_2 = 0 : n-1$ do
16         Compute $\tilde{q}_z^{[0,j_2,j_3]}$ by the first Eq. in Eq. 27
17     for $j_1 = 0 : n-1$ do
18         Compute $\tilde{q}_z^{[j_1,0,j_3]}$ by the second Eq. in Eq. 27
19 for $\mathbf{j} = [j_1, j_2, j_3] \in \{0, \cdots, n-1\}^3$ do
20     Compute $\tilde{\mathbf{q}}^{\mathbf{j}}$ by solving Eq. 28 with constraint Eq. 29
// Compute $q_x^n$, $q_y^n$ and $q_z^n$
21 $q_x^n$ = full-iLWT3($\tilde{q}_x^n$)
22 $q_y^n$ = full-iLWT3($\tilde{q}_y^n$)
23 $q_z^n$ = full-iLWT3($\tilde{q}_z^n$)
```

Appendix B: Algorithms for Full-level Lifting Wavelet Transforms

Algorithm 5: full-LWT (Full-level LWT)

```
Input: $s^n$                  // a 1D n-level signal
Output: $\tilde{s}^n$         // wavelet coefficients
1  for $j = n : 1$ do
2      $\tilde{s}^{j-1}, s^{j-1}$ = LWT($s^j$)
```

Algorithm 6: full-iLWT (Full-level Inverse LWT)

```
Input: $\tilde{s}^n$          // wavelet coefficients
Output: $s^n$                 // output signal
1  for $j = 0 : n-1$ do
2      $s^{j+1}$ = iLWT($\tilde{s}^j, s^j$)
```

Algorithm 7: Full-LWT2 (Full-level 2D LWT)

```
Input: $s^n$                  // a 2D n-level signal
Input: $\tilde{s}^n$          // output wavelet coeffs.
// 1) Take full-LWT to each row of $s^n$
1  for $y = 0 : 2^n - 1$ do
2      $\tilde{s}^n(:, y)$ = full-LWT($s^n(:, y)$)
// 2) Take full-LWT to each column of $\tilde{s}^n$
3  for $x = 0 : 2^n - 1$ do
4      $\tilde{s}^n(x, :)$ = full-LWT($\tilde{s}^n(x, :)$)
```

Algorithm 8: Full-iLWT2 (Full-level Inverse 2D LWT)

```
Input: $\tilde{s}^n$          // wavelet coeffs.
Input: $s^n$                  // output 2d signal
// 1) Take full-iLWT to each column of $\tilde{s}^n$
1  for $x = 0 : 2^n - 1$ do
2      $\tilde{s}^n(x, :)$ = full-iLWT($\tilde{s}^n(x, :)$)
// 2) Take full-iLWT to each row of $\tilde{s}^n$
3  for $y = 0 : 2^n - 1$ do
4      $s^n(:, y)$ = full-iLWT($\tilde{s}^n(:, y)$)
```

Algorithm 9: Full-LWT3 (Full-level 3D LWT)

```
Input: $s^n$                  // a 3D n-level signal
Input: $\tilde{s}^n$          // output wavelet coeffs.
// 1) Take full-LWT2 to each z-slice of $s^n$
1  for $z = 0 : 2^n - 1$ do
2      $\tilde{s}^n(:, :, z)$ = full-LWT2($s^n(:, :, z)$)
// 2) Take full-LWT to each $\tilde{s}^n(x, y, :)$
3  for $x = 0 : 2^n - 1$ do
4      for $y = 0 : 2^n - 1$ do
5          $\tilde{s}^n(x, y, :)$ = full-LWT($\tilde{s}^n(x, y, :)$)
```

Algorithm 10: Full-iLWT3 (Full-level Inverse 3D LWT)

```
Input: $\tilde{s}^n$          // wavelet coeffs.
Input: $s^n$                  // output 3d signal
// 1) Take full-iLWT to each $\tilde{s}^n(x, y, :)$
1  for $x = 0 : 2^n - 1$ do
2      for $y = 0 : 2^n - 1$ do
3          $\tilde{s}^n(x, y, :)$ = full-iLWT($\tilde{s}^n(x, y, :)$)
// 2) Take full-iLWT2D to each z-slice of $\tilde{s}^n$
4  for $z = 0 : 2^n - 1$ do
5      $s^n(:, :, z)$ = full-iLWT($\tilde{s}^n(:, :, z)$)
```