Impulse Fluid Simulation

Fan Feng, Jinyuan Liu, Shiying Xiong, Shuqi Yang, Yaorui Zhang, Bo Zhu

Abstract—We propose a new incompressible Navier–Stokes solver based on the impulse gauge transformation. The mathematical model of our approach draws from the impulse–velocity formulation of Navier–Stokes equations, which evolves the fluid impulse as an auxiliary variable of the system that can be projected to obtain the incompressible flow velocities at the end of each time step. We solve the impulse-form equations numerically on a Cartesian grid. At the heart of our simulation algorithm is a novel model to treat the impulse stretching and a harmonic boundary treatment to incorporate the surface tension effects accurately. We also build an impulse PIC/FLIP solver to support free-surface fluid simulation. Our impulse solver can naturally produce rich vortical flow details without artificial enhancements. We showcase this feature by using our solver to facilitate a wide range of fluid simulation tasks including smoke, liquid, and surface-tension flow. In addition, we discuss a convenient mechanism in our framework to control the scale and strength of the turbulent effects of fluid.

Index Terms—fluid simulation, vortical structures, gauge methods, physics-based animation.

1 Introduction

Solving transformed Navier–Stokes equations has been an intriguing research topic for an extended period in fluid mechanics, numerical mathematics, and computer graphics (e.g., [1], [2]). The most salient example of these transformations is the vorticity form of Navier–Stokes equations: By taking the curl of both sides of the momentum conservation equation, we obtain the description of three-dimensional, incompressible, viscous fluids by evolving its vorticity field instead of its velocity in the fluid domain. At the end of each time step, the velocity field is obtained from the vorticity field by solving a Poisson equation for each axis. Such vorticity–velocity formulation, facilitated by its many Lagrangian numerical schemes (conventionally named as "vortex methods", see [2] for a thorough review), has demonstrated its inherent advantages in expressing and preserving vortical features in many computer graphics and computational physics applications [3], [4], [5], [6].

In addition to vortex methods, many other Navier–Stokes transformations have been explored in the history of computational fluid dynamics (CFD), such as impulse method [1], [7], velicity method [8], magnetization variable [9], impetus term [10], and continuous projection form [11], all of which decouple the system's temporally evolving variable from the fluid velocity. Such mathematical decoupling will create additional freedom in the numerical procedure to tackle the interaction between the fluid boundary and the global incompressibility constraints. In the CFD literature, this family of methods was referred as "gauge methods," and the extra set of variables being evolved is called "gauge variables." Pioneered by Roberts [12] and Oseledets [13] and followed by many others (e.g., see [1], [8], [10], [14]), gauge methods have been used widely in building high-order Navier–Stokes solvers to tackle complex incompressible fluid simulations in engineering. Mathematically, a gauge method solves the evolution of an auxiliary variable (usually denoted as $\boldsymbol{m}$) instead of the fluid velocity $\boldsymbol{u}$ directly. Specifically, $\boldsymbol{m}$'s divergence-free part gives $\boldsymbol{u}$, which can be calculated via a projection step $\boldsymbol{u} = \boldsymbol{m} - \nabla \boldsymbol{p}$ based on Helmholtz-Hodge decomposition, with $\boldsymbol{p}$ as an auxiliary scalar variable. The input of the projection step in a gauge method is the gauge variable, and the output is the divergence-free velocity. In contrast, the input and output of a projection step in a conventional way are both the velocity field. This projection step indicates potentially the most salient feature of a gauge method that the evolved gauge variable and the projected fluid velocity are only loosely coupled and can deviate from each other even drastically over time.

Gauge methods can potentially benefit fluid animation applications in two aspects. First, the weak coupling between the gauge and the physical quantities allows designing an appropriately evolving gauge variable that can produce and preserve the visually essential flow features. Second, the similarities between a gauge formulation and the original Navier–Stokes equations make it possible to build a gauge solver by reusing the plethora of existing high-performance simulation infrastructures. However, to date, notwithstanding its wide applications in solving complex fluid simulations in CFD, there was hardly any previous success in leveraging or modifying a gauge solver to directly furnish support for fluid animation applications. We speculate two reasons for this silent failure: First, most of the gauge formulations are not Lagrangian (e.g., [14]). We cannot obtain a $D\boldsymbol{m}/Dt$ from the gauge expression conveniently, which hinders the usage of many mature Lagrangian-nature numerical techniques such as the semi-Lagrangian advection scheme [15] and the PIC/FLIP method [16] that are vital to a graphics application's effects and performance. Second, despite its high-order accuracy, the numerical implementation of a gauge method is typically overly complicated, especially when it comes to its boundary treatments. Such implementation complexities prevent the technique from being used widely in accommo-

• *Everyone is with the Department of Computer Science, Dartmouth College, Hanover, NH, 03755.*
E-mail: fan.feng.gr@dartmouth.edu

Manuscript received xx xx, 2021; revised xx xx, 2022.

dating light-weighted fluid simulation applications where numerical accuracy is not the top consideration. Third, we lack a clear mechanistic understanding of the relationship between a mathematical gauge formulation and its potential for delivering better fluid simulation effects. In short, the CFD gauge methods are not fully ready for use to support a broad scope of fluid animation applications.

This paper devises a versatile gauge incompressible flow solver to facilitate fluid animation applications. We aim to bring to the computer graphics audience' attention a specific category of gauge solvers — the impulse gauge solvers — which was first derived by Cortez et al. [7], [17], [18] to model the momentum transfer process between inviscid incompressible flow and massless elastic membranes. Compared with other mainstream gauge methods, the impulse gauge method is featured by its particular affinities to the conventional fluid solvers in computer graphics, such as an advection-projection method or a vortex method. We elucidate three of these similarities as follows. First, the impulse gauge variable is Lagrangian. In Cortez's original work [17], the impulse variables were stored and evolved on a set of discrete particles to calculate the evolution of force-bearing filaments or membranes immersed in a fluid environment. Second, the boundary treatments in an impulse solver resemble the traditional way in an advection-projection scheme. Third, the physical meaning of impulse is inherently connected to vorticity, all such fields change in time along the paths like a material element moving with the fluid [19], which indicates the potential of an impulse method better to preserve vortical flow features than the standard methods. In retrospect, an impulse method was typically classified in the same category as vortex methods in many literature surveys (e.g., see [20]). One of the main differences between an impulse solver and a vortex solver is that an impulse solver requires to solve the Poisson equation for once (the same as in a standard advection-projection scheme) to obtain the divergence-free velocity field.

Considering these potential computational benefits, we present a new incompressible fluid solver based on the impulse gauge formulation of Navier–Stokes equations. Our model is discretized on an Eulerian grid. The vast majority of our solver's modules are built based on the existing grid-based fluid simulation techniques, including the marker-and-cell (MAC) discretization, semi-Lagrangian advection, finite difference, Poisson solver and projection, etc. With these modulated high-performance algorithms available in hand, we delivered two key innovations to adapt an impulse fluid model into a classical Eulerian fluid simulation framework. First, we devised an implicit algorithm to handle the impulse stretching term. Second, we employ a harmonic representation to handle the interaction between the free boundary and the interior. With the additional harmonic model integrated into our framework, we can capture the various interfacial flow phenomena such as strong surface tension and solid boundary effects. We show by a host of numerical simulations that our impulse solver can facilitate a rich set of fluid simulation tasks in computer graphics, ranging from smoke, liquid, surface-tension flow to solid-fluid coupling. We show that a naive implementation of our impulse gauge solver can capture and evolve vortical flow structures better than a conventional advection-projection scheme enhanced by vorticity confinement. We further provide an artistic tuning parameter in our framework that can control the scale and strength of the generated vorticity, allowing the animation of turbulent effects with a wide range of vorticity scales.

We summarize our technical contributions as follows:

- We develop the first Eulerian impulse-based gauge method to support vortical flow simulations.
- We devise a novel numerical scheme to facilitate the numerical treatment of stretching, which can be extended to other types of fluid solvers.
- We extend the impulse method to an Eulerian-Lagrangian framework with a harmonic treatment to handle free-surface flow with surface tension.

2 Related works

2.1 Impulse and other gauge methods

Cortez [7] first introduced an impulse-based method in order to treat boundary forces, by defining and evolving the impulse density equivalent to the velocity up to a gradient of a scalar field. His work is limited to free space by assuming uniform density and can not tackle discontinuity near the interface. Cortez [21] analyzes a Lagrangian numerical method based on impulse variables, by showing that the impulse flow approximates the flow induced by smoothed vortex sheets. Thus, the impulse variables can represent dipoles and the impulse equations update the dipole strengths appropriately, which are suitable to model immersed boundary problems with surface force. Cortez [18] uses a combined vortex/impulse method to study flows induced by the motion of thin flexible boundaries immersed in an incompressible fluid. The impulse elements are attached to the boundaries and are used to model the boundary-induced forces. The vortices are surrounding the boundaries to account for the viscous diffusion. Among these works, impulse density must be carried by Lagrangian particles to have physical meaning, and the interfacial discontinuity remains unsolved. In our method, we store impulse variables on an Eulerian grid and design a series of treatments for different types of boundaries commonly seen in fluid simulation. The general gauge method was introduced back in [13] by rewriting the Navier-Stokes equations into its Hamiltonian formula. The main purposes of choosing different types of gauge variables include allowing accurate treatment of boundary conditions [22], preserving invariants of the flow [8] and enhancing the solver's accuracy [23]. The most recent progress in this community includes Robert Saye's works [14], [24], [25] which developed a series of gauge methods for multiphase fluid flow problems with large interfacial discontinuity. Based on the discontinuous Galerkin framework, different choices of the gauge with appropriate boundary handling have been proposed for various types of fluids. In this work, we adapt some of Saye's ideas on boundary treatments for our impulse-based method, but we evolve a different gauge variable. In the most recent work [26], they evolve a transformed wave function as the gauge variable to preserve vortical structures. This work is different from ours for the physical meaning of the gauge variable and the numerical treatment of each term.

2.2 Vortex methods and stretching treatments

Vortex methods are special types of gauge methods by devising vorticity as the gauge variable and rewriting the fluid equations into their vorticity-velocity form [27], [28], [29]. By advecting vorticity directly, these solvers naturally preserve circulation within the fluid [2]. Vorticity methods usually rely on Lagrangian elements to evolve the flow features, for instance, particles [2], [3], [30], filaments [27], [31], [32], and sheets [5], [29]. Vorticity can also be used as a primary variable to improve the mesh-based Eulerian simulation [33], which inspires [34] to use vorticity directly in existing grid-based solvers. The drawbacks of vorticity modeling lie in the difficulties of geometric management and boundary treatments of certain types. Many recent approaches are of a hybrid fashion, either by using grids to improve pure Lagrangian methods [35], or by integrating Lagrangian elements into existing grid-based solvers [28], [36], [37]. A new term that arises from vorticity modeling is stretching, which does not take place in 2D flow [38], but plays an important role in 3D vortex methods by producing local intensification and reorientation of the vorticity [3], [34], [36]. In our work, we explore different implementations of stretching and eventually find that implicit stretching on cell centers produces robust and high-quality results. To avoid the vorticity field becoming divergent and resulting in instability of the simulation, we rely on the viscous diffusion to maintain stability.

2.3 Other vorticity confined methods

Two mainstream approaches to preserve vorticity structures in the graphics community include using high-order numerical schemes to reduce dissipation and adding additional artificial terms to replenish dissipated details. For the first category, advection is usually the main source of diffusion, and people have designed a series of high order advection schemes, such as BFECC [39], MacCormack [40], BiMocq [41], and energy-preserving integrators [42]. A special yet effective technique to alleviate diffusion of grid-based advection is by incorporating Lagrangian particles. Zhu and Bridson [43] adapt Fluid-Implicit-Particle (FLIP) method [44] by interpolating the change of the flow from the grid to reduce diffusion, which inspires the development of a series of hybrid Lagrangian-Eulerian schemes to conserve flow details [45], [46], [47], [48]. For the second category, Fedkiw et al. [15] first adapt the vorticity confinement force derived from the local flow field and add it to the control equation as an artificial force. Kim et al. [49] use the wavelet decomposition to find missing high-frequency components and synthesize them back to the velocity field. Bridson et al. [50] generate turbulent velocity fields based on Perlin noise. Our simulator is orthogonal to these methods and we do incorporate some techniques to reduce dissipation in our experiments. Recently, Chern et al. [51] open another direction for vortical flow simulation by solving Schrödinger's equation, which exhibits visually appealing vortical details.

3 Physical Model

3.1 Incompressible Navier–Stokes equations

We consider the incompressible Navier–Stokes equations in domain Ω with its boundary including free surface $\partial\Omega_f$ and solid wall $\partial\Omega_b$. We write the governing equations as

$$\begin{cases} \dfrac{D\boldsymbol{u}}{Dt} = -\boldsymbol{\nabla}\left(\dfrac{p}{\rho} - G\right) + \nu\nabla^2\boldsymbol{u}, \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0, \end{cases} \tag{1}$$

where $D/Dt = \partial/\partial t + \boldsymbol{u}\cdot\boldsymbol{\nabla}$ is the material derivative, p is the pressure, ρ is the fluid density (constant for incompressible flow), $G = \boldsymbol{g}\cdot\boldsymbol{x}$ is the gravitational potential, and ν is the kinematic viscosity.

TABLE 1: Notation Table

Notation	Description
$\boldsymbol{m}\in\mathbb{R}^d$	Impulse
$\boldsymbol{u}\in\mathbb{R}^d$	Velocity
$\varphi\in\mathbb{R}$	Gauge variable for incompressibility
$q\in\mathbb{R}$	Gauge variable for free surface
ϕ	Levelset for free surface
$\nu\in\mathbb{R}$	Kinematic viscosity
$G\in\mathbb{R}$	Gravitational potential
$\gamma\in\mathbb{R}$	Surface tension
$\kappa\in\mathbb{R}$	Mean curvature of the free surface
$\rho\in\mathbb{R}$	fluid density
$\rho'\in\mathbb{R}$	smoke soot density

On the free surface, the normal stress of the fluid balances the force of surface tension

$$p = \gamma\kappa, \qquad \boldsymbol{x}\in\partial\Omega_f, \tag{2}$$

where γ is the surface tension coefficient and κ is the mean curvature of the free surface. Here, we do not consider the viscous stress on the free surface. On the solid boundary, the flow satisfies non-penetrating boundary conditions, namely the normal flow velocity is equal to the normal solid velocity as

$$\boldsymbol{n}\cdot\boldsymbol{u} = \boldsymbol{n}\cdot\boldsymbol{u}_b, \qquad \boldsymbol{x}\in\partial\Omega_b, \tag{3}$$

where $\boldsymbol{u}_b$ is the velocity of the boundary wall, and $\boldsymbol{n}$ is the surface normal. We do not consider the non-slip condition here.

3.2 Gauge transformation

3.2.1 *Impulse gauge variables*

In this section, we show the transformation of Navier–Stokes equations to its impulse gauge form according to [7].

We first transform the divergence-free velocity field $\boldsymbol{u}$ to its impulse expression $\boldsymbol{m}$ as

$$\boldsymbol{m} = \boldsymbol{u} + \boldsymbol{\nabla}\varphi, \tag{4}$$

where φ satisfies

$$\begin{cases} \nabla^2\varphi = \boldsymbol{\nabla}\cdot\boldsymbol{m}, & \boldsymbol{x}\in\Omega \\ \partial_n\varphi = 0, & \boldsymbol{x}\in\partial\Omega_b, \\ \varphi = 0, & \boldsymbol{x}\in\partial\Omega_f, \end{cases} \tag{5}$$

with $\partial_n = \boldsymbol{n}\cdot\boldsymbol{\nabla}$ as the normal gradient. The equation set (5) can be analogized as the conventional projection step by solving a Poisson equation (the first row) constrained by the standard free-surface (the second row) and solid boundary conditions (the third row), with $\boldsymbol{m}$ and $\boldsymbol{u}$ as the input and output fields of the projection.

Substituting (4) and (5) into (1), (2), and (3) yields the impulse form of Navier–Stokes:

$$\begin{cases} \dfrac{D\boldsymbol{m}}{Dt} = -(\boldsymbol{\nabla u})\cdot\boldsymbol{m} - \boldsymbol{\nabla} q + \nu\nabla^2\boldsymbol{m}, & \boldsymbol{x}\in\Omega, \\ \boldsymbol{u} = \boldsymbol{m} - \boldsymbol{\nabla}\varphi, & \boldsymbol{x}\in\Omega \\ \boldsymbol{n}\cdot\boldsymbol{m} = \boldsymbol{n}\cdot\boldsymbol{u}_b, & \boldsymbol{x}\in\partial\Omega_b, \\ q = \dfrac{\gamma\kappa}{\rho} - \dfrac{1}{2}|\boldsymbol{u}|^2 - G + \nu\nabla^2\varphi, & \boldsymbol{x}\in\partial\Omega_f, \end{cases} \tag{6}$$

where

$$q = \frac{p}{\rho} - \frac{D\varphi}{Dt} - \frac{1}{2}|\boldsymbol{u}|^2 - G + \nu\nabla^2\varphi. \tag{7}$$

We refer the readers to the Appendix for a detailed deduction of (6). Similar to the standard Navier–Stokes equations, the equation set (6) contains a momentum equation (the first row), an incompressibility constraint (the second row), solid (the third row) and free-surface (the fourth row) boundary conditions. The free-surface boundary conditions are enforced via a gauge variable q, which we will explain more in the next sub-subsection. The momentum equation in (6) holds a Lagrangian form of $\boldsymbol{m}$, whose evolution is governed by the three terms on the right-hand side. These three terms include the stretching of $\boldsymbol{m}$, the gradient of q, and the viscosity of $\boldsymbol{m}$. Instead of directly solving fluid velocity $\boldsymbol{u}$, we evolve the impulse $\boldsymbol{m}$, and then recover $\boldsymbol{u}$ from $\boldsymbol{m}$ by (4). The benefit in this construction is that $\boldsymbol{m}$ is freed from having to satisfy a global incompressibility constraint in its dynamics evolution, holding several advantages for designing flexible temporal evolution schemes.

3.2.2 *The choice of q*

We devise a second gauge variable q to transfer boundary forces on the free surface such as surface tension into the bulk domain. Mathematically, because q is a gauge variable, its values can be defined arbitrarily (and the values of φ and $\boldsymbol{m}$ need to change accordingly, of course), without affecting the behavior of $\boldsymbol{u}$. We show a numerical example below to illustrate this gauge-velocity independence. On the other hand, because the values of q, φ, and $\boldsymbol{m}$ are interdependent with each other, we want to enforce additional constraints in the system to guarantee a unique solution of (6). There are many different ways to add such constraints. We choose to enforce a harmonic q following the design choice made in [14] to yield stable free-surface behaviors. We solve $\nabla^2 q = 0$ with a Neumann boundary condition on Ω_b. Other values for the right hand side is also acceptable but a Poisson solve is a must since we want to transfer the surface tension smoothly inside and still maintain a jump in pressure with the air outside. We choose 0 for the right hand side here out of convenience for calculation. Thus, we obtain the constraint equation of q by combining (2) and (7) as

$$\begin{cases} \nabla^2 q = 0, & \boldsymbol{x}\in\Omega, \\ \partial_n q = 0, & \boldsymbol{x}\in\partial\Omega_b \\ q - \dfrac{\gamma\kappa}{\rho} + \dfrac{1}{2}|\boldsymbol{u}|^2 + G = 0, & \boldsymbol{x}\in\partial\Omega_f. \end{cases} \tag{8}$$

We do not count the viscosity effect on the free surface.

It is worth noting that we only solve for a harmonic q when there is a free surface (liquids) in the simulation. If there is no free surface (smoke), we simply set $q = 0$ for the entire domain, which reduces the computation cost of solving the Poisson equation for q with nontrivial conditions. This simplification was introduced and validated by Cortez in his dissertation [17] (Chapter 2) when solving the incompressible impulse flow without boundary. One can also add surface tension into $\boldsymbol{m}$ directly and only perform one Poisson solve to obtain the velocity, although numerically it is not as stable as performing two separate Poisson solves.

We conduct a simple numerical experiment using the pseudo-spectral method [52] to show the gauge-velocity independence by simulating a Taylor vortex flow with different choices of q. Figure 1 shows the simulation results and the temporal evolution of the volume-averaged kinetic energy of Taylor vortex flow for different q selections. We can see that for different q, their flow fields exhibit the same evolution with consistent statistical energy.

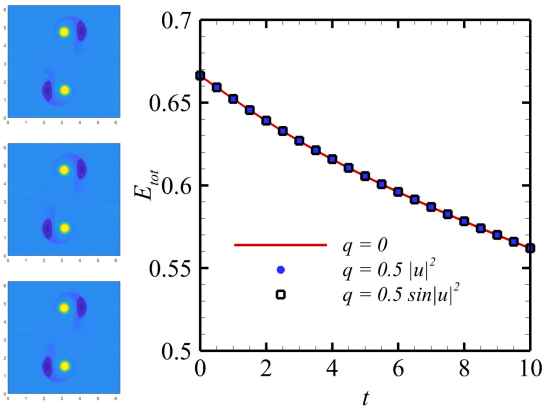


Fig. 1: The temporal evolution of the volume-averaged kinetic energy of Taylor vortex flow with different choices of q. The simulation was implemented using the pseudo-spectral method in MatLab (see the supplementary source code for details).

3.3 Incompressible impulse flows

Finally, we obtain the ready-to-solve impulse flow equations by combining equations (5), (6), and (8) as

$$\begin{cases} \dfrac{D\boldsymbol{m}}{Dt} = -(\boldsymbol{\nabla u})\cdot\boldsymbol{m} - \boldsymbol{\nabla} q + \nu\nabla^2\boldsymbol{m}, \\ \nabla^2 q = 0, \\ \nabla^2\varphi = \boldsymbol{\nabla}\cdot\boldsymbol{m}, \\ \boldsymbol{u} = \boldsymbol{m} - \boldsymbol{\nabla}\varphi, \end{cases} \quad \boldsymbol{x}\in\Omega. \tag{9}$$

The equation set (9) specifies the impulse transformation of Navier–Stokes with the impulse $\boldsymbol{m}$ and gauge variables φ and q. The first equation specifies the momentum conservation. The second equation constraints a harmonic gauge variable q. The third and fourth equations specify the projection from $\boldsymbol{m}$ to $\boldsymbol{u}$.

The solid boundary conditions are specified as

$$\begin{cases} \boldsymbol{n}\cdot\boldsymbol{m} = \boldsymbol{n}\cdot\boldsymbol{u}_b, \\ \partial_n\varphi = 0, \\ \partial_n q = 0, \end{cases} \quad \boldsymbol{x}\in\partial\Omega_b, \tag{10}$$

and the free surface boundary conditions are

$$\begin{cases} \varphi = 0, \\ q - \dfrac{\gamma\kappa}{\rho} + \dfrac{1}{2}|\boldsymbol{u}|^2 + G = 0 \end{cases} \boldsymbol{x} \in \partial\Omega_f.D \tag{11}$$

To summarize, we derive the impulse equations (9), (10), and (11) to facilitate the modeling of incompressible flow with a free surface.

3.4 Buoyancy

We adopt the Boussinesq approximation [53] to simulate fluid with the evolving smoke soot density. Variations in fluid properties other than density are ignored, and density only appears when it is multiplied by the gravitational constant. In particular, buoyancy can be applied to $\boldsymbol{m}$ directly as

$$\begin{cases} \dfrac{D\boldsymbol{m}}{Dt} = -(\boldsymbol{\nabla u})\cdot\boldsymbol{m} - \boldsymbol{\nabla}q + \nu\nabla^2\boldsymbol{m} + \dfrac{\rho'}{\rho}\boldsymbol{g}, \\ \dfrac{D\rho'}{Dt} = 0, \end{cases} \tag{12}$$

where ρ' is the effective smoke density and ρ a constant background density. The other conditions for solving (12) are the same as for (9), (10), and (11).

4 Numerical Algorithm

We devise a numerical framework to solve the incompressible impulse flow equations defined in (9), (10), (11), and (12). Our solver uses a standard MAC grid for spatial discretization, where both $\boldsymbol{u}$ and $\boldsymbol{m}$ are stored on grid face centers and the gauge variables φ, q, smoke soot density ρ' or levelset value ϕ are stored on cell centers. A levelset function is used to track the liquid interface.

We assume that the readers are already familiar with a standard advection-projection fluid simulator such as [15], [54]. Most of the numerical implementations of an impulse solver can reuse parts in a standard advection-projection solver, such as discretization, Semi-Lagrangian advection, projection, and applying body forces. The simulation loop in a single time step consists of reinitialization, advection, stretching, viscosity, and projection. We will discuss each step as follows. Because the impulse variable $\boldsymbol{m}$ is updated in many of these steps, we highlight the naming convention for $\boldsymbol{m}$ that the superscript of $\boldsymbol{m}$ we use here describes the time discretization of impulse while the subscript indicates the specific intermediate step of the time splitting of $\boldsymbol{m}$.

4.1 Reinitialization

At the beginning of each time step, we choose to reinitialize impulse $\boldsymbol{m}^n$ by blending its values with a small portion of the velocity $\boldsymbol{u}^n$ after a time interval. In particular, we set $\boldsymbol{m}_r^n = r\boldsymbol{m}^n + (1-r)\boldsymbol{u}^n$, with r as a global parameter close to 1. In most of our simulations, we use $r = .97$ for each time step. $r = 1$ means no reinitialization, while $r = 0$ means $\boldsymbol{m}^n$ is completely reinitialized as $\boldsymbol{u}^n$. The motivation for this reinitialization step is to avoid the impulse variable accumulate over time and become too large, which causes numerical instability during simulation. It is worth noting that this reinitialization scheme only blurs the non-divergence-free part of $\boldsymbol{m}$, which guarantees to obtain the same projected velocity after projection. Similar numerical treatments have been employed in other forms of gauge methods (e.g., see [14]), but we use reinitialization for each time step instead of once every several steps. We demonstrated the numerical effect of r in our smoke simulation in Section 6.2.

4.2 Advection

We need to advect impulse by velocity as $D\boldsymbol{m}/Dt = 0$, when only considering the advection effect. Specifically, we discretize the advection step as the following formula:

$$\frac{\boldsymbol{m}_r^{n+1} - \boldsymbol{m}_r^n}{\Delta t} + \boldsymbol{u}^n \cdot \boldsymbol{\nabla}\boldsymbol{m}_r^n = \boldsymbol{0}, \tag{13}$$

from which we get

$$\boldsymbol{m}_r^{n+1} = \boldsymbol{m}_r^n - \Delta t\boldsymbol{u}^n \cdot \boldsymbol{\nabla}\boldsymbol{m}_r^n. \tag{14}$$

Since our method is orthogonal to different advection schemes, here we adopt the classic Semi-Lagrangian method for smoke simulations and PIC/FLIP method for liquid simulations to advect the impulse field and other auxiliary quantities. For PIC/FLIP advection, we first interpolate different quantities ($\boldsymbol{m}_r^n$, $\boldsymbol{u}^n$) to particles and then evolve each particle as $\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \Delta t\boldsymbol{u}^n$, where $\boldsymbol{x}$ represents the position of a particle. Then we use extrapolation to return quantities back to the grid and the following operations happen on the grid.

4.3 Stretching

Accompanied by advection, $(\nabla\boldsymbol{u})\cdot\boldsymbol{m}$ in (9) is the stretching term for impulse, which accounts for the deformation of $\boldsymbol{m}$ due to advection. Although the stretching term also appears in the vortex method in three dimension, we need some different treatments specifically to $\boldsymbol{m}$. After extensive experiments with different numerical stretching methods, we find that implicit stretching computation on cells provides the best numerical stability.

With time splitting and discretization in an implicit manner, we have

$$\boldsymbol{m}_s^{n+1} = \boldsymbol{m}_r^{n+1} - \Delta t(\boldsymbol{\nabla}\boldsymbol{u}^n)\cdot\boldsymbol{m}_s^{n+1}. \tag{15}$$

After arranging terms, we obtain

$$\boldsymbol{m}_s^{n+1} = [\boldsymbol{I} + \Delta t(\boldsymbol{\nabla}\boldsymbol{u}^n)^T]^{-1}\boldsymbol{m}_r^{n+1}. \tag{16}$$

Because both velocity and impulse quantities are stored in a staggered version, we can not process this step fully on the center of grid faces. We need to first interpolate impulse and velocity to cell centers respectively to align storage positions of different components. We then compute $\nabla\boldsymbol{u}^n$ using the third-order central difference method discretized on a Cartesian grid. For calculating the inverse transpose of the local deformation gradient, $[\boldsymbol{I} + \Delta t(\boldsymbol{\nabla}\boldsymbol{u}^n)^T]^{-1}$, we use a direct solve. When Δt is small, the local deformation gradient matrix is close to an identity matrix, thus avoiding singularity. Lastly, instead of directly setting the impulse on the face center to the newly calculated impulse, we calculate the difference of $\boldsymbol{m}$ before and after stretching on the cell, interpolating it back to the face and applying the delta difference on the original impulse on the grid face.

4.4 Viscosity

As a numerical solver of incompressible viscous flow, we calculate the viscosity by using a stable numerical scheme. Specifically, we use the implicit backward Euler time integration scheme [55] to discretize $\nu\nabla^2\boldsymbol{m}$ in (9) as

$$\boldsymbol{m}_v^{n+1} = \boldsymbol{m}_s^{n+1} + \nu\Delta t\Delta_h\boldsymbol{m}_v^{n+1}. \tag{17}$$

After gathering $\boldsymbol{m}_v^{n+1}$ term to the left-hand side of (17), we obtain

$$[\boldsymbol{I} - \nu\Delta t\Delta_h]\boldsymbol{m}_v^{n+1} = \boldsymbol{m}_s^{n+1}, \tag{18}$$

where Δ_h is the second-order central difference scheme of the Laplacian operator. By using the red-black Gauss-Seidel iterative method to solve the linear system (18) until convergence, we obtain the numerical solution of $\boldsymbol{m}_v^{n+1}$. Since the kinematic viscosity ν used in our experiments are set to be very small 1×10^{-3}–1×10^{-6}, the iterative solver usually converges in 2-4 iterations. Notice that some other gauge methods also add viscosity either explicitly or implicitly for a stable solve.

4.5 Boundary and external force

We numerically solve the harmonic q for free-surface flow simulation to transfer the boundary pressure jump (such as surface tension) into the interior fluid region. The solver relies on the same Poisson system as solving for φ (the projection step), with differences only occurring on the right-hand side. Therefore, we only assemble the matrix once and it will be used for both the harmonic step and the projection step. We solve (8) to obtain q. Jump conditions that need to be added to the right-hand side include the pressure jump caused by the local curvature, as well as G, which is the gravitational potential $G = \boldsymbol{g}\cdot\boldsymbol{x}$. By solving the Poisson equation for q with these two boundary conditions enforced, the resulting $\boldsymbol{m}$ projected by q can correctly reflect the gravity and surface tension effect.

For other external forces, such as buoyancy in Section 3.4, we add it directly to the impulse $\boldsymbol{m}$. After the boundary projection of gauge variable q for liquid and adding the external force, we obtain the final impulse variable at the end of each time step as $\boldsymbol{m}^{n+1}$.

4.6 Divergence-free projection

In the projection step, we project $\boldsymbol{m}^{n+1}$ to $\boldsymbol{u}^{n+1}$ by removing its non-divergence-free part $\nabla\varphi$. The numerical process is exactly the same as that in a standard advection-reflection solver. Specifically, we use a multigrid preconditioned conjugate gradient solver on GPU to accelerate the numerical solving of the Poisson equation. Note that $\boldsymbol{m}^{n+1}$ does not change and does not need to be divergence-free after the projection. The projected vector field is assigned to $\boldsymbol{u}^{n+1}$.

5 Time Integration

Our solver simulates smoke and liquid with steps as specified in Algorithms 1 and 2. We summarize the overall time splitting as follows:

- First, we reinitialize the impulse $\boldsymbol{m}$ by a weighted blending with the velocity $\boldsymbol{u}$ on the grid. This is to avoid the infinite growth of the impulse $\boldsymbol{m}$.
- Second, we perform an advection operation on $\boldsymbol{m}$ and other auxiliary quantities with the velocity field. We chose Semi-Lagrangian advection for smoke and PIC/FLIP advection for liquid simulations.
- Next, a stretching operation is performed on $\boldsymbol{m}$. Details about stretching are shown in Section 4.3.
- For viscosity, we add $\nu\nabla^2\boldsymbol{m}$ to impulse $\boldsymbol{m}$ implicitly with a red-black Gauss-Seidel solver.
- Only for fluid simulation, we add surface tension and gravity potential to q as specified in (11) on the boundary and pass solve the Poisson problem $\nabla^2 q = 0$ and apply the correction to $\boldsymbol{m}$ in order to smoothly pass in momentum from the boundary.
- Lastly, we use a standard pressure-projection method on $\boldsymbol{m}$ to get the final velocity $\boldsymbol{u}$ that is divergence-free. This is done by solving a Poisson equation constrained by Neumann and Dirichlet boundary conditions. Note that the impulse $\boldsymbol{m}$ itself is not changed at this step.

Algorithm 1 Simulation of Smoke with Buoyancy

Input: $\boldsymbol{u}^n$, $\boldsymbol{m}^n$, ρ'^n, Δt **Output:** $\boldsymbol{u}^{n+1}$, $\boldsymbol{m}^{n+1}$, ρ'^{n+1}

1: Reinitialization: $\boldsymbol{m}_r^n = r\boldsymbol{m}^n + (1-r)\boldsymbol{u}^n$ ▷ Sec.4.1
2: $\boldsymbol{m}_r^{n+1} \leftarrow$ SemiLagrangianAdvection$(\boldsymbol{u}^n, \boldsymbol{m}_r^n, \Delta t)$ ▷ Sec.4.2
3: $\rho'^{n+1} \leftarrow$ SemiLagrangianAdvection$(\boldsymbol{u}^n, \rho'^n, \Delta t)$
4: Implicit stretching: $\boldsymbol{m}_s^{n+1} = [\boldsymbol{I} + \Delta t(\boldsymbol{\nabla}\boldsymbol{u}^n)^T]^{-1}\boldsymbol{m}_r^{n+1}$ ▷ Sec.4.3
5: Implicit viscosity: $\boldsymbol{m}_v^{n+1} = [\boldsymbol{I} - \nu\Delta t\Delta_h^{-1}]^{-1}\boldsymbol{m}_s^{n+1}$ ▷ Sec.4.4
6: Apply buoyancy: $\boldsymbol{m}^{n+1} = \frac{\rho'^{n+1}}{\rho}g\Delta t + \boldsymbol{m}_v^{n+1}$ ▷ Sec.4.5
7: Solving φ ▷ Eq. (5)
8: Divergence projection: $\boldsymbol{u}^{n+1} = \boldsymbol{m}^{n+1} - \boldsymbol{\nabla}\varphi$ ▷ Sec.4.6

Algorithm 2 Simulation of Water with Surface Tension

Input: $\boldsymbol{u}^n$, $\boldsymbol{m}^n$, ϕ^n, Δt **Output:** $\boldsymbol{u}^{n+1}$, $\boldsymbol{m}^{n+1}$, ϕ^{n+1}

1: Reinitialization: $m_r^n = rm^n + (1-r)u^n$ ▷ Sec.4.1
2: Grid $\boldsymbol{m}$ and $\boldsymbol{u}$ to particle ▷ PIC/FLIP
3: Advection of the particles and the level-set ϕ
4: Particle $\boldsymbol{m}$, $\boldsymbol{u}$ to grid ▷ PIC/FLIP
5: Implicit stretching: $\boldsymbol{m}_s^{n+1} = [\boldsymbol{I} + \Delta t(\boldsymbol{\nabla}\boldsymbol{u}^n)^T]^{-1}\boldsymbol{m}_r^{n+1}$ ▷ Sec.4.3
6: Implicit viscosity: $\boldsymbol{m}_v^{n+1} = [\boldsymbol{I} - \nu\Delta t\Delta_h]^{-1}\boldsymbol{m}_s^{n+1}$ ▷ Sec.4.4
7: Solving q ▷ Eq. (8)
8: Boundary projection: $\boldsymbol{m}^{n+1} = \boldsymbol{m}_v^{n+1} - \boldsymbol{\nabla}q$ ▷ Sec.4.5
9: Solving φ ▷ Eq. (5)
10: Divergence projection: $\boldsymbol{u}^{n+1} = \boldsymbol{m}^{n+1} - \boldsymbol{\nabla}\varphi$ ▷ Sec.4.6

6 Results

First, we apply our method to several classical simulation setups to examine the correctness of our method. We then demonstrate our simulation results for smoke and liquid respectively. Within a single uniform framework, we are able to simulate various intriguing fluid phenomena including

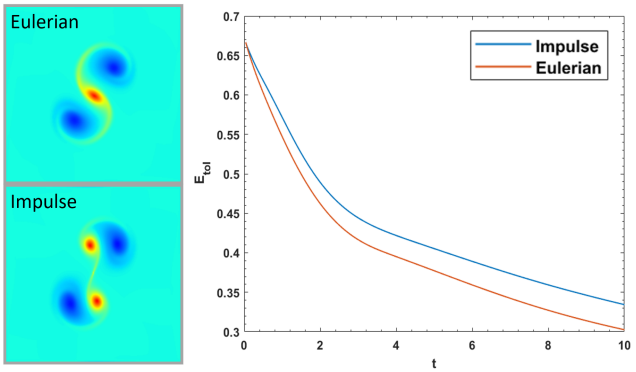


Fig. 2: Left: Visualization of the magnitude of vorticity in Taylor vortex simulation at $t = 8$. The result on the top is produced by the Eulerian method and the one on the bottom is by our method. Right: kinetic energy comparison between these two methods.

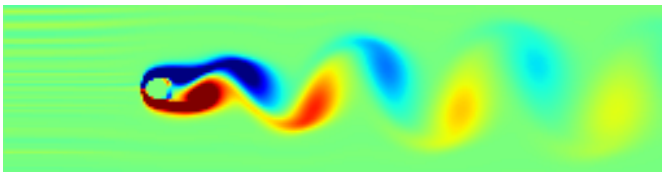


Fig. 3: Visualization of the magnitude of vorticity of Karman vortex street at t=20. Vortices of two different orientations are formed interchangeably with the same time intervals in between.

smoke collision, smoke-solid coupling, water surface oscillation and etc. Last, we show our method's ability to create more artistic effects by adjusting the damping parameter for stretching. Note that the Eulerian baseline we use for smoke simulation is based on [15] and the PIC/FLIP baseline we use for liquid simulation is based on [16].

6.1 Validations

We test our method on three simulation benchmarks as follows, from which we validate the correctness of our method.

In Figure 2, we compare our simulation result of Taylor vortex with that of Eulerian method, both with Semi-Lagrangian advection. We set up two vortices with an initial distance 0.81 in a 256×256 grid with domain size $2\pi \times 2\pi$. Our method not only successfully separates the two vortices and preserves two clear vortices structures, but also preserves kinetic energy better than the baseline method, as illustrated by the graph.

The second benchmark is a 2D Karman Vortex Street with the same setup as [56]. We place a circular obstacle on the left side with diameter $0.3m$ in a $8m \times 8m$ domain. The incoming source speed is $1m/s$ with random disturbance within $\pm 0.01m/s$. The grid resolution is 256×256 and the time step size is 0.001. We calculate the Strouhal number $St = \frac{f_q D}{\boldsymbol{u}_\infty}$ equals 0.161, which measures the frequency of the vortices formulation. This number falls into the range of 0.16–0.18 by previous researchers.

As shown in Figure 4, We first assign a disk in space with a density slightly higher ($\rho' = 0.1$) than the surrounding

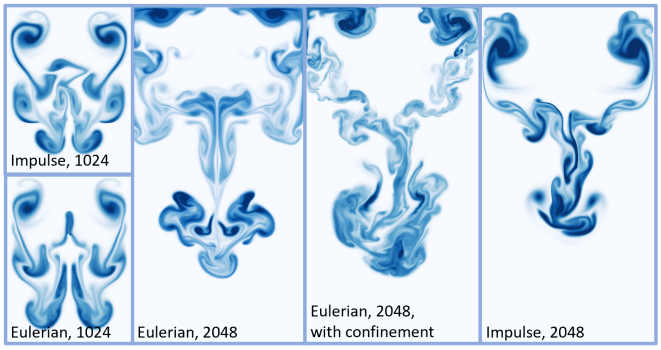


Fig. 4: 2D Ink diffusion: An ink drop spreads out underwater in 2D space. Numbers in the picture denote the grid resolution of simulation height.

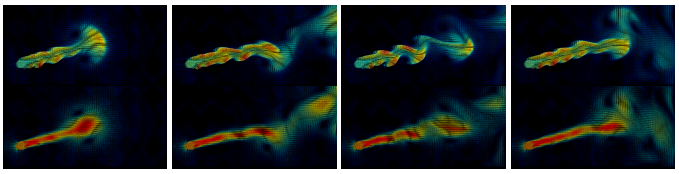


Fig. 5: Top row: The impulse field of a 2d plume example at t=2, t=4, t=8, and t=12s; Bottom row: the corresponding velocity field. Values are mapped to colors.

area as the initial space occupied by ink. Then, the cells with higher density drop down and spread out due to slightly higher gravitational potential than the surrounding areas. With impulse transferring momentum across the entire field, our simulation is able to preserve a more clear vortex structure while the overall shape is similar to the result of the Eulerian method with high resolution.

6.2 Smoke simulations

We first show a simple 2D plume example with visualizations of the impulse field to better demonstrate the evolution of the impulse field and its relationship to the velocity field. As shown in Figure 5, we simulated plume spurting in a 256×128 grid. The impulse vector field forms a bulging pattern thus assisting the formulation and preservation of vortices structures in the velocity field.

As shown in Figure 7, we set a smoke source at the bottom left corner of a $256 \times 128 \times 128$ grid. The smoke with initial velocity spurts out from the pipe and forms a plume blob. We compare our method with the standard semi-Lagrangian advection and projection with vorticity confinement method [15], the McCormack advection [40] with vorticity confinement method, and a vortex method (IVOCK [34]) in a 3D plume example. We first compare the visual effects regarding the vortical details. The vorticity confinement method can preserve large-scale vorticies. The simulation produced by the McCormack advection produces smaller-scale vortical features yet at the same time manifests some unnatural, high-frequency noise. The vortex method can preserve small and medium sized vortices at the beginning but produces some small-scale noises in the later frames. Our impulse method can preserve vortices at different scales. Next, we compare the performance between

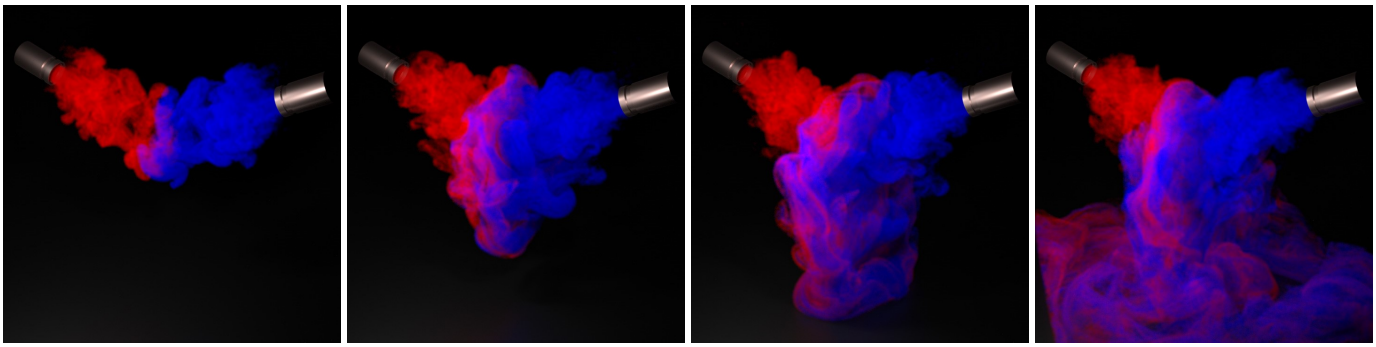


Fig. 6: Impinging Jets: two jets of smoke collide with each other, forming a colorful bulging volume decorated by vortices in different scales due to intense momentum interactions in between.

Fig. 7: Plume: These scenes show how smoke bursts out from a pipe with small diameter by forming a mushroom-like trajectory due to the intense momentum exchange with the surrounding air. First row: standard semi-Lagrangian advection and projection with vorticity confinement. Second row: McCormack advection and projection with vorticity confinement. Third row: IVOCK with grid resolution 384x192x192. Last row: our method. We select frame 65 and 165 for all methods.

Fig. 8: The plume example with different r values: 0.0, 0.5, 0.9, 0.97 (from left to right, top to bottom)

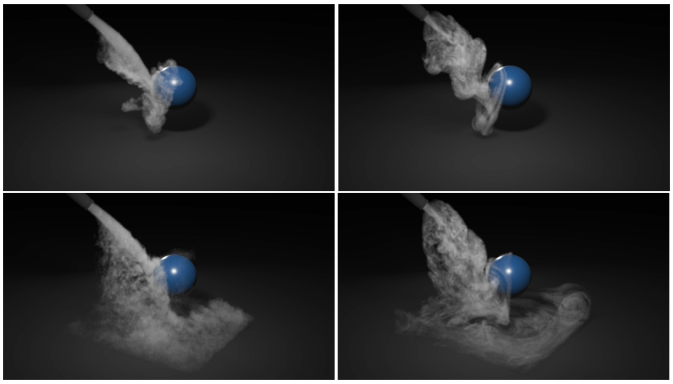


Fig. 9: Dry ice: this scene shows how smoke collides onto a ball at a tilted angle under gravity. Turbulence was generated immediately and formed a chaotic thick smoke layer. When colliding with the ball, a thin layer of smoke starts to crawl around the surface. Left column: simulation by Eulerian method with vorticity confinement. Right column: simulation by our method.

our approach and the vortex method. The average time costs of the first ten frames for our method on advection (including reinitialization), stretching, viscosity, and projection are 617ms (16%), 702 ms (18%), 625ms (16%), 1983ms (50%) respectively. The average time costs for the vortex method with the same resolution on advection, stretching, and vorticity-to-velocity projection (in IVOCK the projection step includes four Poisson solves: three for vorticity increments and one for velocity) are 1330ms (13%), 397ms (3.8%), and 8719ms (83.7%) respectively. Our solver is implemented in double precision with a GPU Poisson solver while the IVOCK vortex method is in float precision and its Poisson solver is on CPU. Both experiments were run on a AMD Ryzen7 3800x-8core-processor with 64 GB memory and NVIDIA GEFORCE RTX 2070 GPU. In contrast to the vortex method, whose vorticity-to-velocity projection

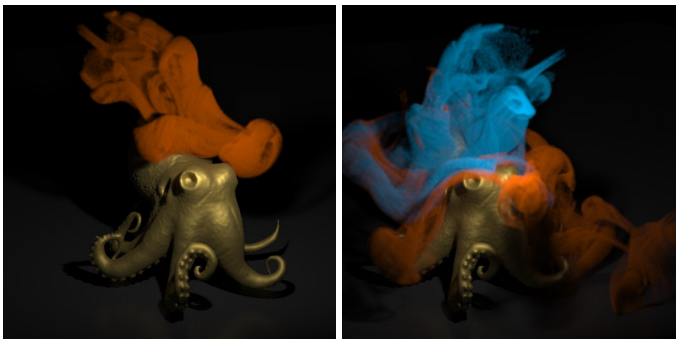


Fig. 10: Smoke on octopus: two jets of smoke collide with the octopus model.

constitutes the major time cost of each simulation step, our impulse solver spent about 50% of the total computation time (equally) on the steps of advection, stretching, and viscosity.

Using the same setup as the plume example, we also conducted a parameter study of the reinitialization coefficient r. As shown in Figure 8, when r is closer to 0, the plume behavior resembles more the result of the traditional Eulerian method, where not many vortices are preserved; when r is closer to 1, the result resembles more the look of plume smoke produced by Eulerian method with vorticity confinement.

As shown in Figure 6, we set up two streaks of smoke in blue and red color spurt from pipes and collide with each other within a grid with dimension $256 \times 256 \times 256$. In this process, impulse transmits the momentum from one streak of smoke to the other and the other way around. Eventually they form the dynamic mix of two colors with bulging form.

We demonstrate flow with a solid boundary in Figure 9. Within a $256 \times 256 \times 256$ grid, we set up a smoke source with a constant velocity in the upper left corner. The effective density of the source is $\rho' = 0.1$, which makes the smoke slowly drop down and flow away. Turbulence developed at a fast speed before hitting the ball, then generated a layer of smoke with a clear vortex structure wrapping up the ball with a continuous supplement of the smoke. Compared with the results of the Eulerian method with vorticity confinement, our method generates more realistic-looking smoke structures. We also performed an experiment with a more complex solid boundary (as shown in Figure 10) to demonstrate the flow around complex solid boundary. We can observe complex vortical structures developing around the octopus boundary.

For all smoke simulations, we use reinitialization coefficient $r = 0.97$ and kinematic viscosity coefficient $\nu = 1e-6$. We use $cfl = 0.6$ for the plume and dry ice example and $cfl = 0.5$ for the impinging smoke example.

6.3 Water simulations

We show three examples water simulations with surface tension. In all these examples, we use the PIC/FLIP method for advection, a level-set to track the free surface, and solve q to handle the pressure jump caused by surface tension. We set $cfl = 0.5$ for all the examples.

As shown in Figure 11, we set up a water tank and let one droplet fall into the tank within a $128 \times 128 \times 128$ grid. Driven by gravitational pull, the external droplet introduces large momentum to the bulk water on the bottom. Thus, a large hole formed as our method is able to transport momentum deep down into the water body.

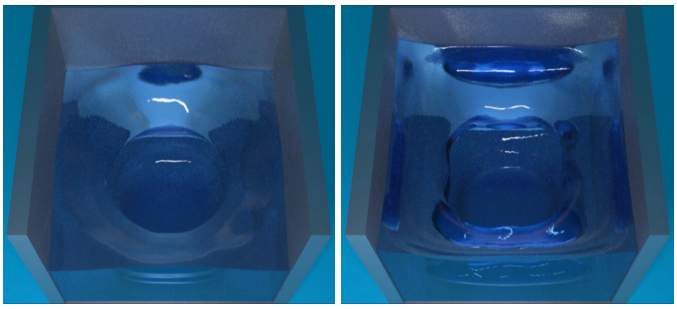


Fig. 11: Water tank: A droplet with a high initial speed falls into the tank, causing intensive surface fluctuation and momentum exchange underneath the water. The surface tension is $200mN/m$. From left to right the simulation is at frames 15, 33 respectively.

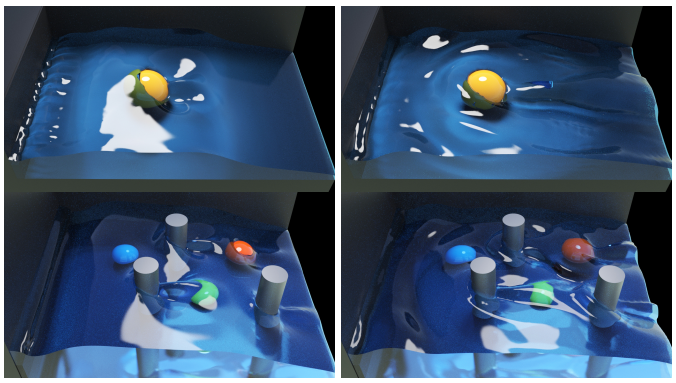


Fig. 12: Top row: Stream passing over a sphere obstacle. Grid resolution: $128 \times 80 \times 48$. Bottom row: Stream passing over multiple obstacles. Grid resolution: $160 \times 120 \times 60$. The surface tension of both these two examples is $500mN/m$.

As shown in Figure 12, we show two examples of the flowing stream. Same for both examples, we place left, front, back, and bottom walls, and an incoming source on the left wall. For the first example, one sphere obstacle is placed in the middle region of the stream, whereas multiple obstacles with different geometries are placed in the second example. Here, our method is able to generate the detailed turbulence on water surface and vortices behind the obstacles.

For the stream with a sphere obstacle example, the typical time spent on re-initialization, advection, particle-to-grid, grid-to-particle, stretching, viscosity, boundary projection and divergence projection steps are 2 ms, 208 ms, 1247 ms, 137 ms, 160 ms, 158 ms, 102 ms and 119 ms respectively for each time step on a 20-core serverwith Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz, Tesla V100 GPU and 256 GB RDIMM DDR4 memory. Our method adds 23% overhead than the baseline PIC/FLIP method in this case. This experiment is performed with the same hardware setup as mentioned for the smoke example.

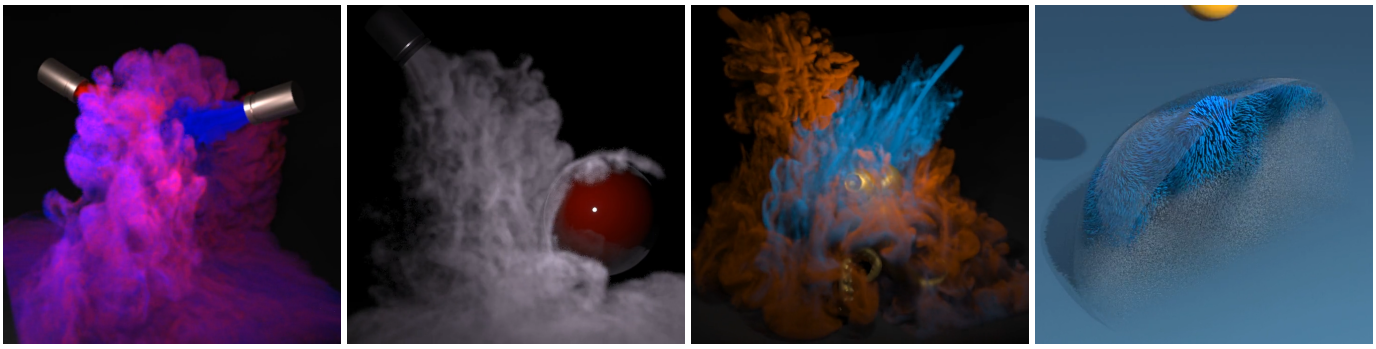


Fig. 13: Simulations of turbulent flow with small vortical structures controlled by the stretching parameter. Column 1: Impinging jets with $\alpha = 0.21$; Column 2: Simulation of dry ice $\alpha = 0.21$; Column 3: Jets with an octopus obstacle with $\alpha = 0.25$; Column 4: A ball hitting on a droplet with $\alpha = 0.3$ (cross-section view, impulse visualized as segments).

6.4 Stretching parameters

Since the stretching term can sometimes lead to instability in the simulation with a large CFL number, we also experiment with a damped stretching as $\boldsymbol{m}_s^{n+1} = [\boldsymbol{I} + \alpha\Delta t(\boldsymbol{\nabla u}^n)^T]^{-1}\boldsymbol{m}_r^{n+1}$, where α is the stretching damping coefficient. We show in this section that by using a small stretching coefficient α, we can obtain visually interesting vortical effects with different strengths and sizes. For smoke simulations, the damped stretching can instantaneously produce numerous small vortices structures that highlight the turbulence of the fluid. For liquid simulations, we also observe small vortices formed inside the liquid. We hypothesize that by weakening the stretching effect, the advection effect is relatively large thus small vortices are separated from larger vortices.

As shown in Figure 13 (column 1), we use the same settings as the example of impinging jets in Figure 6, with a smaller stretching coefficient $\alpha = 0.21$. As shown in Figure 13 (column 2), we use the same settings as in the dry ice example in Figure 9, with a smaller stretching coefficient $\alpha = 0.21$. Both in the impinging jets and dry ice examples, the results of using small α have similar properties as that using $\alpha = 1$, while small α can generate and preserve abundant small-scale vortical structures and the results with $\alpha = 1$ demonstrate vortices preserves a bigger volume. A shown in Figure 13 (column 3), the orange and blue smoke shoot from air interchangeably with gradually increasing initial velocities. We placed a model of an octopus on the floor to showcase the beautiful interaction of smoke with geometry in an organic form. Numerous small-scale vortices near the smoke-air layer were created due to the momentum transformation with the octopus and surrounding mixture of air and smoke. As shown in Figure 13 (column 4), the yellow ball pushes the surface of the droplet and bounces back; the lengths of the segments inside the droplet indicate the magnitude of the velocity of each particle, and blue color demonstrates the strength of the impulse carried by the PIC/FLIP particles.Here, the impulse of the strike can be transformed towards the inside of the droplet and forms small vortexes in the three spatial directions, instead of bouncing around at the fluid-air interface. Also, as indicated by the length of the internal segments, our method preserves momentum for an extended period of time.

7 Discussion

By observing a host of simulations with our impulse method, we conclude the following characteristics from the simulations that our solver can produce. First, our impulse method can produce physically accurate simulations as demonstrated in a series of validation tests (see Figure 2 for Taylor vortex evolution, Figure 3 for frequency measurement of the Karman vortex street, Figure 4 for ink drop comparison with Euler method). Second, our method embraces better fluid dynamics and makes the transfer of momentum easier inside the fluid. In the impinging jet (Figure 6) and dry ice (Figure 9) examples, the simulations exhibit thin and coherent flow structures when interacting with the solid boundaries, which was not feasible for standard advection-projection methods. By decoupling the impulse variable from the fluid pressure, the impulse will not accumulate the compulsory correction at each step, and hence can evolve in a more unrestricted fashion over time (see Figure 5 for the visualization of impulse-velocity in 2D). Third, our method preserves vorticity structures on the long run as shown in our plume (Figure 7), dry ice (Figure 9), and impinging jet (Figure 6) examples. In addition, our approach also provide a tunable parameter to control the strength of stretching, which further allows artistic flexibility in creating turbulent flow details with different scales of vortices.

Compared with its original Lagrangian form developed in [17], our version incorporated modern gauge techniques (e.g., the harmonic q) to enable the simulation of interfacial flow phenomena. In addition, our numerical solver is built on an Eulerian discretization, which inherently connects to many existing large-scale, grid-based simulation methods. We also demonstrate the efficacy of our impulse model in the hybrid Eulerian-Lagrangian (PIC/FLIP) simulation framework. Compared to other gauge methods such as [14], our impulse-based form enjoys its Lagrangian nature, making it convenient to devise Lagrangian advection schemes (e.g., we developed all our liquid simulations using PIC/FLIP method). Compared to Clebsch gauge method, our method naturally deals with the viscosity term.

8 Conclusion and Future Work

We propose an impulse-based fluid simulation framework by reformulating the Navier–Stokes equation into a velocity-impulse format. By introducing the impulse variable to the

evolution of the fluid, fluid dynamics can be improved and vorticity details can be preserved for longer. We show our method as highly adaptive to simulate various kinds of fluid phenomena (smoke, free-surface flow and hybrid particle/grid, etc). Our solver is so simple that decorating it with higher-order advection/projection schemes is simply straightforward. Moreover, this newly involved impulse variable offers more flexibility at handling complex boundary conditions that are previously hard to capture. This benefit is also one of our major concerns when choosing the appropriate gauge variable ,for instance, how to directly apply a local force or impulse at the surface of the fluid. Based on our observation, representing this disturbance by pressure jump is not a proper way. This indirect form of transforming momentum will be smoothed out quickly near the interface and hardly can be transformed inward. Applying the disturbance to the fluid velocity directly will break the incompressibility constraint, which is also infeasible. In our work, we can manipulate the impulse directly whose divergence has no constraint. We find that momentum can be transformed into and sustained inside the fluid in the form of vorticity. We validate the effectiveness of our method by simulating a series of complex fluid phenomena and comparing it with the baseline methods. Our showcases varying from smoke, ink, dry ice to free surface fluid with strong and weak surface tension demonstrate the generalizability of our method. Our method does have some limitations compared with the traditional advection-projection method. First, by introducing a new state variable, an additional projection step might be necessary to reflect the effect of some specific boundary conditions. Second, a new stretching step that does not exist in the Eulerian method appears by reformulating the Navier-Stokes equations. This operation needs careful treatment and can easily become unstable if the system is under diffused or the CFL number is not set appropriately. This may be caused by calculating the stretching term on the Cartesian grid and having velocity defined on the cell faces. It is not very accurate to interpolate the impulse variable to the cell centers, and interpolate the delta impulse field back to the cell faces. In the future, we hope to use a surface-only approach by using surface mesh or particles to evolve the fluid, with boundary conditions being directly enforced on the surface.

REFERENCES

[1] D. M. Summers and A. J. Chorin, "Numerical vorticity creation based on impulse conservation," vol. 93, no. 5, pp. 1881–1885, 1996.

[2] G.-H. Cottet, P. D. Koumoutsakos *et al.*, *Vortex methods: theory and practice*. Cambridge university press Cambridge, 2000, vol. 8.

[3] S. Park and M. Kim, "Vortex fluid for gaseous phenomena," in *SCA*, 2005.

[4] S. Weißmann and U. Pinkall, "Filament-based smoke with vortex shedding and variational reconnection," *ACM Transactions on Graphics*, vol. 29, pp. 115:1–115:12, 2010.

[5] M. J. Stock, W. J. Dahm, and G. Tryggvason, "Impact of a vortex ring on a density interface using a regularized inviscid vortex sheet method," *Journal of Computational Physics*, vol. 227, no. 21, pp. 9021–9043, 2008.

[6] X. Zhang, R. Bridson, and C. Greif, "Restoring the missing vorticity in advection-projection fluid solvers," *ACM Transactions on Graphics*, vol. 34, pp. 1 – 8, 2015.

[7] "An impulse-based approximation of fluid motion due to boundary forces," *Journal of Computational Physics*, vol. 123, no. 2, pp. 341–353, 1996.

[8] T. F. Buttke, "Velicity methods: Lagrangian numerical methods which preserve the hamiltonian structure of incompressible fluid flow," in *Vortex flows and related numerical methods*. Springer, 1993, pp. 39–57.

[9] "Turbulence calculations in magnetization variables," *Applied Numerical Mathematics*, vol. 12, no. 1, pp. 47 – 54, 1993, sPECIAL ISSUE.

[10] J. H. Maddocks and R. L. Pego, "An unconstrained hamiltonian formulation for incompressible fluid flow," *Communications in Mathematical Physics*, no. 1, pp. 207–217.

[11] "A class of fully second order accurate projection methods for solving the incompressible navier–stokes equations," *Journal of Computational Physics*, vol. 200, no. 1, pp. 325 – 346, 2004.

[12] P. H. Roberts, "A hamiltonian theory for weakly interacting vortices," *Mathematika*, vol. 19, no. 2, p. 169–179, 1972.

[13]

[14] R. Saye, "Interfacial gauge methods for incompressible fluid dynamics," *Science Advances*, vol. 2, p. e1501869, 2016.

[15] R. Fedkiw, J. Stam, and H. Jensen, "Visual simulation of smoke," *ACM SIGGRAPH Papers*, 2001.

[16] Y. Zhu and R. Bridson, "Animating sand as a fluid," *ACM Transactions on Graphics*, vol. 24, pp. 965–972, 2005.

[17] R. Cortez, "Impulse-based particle methods for fluid flow," Ph.D. dissertation, University of California, Berkeley, 1995.

[18] ——, "A vortex/impulse method for immersed boundary motion in high reynolds number flows," *Journal of Computational Physics*, vol. 160, pp. 385–400, 2000.

[19] G. A. Kuz'min, "Ideal incompressible hydrodynamics in terms of the vortex momentum density," *Physics Letters A*, vol. 96, no. 2, pp. 88–90, 1983.

[20] G. Winckelmans, "Vortex methods," *Encyclopedia of computational mechanics*, 2004.

[21] R. Cortez, "On the accuracy of impulse methods for fluid flow," *SIAM Journal on Scientific Computing*, vol. 19, no. 4, pp. 1290–1302, 1998.

[22] "A representation of bounded viscous flow based on hodge decomposition of wall impulse," *Journal of Computational Physics*, vol. 158, no. 1, pp. 28 – 50, 2000.

[23] "Finite difference schemes for incompressible flows in the velocity–impulse density formulation," *Journal of Computational Physics*, vol. 130, no. 1, pp. 67 – 76, 1997.

[24] "Implicit mesh discontinuous galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part i," *Journal of Computational Physics*, vol. 344, pp. 647 – 682, 2017.

[25] "Implicit mesh discontinuous galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part ii," *Journal of Computational Physics*, vol. 344, pp. 683 – 723, 2017.

[26] S. Yang, S. Xiong, Y. Zhang, F. Feng, J. Liu, and B. Zhu, "Clebsch gauge fluid," vol. 40, no. 4, 2021.

[27] S. Weißmann and U. Pinkall, "Filament-based smoke with vortex shedding and variational reconnection," *ACM Transactions on Graphics*, vol. 29, p. 115, 2010.

[28] T. Pfaff, N. Thuerey, and M. Gross, "Lagrangian vortex sheets for animating fluids," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 1–8, 2012.

[29] T. Brochu, T. Keeler, and R. Bridson, "Linear-time smoke animation with vortex sheet meshes," in *SCA*, 2012, pp. 87–95.

[30] A. Leonard, "Vortex methods for flow simulation," *Journal of Computational Physics*, vol. 37, no. 3, pp. 289–335, 1980.

[31] A. Angelidis and F. Neyret, "Simulation of smoke based on vortex filament primitives," in *SCA*, 2005, pp. 87–96.

[32] P. Moin, A. Leonard, and J. Kim, "Evolution of a curved vortex filament into a vortex ring," *Physics of Fluids*, vol. 29, pp. 955–963, 1986.

[33] S. Elcott, Y. Tong, E. Kanso, P. Schröder, and M. Desbrun, "Stable, circulation-preserving, simplicial fluids," *ACM Transactions on Graphics*, vol. 26, no. 1, 2007.

[34] X. Zhang, R. Bridson, and C. Greif, "Restoring the missing vorticity in advection-projection fluid solvers," *ACM Transactions on Graphics*, vol. 34, no. 4, pp. 1–8, 2015.

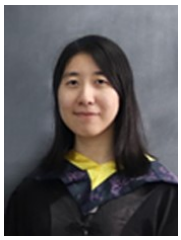
[35] P. Koumoutsakos, G.-H. Cottet, and D. Rossinelli, "Flow simulations using particles-bridging computer graphics and cfd," in *ACM SIGGRAPH Papers*. ACM, 2008, pp. 1–73.

[36] A. Selle, N. Rasmussen, and R. Fedkiw, "A vortex particle method for smoke, water and explosions," in *ACM SIGGRAPH Papers*, 2005, pp. 910–914.
[37] D. Kim, O.-Y. Song, and H. Ko, "Stretching and wiggling liquids," *ACM SIGGRAPH Asia Papers*, 2009.
[38] M. N. Gamito, P. F. Lopes, and M. R. Gomes, "Two-dimensional simulation of gaseous phenomena using vortex particles," in *Computer Animation and Simulation*. Springer, 1995, pp. 3–15.
[39] B. Kim, Y. Liu, I. Llamas, and J. Rossignac, "Flowfixer: Using bfecc for fluid simulation," in *NPH*, 2005.
[40] A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac, "An unconditionally stable maccormack method," *Journal of Scientific Computing*, vol. 35, pp. 350–371, 2008.
[41] Z. Qu, X. Zhang, M. Gao, C. Jiang, and B. Chen, "Efficient and conservative fluids using bidirectional mapping," *ACM Transactions on Graphics*, vol. 38, no. 4, 2019.
[42] P. Mullen, K. Crane, D. Pavlov, Y. Tong, and M. Desbrun, "Energy-preserving integrators for fluid animation," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 1–8, 2009.
[43] Y. Zhu and R. Bridson, "Animating sand as a fluid," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 965–972, 2005.
[44] J. U. Brackbill and H. M. Ruppel, "Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions," *Journal of Computational Physics*, vol. 65, no. 2, pp. 314–343, 1986.
[45] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin, "The affine particle-in-cell method," *ACM Transactions on Graphics*, vol. 34, pp. 1 – 10, 2015.
[46] C. Fu, Q. Guo, T. F. Gast, C. Jiang, and J. Teran, "A polynomial particle-in-cell method," *ACM Transactions on Graphics*, vol. 36, pp. 1 – 12, 2017.
[47] S. Gagniere, D. Hyde, A. Marquez-Razon, C. Jiang, Z. Ge, X. Han, Q. Guo, and J. Teran, "A hybrid lagrangian/eulerian collocated advection and projection method for fluid simulation," *ArXiv*, vol. abs/2003.12227, 2020.
[48] L. Boyd and R. Bridson, "Multiflip for energetic two-phase fluid simulation," *ACM Transactions on Graphics*, vol. 31, no. 2, pp. 1–12, 2012.
[49] T. Kim, N. Thürey, D. James, and M. Gross, "Wavelet turbulence for fluid simulation," *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 1–6, 2008.
[50] R. Bridson, J. Houriham, and M. Nordenstam, "Curl-noise for procedural fluid flow," *ACM Transactions on Graphics*, vol. 26, no. 3, 2007.
[51] A. Chern, F. Knöppel, U. Pinkall, P. Schröder, and S. Weißmann, "Schrödinger's smoke," *ACM Transactions on Graphics*, vol. 35, p. 77, 2016.
[52] R. S. Rogallo, *Numerical experiments in homogeneous turbulence*. National Aeronautics and Space Administration, 1981, vol. 81315.
[53] M. Padilla, A. Chern, F. Knöppel, U. Pinkall, and P. Schröder, "On bubble rings and ink chandeliers," *ACM Transactions on Graphics*, vol. 38, no. 4, 2019.
[54] J. Stam, "Stable fluids," in *ACM SIGGRAPH Papers*, 1999.
[55] R. Bridson, *Fluid simulation for computer graphics*. CRC press, 2015.
[56] M.-C. Lai and C. S. Peskin, "An immersed boundary method with formal second-order accuracy and reduced numerical viscosity," *Journal of computational Physics*, vol. 160, no. 2, pp. 705–719, 2000.

Fan Feng Fan Feng is a first-year Ph.D candidate advised by Prof. Bo Zhu in the VCL Lab at Dartmouth College. She received B.S in computer science and B.A in Mathematics from University of North Carolina at Chapel Hill. Her research interests include computer graphics, physics-based fluid simulation, topology optimization, and Scientific Machine Learning.

Jinyuan Liu Jinyuan Liu received the BEng degree in astronautical engineering in 2015 from Beihang University, Beijing, China and the M.S degree in aeronautics & astronautics in 2017 from Stanford university. He is currently a PhD in computer science at Dartmouth College, advised by Prof. Bo Zhu. His research is focused on physically-based modeling and multi-physics simulation, including fluid, deformable bodies, solid-fluid interaction and topology optimization.

Shiying Xiong Shiying Xiong is a Post-Doctoral researcher working with Prof. Bo Zhu in the VCL Lab at Dartmouth College. Before starting his Postdoc, he obtained a Ph.D. in the College of Engineering, Peking University in June 2019 and advised by Prof. Yue Yang. His research interests include Hamiltonian Fluid Mechanics, Vortex Dynamics, Computational Physics, and Scientific Machine Learning.



Shuqi Yang Shuqi Yang studied for her Master's degree at Dartmouth College and for her Bachelor's degree at Dalian University of Technology. Her research interests mainly focus on physical simulation and machine learning.

Yaorui Zhang Yaorui Zhang received her MS in Computer Science in June 2021 from Dartmouth College, and a B.E. in Digital Media in 2019 from Zhejiang University, China. Her research is focused on simulating and controlling soft dynamic systems, as well as creating interactive design interfaces.

Bo Zhu Bo Zhu is an assistant professor of Computer Science at Dartmouth College. Prior to joining Dartmouth, he obtained his Ph.D. from Stanford and conducted his postdoctoral research at MIT CSAIL. His research interests include computer graphics, computational physics, computational fluid dynamics, and scientific machine learning. He mainly focuses on building computational approaches to simulate complex fluid systems.