Efficient and Conservative Fluids Using Bidirectional Mapping

ZIYIN QU*, AICFVE and University of Pennsylvania
XINXIN ZHANG*, AICFVE
MING GAO, University of Pennsylvania
CHENFANFU JIANG, University of Pennsylvania
BAOQUAN CHEN, Peking University

In this paper, we introduce BiMocq^2, an unconditionally stable, pure Eulerian-based advection scheme to efficiently preserve the advection accuracy of all physical quantities for long-term fluid simulations. Our approach is built upon the method of characteristic mapping (MCM). Instead of the costly evaluation of the temporal characteristic integral, we evolve the mapping function itself by solving an advection equation for the mappings. Dual mesh characteristics (DMC) method is adopted to more accurately update the mapping. Furthermore, to avoid visual artifacts like instant blur and temporal inconsistency introduced by re-initialization, we introduce multi-level mapping and back and forth error compensation. We conduct comprehensive 2D and 3D benchmark experiments to compare against alternative advection schemes. In particular, for the vortical flow and level set experiments, our method outperforms almost all state-of-art hybrid schemes, including FLIP, PolyPic and Particle-Level-Set, at the cost of only two Semi-Lagrangian advections. Additionally, our method does not rely on the particle-grid transfer operations, leading to a highly parallelizable pipeline. As a result, more than 45× performance acceleration can be achieved via even a straightforward porting of the code from CPU to GPU.

CCS Concepts: • **Computing methodologies** → **Continuous models**; **Physical simulation**.

Additional Key Words and Phrases: Fluid Simulation, Conservative Advection, MCM,

ACM Reference Format:
Ziyin Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. 2019. Efficient and Conservative Fluids Using Bidirectional Mapping. *ACM Trans. Graph.* 38, 4, Article 128 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3322945

1 INTRODUCTION

Eulerian based fluid simulations have achieved great success in reproducing a wide range of phenomena in computer graphics, such as liquids [Aanjaneya et al. 2017; Enright et al. 2002b; Foster and Fedkiw 2001], smoke and fire [Fedkiw et al. 2001; Nguyen et al. 2002; Rasmussen et al. 2003; Setaluri et al. 2014]. The framework to solve the Navier-Stokes equation in a time splitting manner, especially the critical insight to treat the nonlinear advection term using Semi-Lagrangian scheme [Stam 1999], results in its most significant advantage of stability, hence ease of control, along with its biggest criticism: the numerical diffusion. For more details about the background, we refer to the book by Bridson [2008].

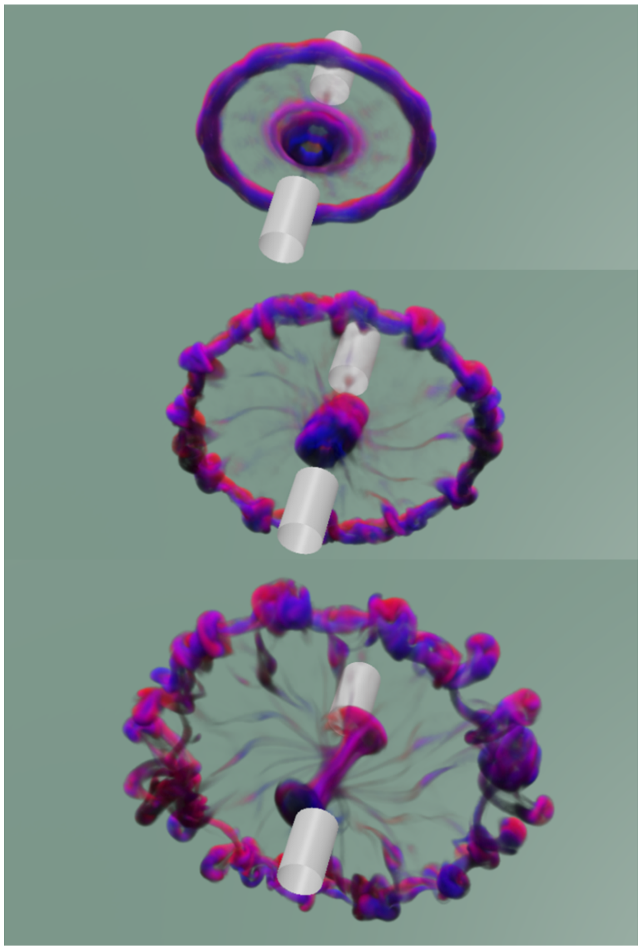
Fig. 1. Frames of vortex ring colliding at Re=2000 simulated using BiMocq^2, from top to bottom: frame 80, frame 140, and frame 280. The proposed method reproduces the whole process where colliding vortex rings stretch and form the out-shooting jets from the side of the rings.

*Z. Qu, X. Zhang are joint first authors.

Authors' addresses: Ziyin Qu, AICFVE and University of Pennsylvania, ziyinq@seas.upenn.edu; Xinxin Zhang, AICFVE, zhangshinshin@gmail.com; Ming Gao, University of Pennsylvania, minggao@seas.upenn.edu; Chenfanfu Jiang, University of Pennsylvania, cffjiang@seas.upenn.edu; Baoquan Chen, Peking University, baoquan@pku.edu.cn.

Eulerian methods start from the incompressible Navier-Stokes equations,

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} &= -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{f} \\ \nabla\cdot\mathbf{u} &= 0 \end{aligned} \tag{1}$$

where $\mathbf{u}$, p, ρ, ν, $\mathbf{f}$ represent velocity, pressure, density, kinematic viscosity and external forces such as gravity. We adopt the standard staggered MAC grid discretization. In the time-splitting framework, the fluid quantities are first advected by solving

$$\frac{\mathbf{D}\mathbf{u}}{\mathbf{D}t} = 0 \tag{2}$$

to obtain an intermediate velocity field $\tilde{\mathbf{u}}$. Boundary conditions, fluid emission, and extra controls can then be enforced along with the pressure projection step to obtain the velocity field $\mathbf{u}^{n+1}$ of the next time step, with the desired target divergence. Notice the physical states are updated solely based on the information from the last time step. With repeated Semi-Lagrangian advection, numerical diffusion keeps being accumulated, degrading the prediction accuracy of the solver.

On the other hand, in particle-grid hybrid methods, e.g. [Fu et al. 2017; Zhu and Bridson 2005], the particles carry the fluid quantities while the momentum equation remains to be solved on the background grid. Those hybrid methods require additional particle-grid transfer routines, which are usually parallel-unfriendly, presenting challenges to high-performance implementations, as has been pointed out recently by [Ferstl et al. 2016].

In contrast to the two existing popular strategies, our method represents the fluid state at time $t = T$ as the temporal integral of its material derivative from $t = 0$ along the trajectory, as can be directly derived from Eqn. (1) [Tessendorf and Pelfrey 2011].

$$\phi(\mathbf{x}(T), T) = \phi(\mathbf{x}(t_0), t_0) + \int_{t_0}^{T} \frac{\mathbf{D}\phi}{\mathbf{D}t}(\mathbf{x}(\tau), \tau)d\tau, \tag{3}$$

where ϕ denotes convective fluid quantities such as velocity, density, temperature, fuel, level-sets, etc.; while $\frac{D\phi}{Dt}$ denotes the rate of change of the quantity, as often due to acceleration, emission, or combustion reaction.

One observation from Eqn. (3) is that when there is no external influences (e.g., no acceleration/emission), the fluid quantity at $t = T$ should retain exactly the same value as it originally possessed. As a result, if we keep an image of the fluid state at the initial time t_0, along with a backward mapping

$$\mathcal{X}(\mathbf{x}(T)) : \mathbf{x}(T) \rightarrow \mathbf{x}(t_0), \tag{4}$$

which maps a spatial point $\mathbf{x}(T)$ back to its position at t_0, it would be possible to acquire the exact state, avoiding the accumulated numerical diffusion.

Following [Sato et al. 2017], given a particular degree-of-freedom $\mathbf{x}(T)$, the mapping can be found by integrating the trajectory back in time assuming the availability of the velocity field at every single time step:

$$\begin{aligned} \mathbf{x}(t_0) &= \mathcal{X}(\mathbf{x}(T)) \\ &= \mathbf{x}(T) - \int_{t_0}^{T} \mathbf{u}(\mathbf{x}(\tau), \tau)\, d\tau \end{aligned} \tag{5}$$

By substituting Eqn. (5) into Eqn. (3) and taking $\mathbf{u}$ as the convective quantity, it gives the momentum equation as

$$\begin{aligned} \mathbf{u}(\mathbf{x}(T), T) = \mathbf{u}\Big(&\mathbf{x}(T) - \int_{t_0}^{T} \mathbf{u}(\mathbf{x}(\tau), \tau)\, d\tau, t_0\Big) \\ &+ \int_{t_0}^{T} \frac{D\mathbf{u}}{Dt}(\mathbf{x}(\tau), \tau)\, d\tau \end{aligned} \tag{6}$$

In the case of $T = t_0 + \Delta t$, it is equivalent to the standard Semi-Lagrangian advection. Direct evaluation of this integral in Eqn. (5) is impractical since we need to explicitly store and access the fluid states for every single time step. As mentioned in [Sato et al. 2017], with explicit tracking, it becomes intractable for high-resolution simulations. Instead, we choose to dynamically advect the mapping $\mathcal{X}$ itself as the simulation proceeds to achieve high computational efficiency.

While the backward mapping Eqn. (4) provides sufficient information to predict the temporal states for pure advection, a practical advection scheme has to also take external influences into consideration. Previous methods, e.g. [Sato et al. 2017; Tessendorf and Pelfrey 2011], directly evaluate the integral Eqn. (6) back in time, which again has proven to be inefficient. We extend our idea of the backward mapping to introduce a second mapping - the forward mapping

$$\mathcal{Y}(\mathbf{x}(t_0)) : \mathbf{x}(t_0) \rightarrow \mathbf{x}(T) \tag{7}$$

to facilitate the tracking of the changes in the flow due to external interference. This forward mapping can be efficiently evolved by solving a partial differential equation.

Nevertheless, the two mappings could quickly become too distorted to stay effective, due to the possible intense stretching/shearing of the underlying geometry in drastically deforming scenarios. We further improve our backward and forward mappings by proposing a remeshing and long-term error correction schemes.

In summary, we propose a novel approach BiMocq^2 (n levels of Bi-directional mapping of convective quantities) for conservative long-term advection of fluid quantities, with the following features:

- The proposed scheme is unconditionally stable, allowing arbitrary large Δt for the simulation (even with CFL > 30 in some cases).
- The advection scheme is purely Eulerian, which can be easily parallelized to achieve 45× performance boost with a straightforward and simple GPU implementation.
- We track multi-level mappings to improve both the sharpness and temporal coherence of the simulation while maintaining computational efficiency.
- Based on the mapping functions, we propose long-term error correction schemes to improve the visual quality of the simulation further.

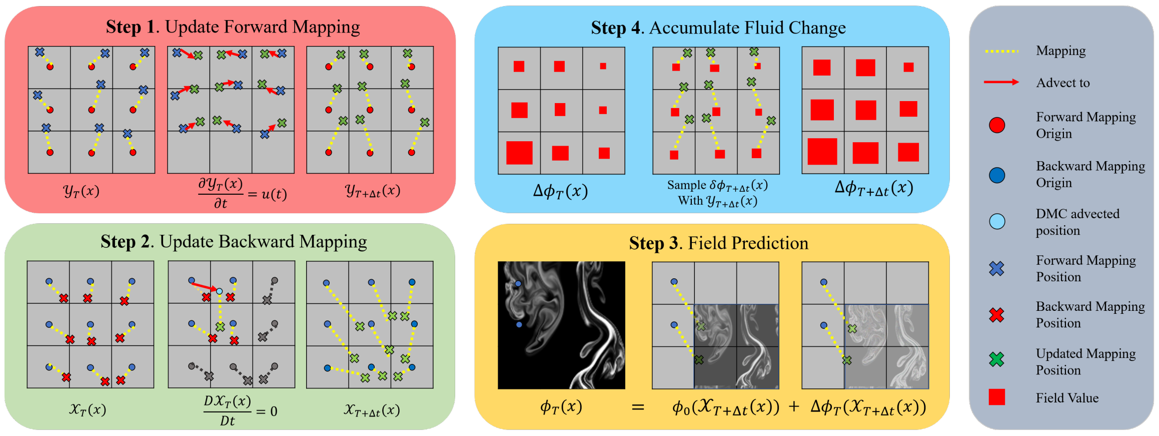
Step 1. Update Forward Mapping
$\mathcal{Y}_T(x)$
$\frac{\partial \mathcal{Y}_T(x)}{\partial t} = u(t)$
$\mathcal{Y}_{T+\Delta t}(x)$
Step 4. Accumulate Fluid Change
$\Delta\phi_T(x)$
Sample $\delta\phi_{T+\Delta t}(x)$ With $\mathcal{Y}_{T+\Delta t}(x)$
$\Delta\phi_{T+\Delta t}(x)$
Step 2. Update Backward Mapping
$\mathcal{X}_T(x)$
$\frac{D\mathcal{X}_T(x)}{Dt} = 0$
$\mathcal{X}_{T+\Delta t}(x)$
Step 3. Field Prediction
$\phi_T(x) = \phi_0(\mathcal{X}_{T+\Delta t}(x)) + \Delta\phi_T(\mathcal{X}_{T+\Delta t}(x))$
Mapping
Advect to
Forward Mapping Origin
Backward Mapping Origin
DMC advected position
Forward Mapping Position
Backward Mapping Position
Updated Mapping Position
Field Value

Fig. 2. Schematic illustration of a one level BiMocq². Step 1: update the forward mapping by solving a partial differential equation of the mapping $\mathcal{Y}$. Step 2: update the backward mapping by solving the advection equation of the mapping. Step 3: any fluid quantity ϕ is obtained by using the updated backward mapping $\mathcal{X}_{T+\Delta t}$ and interpolating the fluid origin buffer ϕ_0 and accumulation buffer $\Delta\phi_T$. Step 4: accumulate any fluid change through the updated forward mapping $\mathcal{Y}_{T+\Delta t}$ to accumulation buffer $\Delta\phi$.

- The approach is orthogonal to previous fluid solvers, hence can be integrated into existing pipelines with minor modifications.

In addition, to extensively evaluate our solver, comprehensive comparisons against state-of-the-art algorithms are conducted, including 2D and 3D vortical flow §4.1-4.2, level-set advection §4.1 and basic 3D smoke simulations. In addition, the solver is practically refined for complex 3D simulations with moving boundaries §4.2 and volumetric combustion §4.3. A schematic illustration of a one-level BiMocq² is given in Fig. 2.

2 RELATED WORK

By trading accuracy for stability, Semi-Lagrangian advection method [Stam 1999] has made fluid simulation highly practical for computer graphics and opened a wide range of applications for physically based fluid animation, such as smoke [Fedkiw et al. 2001], liquids [Foster and Fedkiw 2001], flames [Nguyen et al. 2002], and volumetric combustion [Feldman et al. 2003]. However, the Semi-Lagrangian method manifests severe numerical diffusion due to the repeated applications of field interpolation. A large volume of algorithms has been proposed to address this issue.

Higher order Semi-Lagrangian methods such as BFECC [Kim et al. 2005] and MacCormack [Selle et al. 2008] take Semi-Lagrangian as the building block, and restore the numerical accuracy of the flow by measuring the difference of the flow both backward and forward in time for the given time step; nevertheless, this kind of methods lacks the essential mechanism to avoid accumulated errors, thus the flow would eventually fade-out. Our approach takes advantage of the flow information from a long-term period to largely reduce the accumulation of numerical diffusion.

Particle-grid hybrid methods store all the fluid quantities on the particles and rely on particle-mesh routines to transfer physical values from particles to the background grid to resolve the momentum conservation, and from the grid to particles to advect the system status. While FLIP [Zhu and Bridson 2005] removes the numerical diffusivity by only interpolating the change of the flow from the grid, APIC [Jiang et al. 2015] and PolyPic [Fu et al. 2017] further fix the numerical noise of FLIP by constructing additional functions to preserve local flow features. Nevertheless, particle-mesh operations require non-trivial effort to achieve optimized parallelization for efficiency. As observed by [Ferstl et al. 2016; Gao et al. 2018], particle-mesh transfers are becoming the new bottleneck of high-performance Eulerian fluid solvers due to the potential write-conflicts when multiple particles attempt to write data into the same node. Our approach only requires buffer samplings, thus can be easily parallelized on shared memory computation architectures.

Vorticity modeling looks at the curl form of Navier-Stokes equations [Cottet et al. 2000], by using either vortex filaments [Weißmann and Pinkall 2010] or vortex sheets [Pfaff et al. 2012], or even with Eulerian representations [Elcott et al. 2007; Zhang et al. 2015]. They have achieved great promises capturing vortical flow motions. However, the need of a stream function solver [Ando et al. 2015], or the need of managing the geometry [Brochu et al. 2012], or even simply the need of artistic controls (with arbitrary boundary motion and external forces) [Angelidis 2017] challenges the practicality of vortex methods.

Energy preserving solvers, e.g., [Mullen et al. 2009], are able to preserve fluid energy discretely; however the requirement of applying Newton solvers to solve the non-linearly coupled equations makes it a costly choice for practical applications. A recently proposed solver [Zehnder et al. 2018] demonstrates strong energy preservation. The reflection time stepping essentially integrates the Navier-Stokes equation in a prediction-correction manner where a predictive pressure $\hat{p}$ can be obtained from the previous time step. And stepping the advection term with pressure correction improves the temporal approximation order. However, this approach is mainly designed for energy conservation of velocity dynamics. In fact, even with an analytic velocity field, a straightforward advector may still fail to preserve fluid details.

Fluid details can be enhanced in many different ways: by ejecting external forces derived from the local flow field [Fedkiw et al. 2001], or simply as a post-process for fluid upscaling, such as [Kim et al. 2008] and [Xie et al. 2018]. Our method is orthogonal to turbulence models hence has no difficulty to be combined with them.

Fluid mapping techniques have been introduced to graphics in previous works, for flow field visualization [Crawfis and Max 1993] or procedural flow images [Sims 1992]. [Max et al. 1992] advect cloud texture to visualize the wind field. [Stam and Fiume 1995] traces back and warp blob particles to model fire appearance with a diffusion model. In [Stam 1999], a back-warped mapping at every time-step is essentially used to update flow fields.

Our method is based on the **methods of characteristic mapping** [Wiggert and Wylie 1976]. Despite some attempts in computer graphics to use MCM for fluid simulation [Sato et al. 2017; Tessendorf and Pelfrey 2011], existing MCM solvers are still impractical to be used, due to the expensive back time integration and artifacts associated with the highly distorted mapping. [Tessendorf 2015] provides insightful analysis and derivations for the MCM scheme with backward mapping, and even demonstrates exact solutions in the case of simple velocity fields. On the other hand, our work introduces the forward mapping to practically advect flow fields in more general situations. We refer to [Tessendorf 2015] for more theoretical backgrounds of the scheme.

In this paper, we explore the full potential of MCM for practical high-quality fluid simulations. We first completely avoid the expensive integral of characteristics by advecting the mappings dynamically, which can further be augmented with the dual mesh characteristic (DMC) model to ensure the accuracy. Then we introduce a multi-level mappings technique to preserve long-term sharpness and temporal coherence of the flow field. The obtained solver is efficient, highly parallel-friendly and yet, easy to implement. Our method outperforms several state-of-the-art alternatives on the standard benchmarks, and it is also capable of generating visually appealing results with strong agreements with real phenomena.

3 BiMocq^2

In this section, we explain in detail the proposed algorithm, including backward mapping §3.2, forward mapping §3.3, error correction §3.7 and our time integration scheme §3.6 which takes two levels of mapping for efficient long-term prediction of the post-advection velocity field. We found $n = 2$ to be a good balance in preserving both the visual quality and the computational efficiency.

The outline of BiMocq^2 is given in Alg. 1. For the sake of conciseness, the discussion and explanation of our algorithm will be focused on the velocity field only, but the same derivations stay valid for any other fluid fields.

3.1 Data Structure

Dense grid storage scheme and Marker-And-Cell (MAC) discretization are adopted in this paper. In addition to the buffers as required by typical Eulerian fluid solvers, we use extra buffers to track the forward and backward mappings. Each of them stores an individual 3D vector for every single cell, with index (i, j, k), in the simulation domain. At time t_0, the 3D vector of each grid cell of both buffers

ALGORITHM 1: Simulation time step with BiMocq^2

```
Input: $\mathbf{u}^n$, $\mathcal{X}_{\text{curr}}$, $\mathcal{X}_{\text{prev}}$, $\mathcal{Y}_{\text{curr}}$, $\Delta\mathbf{u}_{\text{curr}}$, $\Delta\rho_{\text{curr}}$, $\Delta T_{\text{curr}}$, $\Delta\mathbf{u}_{\text{prev}}$, $\Delta\rho_{\text{prev}}$,
       $\Delta T_{\text{prev}}$, $\mathbf{u}_0$, $\mathbf{u}_1$, $\rho_0$, $\rho_1$, $T_0$, $T_1$, $\Delta t$ fluid state and auxiliary buffers
       available at time step $n$.
Output: $\mathbf{u}^{n+1}$, $\rho^{n+1}$, $T^{n+1}$, $\mathcal{X}_{\text{curr}}$,$\mathcal{Y}_{\text{curr}}$, $\Delta\mathbf{u}_{\text{curr}}$, $\Delta\rho_{\text{curr}}$, $\Delta T_{\text{curr}}$, updated
        fluid quantity and auxiliary buffers.
1:  $\mathcal{X}_{\text{curr}} \leftarrow$ Advect($\mathcal{X}_{\text{curr}}$, $\mathbf{u}^n$, $\Delta t$) {§3.2.1}
2:  $\mathcal{Y}_{\text{curr}} \leftarrow$ ForwardMap($\mathcal{Y}_{\text{curr}}$, $\mathbf{u}^n$, $\Delta t$){§3.3}
3:  // Denoting $[\mathbf{u},\rho,T]$ as $\phi$
4:  $\phi^* \leftarrow$ IntegrateMultiLevel($\phi_0$, $\Delta\phi_{\text{prev}}$, $\Delta\phi_{\text{curr}}$){§3.5}
    ErrorCorrect($\phi^*$) {§3.7}
5:  $\mathbf{u}^{**}$, $\rho^{n+1}$, $T^{n+1} \leftarrow$ AddFluidEmissionAndDiffusion()
6:  $\delta\mathbf{u}$, $\delta\rho$, $\delta T \leftarrow$ FluidChange()
7:  $\Delta\mathbf{u}_{\text{curr}}$, $\Delta\rho_{\text{curr}}$, $\Delta T_{\text{curr}} \leftarrow$
    ApplyChangeForwardMap($\delta\mathbf{u}$, $\delta\rho$, $\delta T$){§3.3}
8:  $\mathbf{u}^{n+1} \leftarrow \mathcal{P}(\mathbf{u}^{**})$
9:  $\delta\mathbf{u} \leftarrow \mathbf{u}^{n+1} - \mathbf{u}^{**}$
10: $c_p \leftarrow$ (ReinitializationCond == true)?1 : 2
11: $\Delta\mathbf{u}_{\text{curr}} \leftarrow$ ApplyChangeForwardMap($c_p\delta\mathbf{u}$){§3.3}
12: if (ReinitializationCond == true) then
13:     Reinitialize(){§3.4}
14:     $\Delta\mathbf{u}_{\text{curr}} \leftarrow$ ApplyChangeForwardMap($c_p\delta\mathbf{u}$)
15: end if
```

was initialized to the cell-centered coordinate of the grid cell,

$$\mathcal{X}(i, j, k) = \mathcal{Y}(i, j, k) = h \times (i + 0.5, j + 0.5, k + 0.5)$$

where h represents the cell size. Those 3D vectors will then be updated by the flow, and more details can be found in later sections.

Furthermore, we dynamically track the accumulated changes of fluid quantities using additional ***accumulation*** buffers, e.g., $\Delta\mathbf{u}$ for velocities, $\Delta\rho$ for density, and ΔT for temperature. Each grid cell of these buffers represents the corresponding accumulated physical quantity changes.

3.2 Backward mapping

Applying Semi-Lagrangian advection repeatedly is one of the major origination of numerical diffusion. Equivalently a low pass-filter is constantly applied to the flow field, causing both fluid momentum and mass to dissipate as the simulation proceeds as shown in Fig. 3.

On the other hand, MCM is designed to skip all intermediate interpolations by tracing back to a previous time instant and directly interpolating from that time instant to obtain a much sharper fluid state. The idea is to find a backward mapping $\mathcal{X}$, such that given a spatial position $\mathbf{x}(T)$, this mapping computes its original position $\mathbf{x}(t_0)$, satisfying Eqn. (5). This mapping can be generated on-the-fly, by tracing back many time steps with the previously computed and stored fluid velocity fields as in [Sato et al. 2017], which turns out to be both time-consuming and memory-inefficient. To avoid such difficulties, we instead dynamically advect the mapping as:

$$\frac{D\mathcal{X}}{Dt} = \frac{\partial\mathcal{X}}{\partial t} + \mathbf{u}(t) \cdot \nabla\mathcal{X} = 0 \tag{8}$$

Given a fluid parcel at $\mathbf{x}(T)$, the backward mapping of $\mathcal{X}(\mathbf{x}(T))$ returns its original position $\mathbf{x}(t_0)$ which actually remains fixed as the fluid parcel moves. This observation indicates that the mapping $\mathcal{X}$ essentially has a zero material derivative over time.

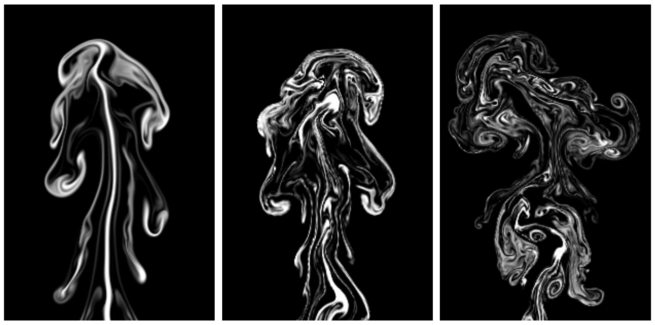

Fig. 3. 2D simulation of a released smoke in a tank, from left to right: results obtained with Semi-Lagrangian simulation, results from only advecting the density field with BiMocq^2, and results obtained by applying BiMocq^2 to both velocity and density fields. Note that simply improving the accuracy of density advection is able to bring more detailed flow field as the density field affects the flow simulation via buoyancy force. With BiMocq^2 used for both fields, the flow becomes more energetic due to better momentum conservation.

With Eqn. (8), the mapping of the new time step can be obtained by simply advecting the mapping of the previous time step. We can employ Semi-Lagragian advection to get:

$$\mathcal{X}^{n+1}(\mathbf{x}) = \mathcal{X}^{n}(\hat{\mathbf{x}}) \tag{9}$$

where $\hat{\mathbf{x}}$ denotes the departing position at the previous time step of a fluid parcel $\mathbf{x}$ and can be calculated as $\hat{\mathbf{x}} = \mathbf{x} - \Delta t \mathbf{u}^n(\mathbf{x})$ in the standard Semi-Lagrangian solvers.

3.2.1 Dual mesh characteristic. Although Semi-Lagrangian advection for the mapping diffuses the mapping itself, other fluid fields may still retain certain sharpness using this mapping for interpolation, as demonstrated in many previous works. Unfortunately, in highly rotational velocity fields, this simple strategy drastically loses its accuracy. To alleviate these issues, we apply dual-mesh characteristic from [Cho et al. 2018], which improves the numerical accuracy for the mapping advection.

Essentially, DMC solves a final value problem to find the starting point of fluid parcel whose trajectory ends at fixed gird nodes. With piece-wise linear velocity field, DMC derives an analytical solution which contains exponentials. The DMC routine proceeds as follows (explained in 1D): let x be the node position where we store our mapping, we first calculate the other node position $\hat{x}$ as

$$\hat{x} = \begin{cases} x - h, & \text{if } v(x) \leq 0, \\ x + h, & \text{otherwise.} \end{cases} \tag{10}$$

Supposing the velocity can be linearly blended between x and $\hat{x}$:

$$\begin{aligned} v(\xi) &= v(\hat{x}) + (\xi - \hat{x})a \\ a &= \frac{v(x) - v(\hat{x})}{x - \hat{x}} \end{aligned} \tag{11}$$

Note that $v(x)dt = d\xi$, we have

$$\Delta t = \int_{t^{n-1}}^{t^n} dt = \int_{\overline{x}}^{x} \frac{1}{v(x)} d\xi \tag{12}$$

Combining Eqn. (11) and Eqn. (12) we can calculate the tracing-back position as

$$\overline{x} = \begin{cases} x - v(x)\Delta t, & \text{if } a = 0, \\ x - \frac{1}{a}(1 - e^{-a\Delta t})v(x), & \text{otherwise.} \end{cases} \tag{13}$$

where $\overline{x}$ and $\overline{v}$ also satisfy

$$\begin{aligned} \overline{x} &= x - \Delta t \overline{v} \\ \overline{v} &= \frac{1}{a\Delta t}(1 - e^{-a\Delta t})v(x) \end{aligned} \tag{14}$$

$\overline{v}$ is the average velocity between x and $\overline{x}$. Since DMC requires CFL number to be less than 1, we perform multi-substeps with a freezed velocity field to advect the mapping, similar as advecting fluid buffers with many sub-steps for large Δt in modern fluid solvers.

As shown in the 2D Taylor vortex experiment in Fig. 4, DMC better preserves vorticity than Semi-Lagrangian method. It can also be observed that the separation positions of these two vortices are different. Comparing with the result from the reflection solver in Fig. 9, it becomes clear that DMC scheme produces a more accurate result.

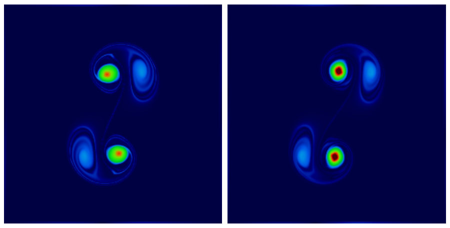

Fig. 4. Vorticity visualization of 2D Taylor Vortex test at $t = 7.5$, left: mapping advected with Semi-Lagrangian method, right: mapping advected with DMC method.

3.3 Forward mapping and accumulated fluid changes

Besides backward mapping, we introduce its counterpart, forward mapping, to track the accumulated fluid changes along the characteristics. After the backward mapping of the current coordinates has been constructed, the pure advection part of the flow can be computed by

$$\mathbf{u}(\mathbf{x}(t), t) = \mathbf{u}(\mathcal{X}(\mathbf{x}(t)), t_0). \tag{15}$$

However, when there exists external influences, such as viscosity and pressure, we should take into account the fluid accelerations.

To track the acceleration, Sato et al. [2017] evaluate an integral along the path back-in-time to accumulate fluid changes. To avoid this time-consuming integration, we introduce a forward mapping to expedite the computation.

Conceptually, if we trace massless infinitesimal monitors from the initial time which are passively advected by the flow, we would be able to observe and accumulate the flow acceleration along its trajectory.

ALGORITHM 2: Re-initialization with BiMocq^2

```
Input: $\mathcal{X}_{\text{prev}}$, $\mathcal{X}_{\text{curr}}$, $\mathcal{Y}_{\text{curr}}$, $\phi_{\text{prev}}$, $\phi_{\text{curr}}$, $\phi^n$, $\Delta\phi_{\text{prev}}$, $\Delta\phi_{\text{curr}}$
Output: $\mathcal{X}_{\text{prev}}$, $\mathcal{X}_{\text{curr}}$, $\mathcal{Y}_{\text{curr}}$, $\phi_{\text{prev}}$, $\phi_{\text{curr}}$, $\Delta\phi_{\text{prev}}$, $\Delta\phi_{\text{curr}}$
  1: $\mathcal{X}_{\text{prev}}(i,j,k) = \mathcal{X}_{\text{curr}}(i,j,k)$
  2: $\mathcal{X}_{\text{curr}}(i,j,k) = \mathcal{Y}_{\text{curr}}(i,j,k) = h \times (i+0.5, j+0.5, k+0.5)$
  3: $\phi_{\text{prev}}(i,j,k) = \phi_{\text{curr}}(i,j,k)$
  4: $\phi_{\text{curr}}(i,j,k) = \phi^n(i,j,k)$
  5: $\Delta\phi_{\text{prev}}(i,j,k) = \Delta\phi_{\text{curr}}(i,j,k)$
  6: $\Delta\phi_{\text{curr}}(i,j,k) = 0$
     $\phi$ denotes any fluid quantities, e.g. $u$, $\rho$, $T$ and etc.
```

Mathematically, the accumulated fluid change as a function of its original position can be defined as:

$$\Delta\mathbf{u}(\mathbf{x}(t_0), t) \coloneqq \int_{t_0}^{t} d\tau \frac{D\mathbf{u}}{Dt}(\mathbf{x}(\tau), \tau) \tag{16}$$

Implementation-wise, we use the accumulation buffer $\Delta\mathbf{u}$, to track the accumulated change: we follow the trajectory of a fluid parcel, access the fluid change at given time with the parcel's temporal position, and add this change back to its origination position. In our case, all infinitesimal fluid monitors were initiated at fixed grid centers, we need to know the positioning of those infinitesimal monitors at time t, hence we define a forward mapping $\mathcal{Y}_{t_0}^{t}(\mathbf{x}) : \mathbf{x}(t_0) \mapsto \mathbf{x}(t)$ to keep such information.

With forward mapping, the accumulated change can now be efficiently updated as:

$$\Delta\mathbf{u}(\mathbf{x}, t+\Delta t) = \Delta\mathbf{u}(\mathbf{x}, t) + \delta\mathbf{u}(\mathcal{Y}_{t_0}^{t+\Delta t}(\mathbf{x}), t+\Delta t) \tag{17}$$

where $\delta\mathbf{u}(\mathcal{Y}_{t_0}^{t+\Delta t}(\mathbf{x}), t+\Delta t)$ can be interpolated from the nodal momentum change obtained by numerically differencing the Navier-Stokes equations after self-advection, sourcing from viscosity, pressure gradients and external forces.

In fact, $\mathcal{Y}_{t_0}^{t}$ can be evolved by forward tracing the particles' positions; for example, given $\mathcal{Y}_{t_0}^{t}(\mathbf{x}) : \mathbf{x}(t_0) \mapsto \mathbf{x}(t)$, we can compute $\mathcal{Y}_{t_0}^{t+\Delta t}(\mathbf{x}) : \mathbf{x}(t_0) \mapsto \mathbf{x}(t+\Delta t)$, e.g. by one step Euler's method, as:

$$\mathcal{Y}_{t_0}^{t+\Delta t}(\mathbf{x}) = \mathcal{Y}_{t_0}^{t}(\mathbf{x}) + \Delta t\mathbf{u}(\mathcal{Y}_{t_0}^{t}(\mathbf{x}), t) \tag{18}$$

whose limit case becomes

$$\frac{\partial \mathcal{Y}_{t_0}^{t}}{\partial t}(\mathbf{x}) = \mathbf{u}(\mathcal{Y}_{t_0}^{t}(\mathbf{x}), t) \tag{19}$$

In practice, all our numerical tests adopt the third order accurate Runge-Kutta method.

In contrast to particle-mesh hybrid methods where data splatting is required, our approach remains parallel-friendly since interpolations are the only critical operations.

3.4 Re-initialization and its criterion

When the backward and forward mappings become highly distorted, the represented mapping functions are no longer valid; hence re-initialization is required to re-initialize the mapping functions. During the re-initialization, the mappings will be set to an initial state and the t_0 flow state will be set to start from the current time. The re-initialization algorithm is summarized in Alg. 2.

However, frequent re-initialization can gradually blur the flow. Thus we propose a re-initialization criterion to determine when the re-initialization operations should be applied.

Ideally, if all the computations are accurate, we argue that the backward mapping of a forward mapping of a position shall return the position itself. Hence, we measure the inaccuracy by looking at the differences of the back and forth mapping of any test position in the domain. Formally,

$$\begin{aligned} d_1 &= \|\mathcal{Y}(\mathcal{X}(\mathbf{x}(t))) - \mathbf{x}(t)\|_\infty \\ d_2 &= \|\mathcal{X}(\mathcal{Y}(\mathbf{x}(t_0))) - \mathbf{x}(t_0)\|_\infty \\ d_{\max} &= \max(d_1, d_2) \end{aligned} \tag{20}$$

To ensure the accuracy and robustness, this criterion should be dimensionless and not be affected by resolution. Empirically we design a dimensionless quantity q as

$$q = \frac{d_{\max}}{\Delta t v_{\max}} \tag{21}$$

where $v_{\max}$ is the the maximum velocity component and Δt is the time step. When q is larger than some threshold $\hat{q}$, it is preferred to re-initialize the mapping to maintain the accuracy. We find $\hat{q} \in [1.0, 1.5]$ to be a relatively well-balanced choice for velocity fields, other fields may use more relaxed criteria.

When re-initialization with double characteristic mappings, we first set the previous mapping and corresponding mapped fluid states to the recent mapping information, keeping the previous backward mapping, and at last re-initialize the current backward and forward mappings to grid-cells' positions.

3.5 Multi-level mapping

While the re-initialization of the mappings could possibly address the issue of distortions, it would introduce noticeable blurs at frames right after the re-initialization operations. To better preserve the visual sharpness and temporal coherence of the simulation, two levels of mapping can be employed for tracking fluid quantities. Let t_{I-1} be the time when the previous re-initialization of the mapping happens, with

$$\mathcal{X}_0(\mathbf{x}(t_I)) : \mathbf{x}(t_I) \mapsto \mathbf{x}(t_{I-1})$$

and let t_I be the time when the most recent re-initialization of the mapping happens, with

$$\mathcal{X}_1(\mathbf{x}(t)) : \mathbf{x}(t) \mapsto \mathbf{x}(t_I)$$

where t is the current time step to be solved for. The mapping $\mathcal{X}(\mathbf{x}(t)) : \mathbf{x}(t) \mapsto \mathbf{x}(t_{I-1})$ can be readily retrieved via

$$\mathcal{X}_t(\mathbf{x}(t)) : \mathbf{x}(t) \mapsto \mathbf{x}(t_{I-1}) = \mathcal{X}_0(\mathcal{X}_1(\mathbf{x}(t))) \tag{22}$$

Consequently, the integral of fluid changes can be obtained by interpolating twice from the two levels of forward mappings. For the velocity components at a grid cell $\mathbf{x}_g$, we have:

$$\begin{aligned} \mathbf{x} &= \mathcal{X}_t(\mathbf{x}_g) \\ \mathbf{u}(\mathbf{x}_g, t) &= \mathcal{I}(\mathbf{u}_{I-1}, \mathbf{x}) + \mathcal{I}(\Delta\mathbf{u}_{I-1}, \mathbf{x}) \\ &\quad + \mathcal{I}(\Delta\mathbf{u}_I, \mathcal{X}_1(\mathbf{x}_g)) \end{aligned} \tag{23}$$

Where $\mathcal{I}$ represents a sampling operator. Other fluid fields can be obtained in a similar way, as outlined in Alg. 1.

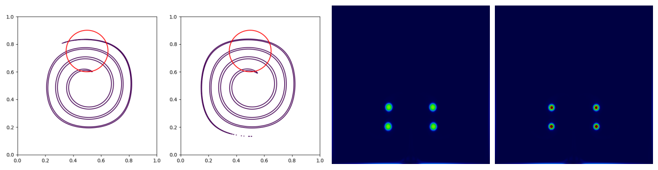

Fig. 5. Two simulation cases with single layer mapping and double layer mapping. Left side: for level-set advection, double characteristic mapping(middle left) improves the volume conservation. Right side: when looking at the vorticity field of a vortical flow simulation, double characteristic mapping(right most) preserves circulation much better.

In Fig. 5, we demonstrate that simulation quality can be significantly improved by simply doing one more level of mapping.

3.6 Time integration

In this section, we extend a bit more with regard to some important algorithmic choices as made in Alg. 1.

Pressure correction at the re-initialization instant. As per [Zehnder et al. 2018], energy preserving can be achieved by adding twice the amount of the projective pressure correction to the post-advected velocity field. In our approach, this can be easily achieved by cumulating twice the amount of pressure gradient to $\delta\mathbf{u}$.

When the re-initialization is conducted (usually at the end of one time step), some portions of the pressure gradient should be added to $\Delta\mathbf{u}_{I-1}$ (i.e. the previous cumulation buffer); while the other portions of the pressure gradient should be added to $\Delta\mathbf{u}_I$ (i.e., the new cumulation buffer). Whereas if we simply accumulate both portions to $\Delta\mathbf{u}_{I-1}$, we could end up losing the energy at the next re-initialization point when $\Delta\mathbf{u}_{I-1}$ gets erased. Artifacts due to this temporal-misalignment of pressure corrections can be found in our supplemental video.

Temporal blend. With our two levels of mappings, the post-advection velocity field can be obtained from two integral trajectories: one from the previous mapping, and the other one from the current mapping as per Eqn. (23). We blend the two results with a blending weight by inserting one more term into Eqn. (23):

$$\begin{aligned}\mathbf{x} &= \mathcal{X}_t(\mathbf{x}_g)\\ \mathbf{u}(\mathbf{x}_g,t) &= \frac{1}{2}\mathcal{I}(\mathbf{u}_{I-1},\mathbf{x}) + \frac{1}{2}\mathcal{I}(\mathbf{u}_I,\mathcal{X}_1(\mathbf{x}_g))\\ &\quad + \frac{1}{2}\mathcal{I}(\Delta\mathbf{u}_{I-1},\mathbf{x}) + \mathcal{I}(\Delta\mathbf{u}_I,\mathcal{X}_1(\mathbf{x}_g))\end{aligned} \tag{24}$$

Again, the same computation can be applied to other fluid buffers.

3.7 Error correction with BiMocq^2

Till now, our pipeline would already be able to outperform most alternative methods in the standard benchmarks as shown in Section 4. However, there are still two major sources of possible numerical diffusion which may hinder us from getting even more visual details in practical simulations:

- Any numerical dissipation generated up to the re-initialization time would be inherited by the simulations afterward.
- The sampling from a previously mapped fluid state to obtain the current flow prediction could also suffer from diffusion.

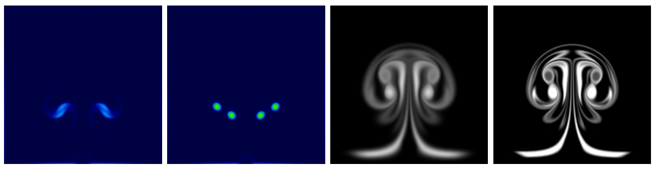

Fig. 6. Two vortex pairs leapfrogging. Left: vorticity field result without any error correction. Middle left: vorticity field obtained with the gapped EC. Middle right: with the same velocity field, density field obtained without EC. Right: with the same velocity field, density field obtained with the gapped EC.

We propose an error correction method to tackle these two types of numerical diffusion to achieve significant improvements in our experiments, e.g. Fig. 6.

Extending the derivations in [Kim et al. 2005], we estimate the numerical diffusion e between the long-term backward and forward mappings of a filed ϕ as:

$$\begin{aligned}\phi^*(\mathbf{x}_g) &= \mathcal{I}(\phi_0,\mathcal{X}(\mathbf{x}_g))\\ e(\mathbf{x}_g) &= \frac{1}{2}(\mathcal{I}(\phi^*,\mathcal{Y}(\mathbf{x}_g)) - \phi_0(\mathbf{x}_g))\end{aligned} \tag{25}$$

where $\mathcal{I}$ represents a sampling method and ϕ_0 represents the initial state at the previously re-initialized mapping instant.

After we get e, we map it to the future time by sampling it with the backward mapping, and consequently correct the flow field (velocity, density, temperature, etc.) via

$$\phi^*(\mathbf{x}_g) = \phi^*(\mathbf{x}_g) - \mathcal{I}(e,\mathcal{X}(\mathbf{x}_g)) \tag{26}$$

Here we identified this BFECC type of error correction ([Kim et al. 2005]) works extremely well with our long-term mappings, while the MacCormack type of error correction ([Selle et al. 2008]) loses its effectiveness. This discrepancy also implies that the error e is a quantity from the past moment.

When dealing with our long-term corrections, we must take into account the accumulated changes. In our implementation, for every time step we first remove the sampling errors from the fluid changes, such as $\delta\mathbf{u}$, $\delta\rho$ and δT, before adding them to the accumulation buffer. Then we compute the long-term error as:

$$\begin{aligned}\phi^*(\mathbf{x}_g) &= \mathcal{I}(\phi_0,\mathcal{X}(\mathbf{x}_g)) + \mathcal{I}(\Delta\phi,\mathcal{X}(\mathbf{x}_g))\\ e(\mathbf{x}_g) &= \frac{1}{2}(\mathcal{I}(\phi^*,\mathcal{Y}(\mathbf{x}_g)) - \Delta\phi(\mathbf{x}_g) - \phi_0(\mathbf{x}_g))\end{aligned} \tag{27}$$

where $\mathcal{I}$ indicates a sampling operation and $\Delta\phi$ represents the accumulated changes from the previous re-initialization instant.

3.7.1 *Discussion of error correction.*

The error correction (EC) can be applied either for every time step (the regular EC), or be applied to the "t_0 buffers" only when the re-initialization is performed (the gapped EC).

In Fig. 6, we compare the solver accuracy with and without error correction, where in particular the gapped EC is employed. It is obvious that the gapped EC not only improves the accuracy of the velocity field prediction, resulting in qualitatively different velocity fields, but also increases the sharpness of density field advection over a long-term period, enhancing the animation richness.

In Fig. 7, we further examine the effects of both the regular EC and the gapped EC. We observe that in the vortex leapfrogging example, error correction effectively improves the dynamics of the flow, leads to multiple rounds of vortex pair leapfrogs. In the Taylor vortex example, EC has less effect in improving the visual quality of the flow; our hypothesis is that the effectiveness of EC is highly related to the resolution of the vortical features. In both cases, no significant discrepancy is observed between the standard EC and the gapped EC; thus we recommend using gapped EC for efficiency considerations. The energy plots of the Taylor vortex simulation with different EC options are also provided in Fig. 8.

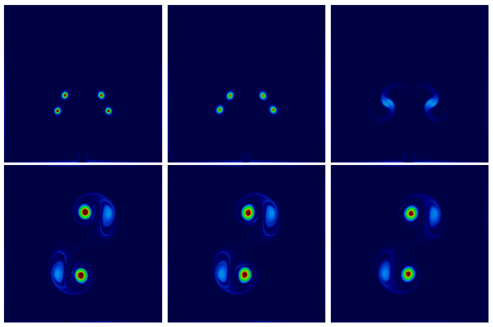

Fig. 7. Flow field visualization of two simulation cases. Left column: flow field obtained with the regular EC, middle column: flow field obtained with the gapped EC, right column: flow field obtained without EC.

3.7.2 *Clamp Extrema.* According to [Selle et al. 2008], such error correction may introduce new local extrema into the flow, degrading the stability. Hence, before writing back the corrected value, we clamp it by the maximum and minimum value from the neighboring cells, which are the post-advection values of the flow field.

4 RESULTS

We evaluate our BiMocq^2 algorithm with standard 2D benchmarks and test its practicality and performance with 3D simulations. The experiments reveal that BiMocq^2 has the potential to deliver much more accurate predictions for the fluid momentum with extremely sharp scalar field advection, beyond the capability of existing alternatives. We summarize the configurations for the benchmarks in Table 1. All the corresponding visual comparisons can be found in the supplemental video.

4.1 2D Results

Taylor Vortex. We follow the settings in [McKenzie 2007]: two vortices are placed close to each other, and they can either separate or merge as the simulation proceeds depending on their initial distance. We set this distance to 0.81, which is slightly larger than the critical separation distance. As shown in Fig. 9, most previous methods fail to separate the two vortices; while our solver produces very similar results as the reflection-advection solver from [Zehnder et al. 2018].

Level Set Advection. In the first test, we adopt the classical Zalesak's disk setting described in [Selle et al. 2008]. As shown in

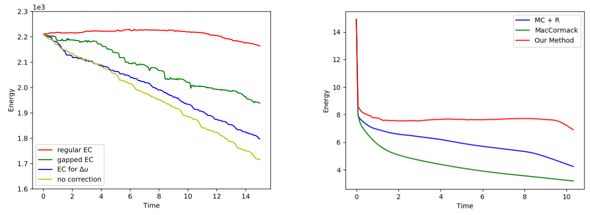

Fig. 8. Energy plots of 2D and 3D simulations. Left: energy curves produced with different error correction schemes, although the regular EC (red) gives best energy preserving result, the gapped EC (green) is more computationally efficient and produces almost visually identical result to that of regular EC. Simply correcting the accumulation buffer (blue) improves little compared to the one without any error correction (yellow). Right: energy curves produced by a 3D simulation of a injected smoke, our method (red) demonstrated superior preservation of energy.

Table 1. Simulation Configuration. (1) Smoke Emitter, Fig. 3, (2) Taylor Vortex, Fig. 9, (3) Zalesak's Disk, Fig. 10, (4) Vortex Leapforgging, Fig. 12, (5) Vortex in Box, Fig. 11, (6) Rayleigh-Taylor instability, Fig. 13, (7) Simple Smoke, Fig. 14, (8) Smoke with Moving Boundary, Fig. 16, (9) Vortex Collide, Fig. 1, (10) Combustion, Fig. 18, (11) Skidding car, Fig. 15

	Domain	Resolution	Δt	CFL[a]
1	1×2	256×1024	0.03	∞
2	$2\pi \times 2\pi$	256×256	0.025	∞
3	1×1	200×200	–	0.75
4	$2\pi \times 2\pi$	256×256	0.025	∞
5	1×1	512×512	–	0.5
6	0.2×1	256×1280	0.01	∞
7	$1 \times 2 \times 1$	$200 \times 400 \times 200$	0.02	∞
8	$2 \times 4 \times 2$	$256 \times 512 \times 256$	0.04	∞
9	$0.2 \times 0.4 \times 0.4$	$200 \times 400 \times 400$	0.08	∞
10	$2 \times 2 \times 2$	$256 \times 256 \times 256$	0.02	∞
11	$25 \times 10 \times 5$	$800 \times 320 \times 160$	0.01	∞

[a]Time-step restriction, with ∞ indicating no restriction on the time-step.

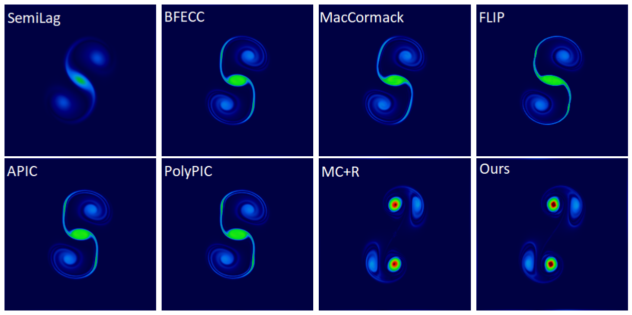

Fig. 9. 2D Taylor Vortices results at $t = 7.5$ seconds. Color indicates the vorticity magnitude.

Fig. 10, Semi-Lagrangian scheme suffers from serious dissipation, while BFECC generates a visually better result but nevertheless in a distorted way. On the contrary, our scheme introduces almost zero dissipation and is capable to maintain the disk shape even after three revolutions of rotations.

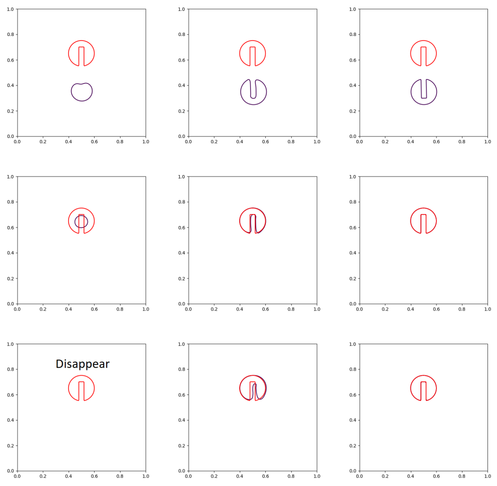

Fig. 10. 2D Zalesak's disk. Results shown are Semi-Lagrangian (left), BFECC (middle) and BiMocq^2 (right) at $t = 79$ seconds (top), $t = 628$ seconds (middle) and $t = 1884$ seconds (bottom). Note that our method can almost perfectly match the original shape after three revolutions.

The second level set test follows [Aanjaneya et al. 2017], where a level set circle deforms under the vortex velocity field in a box. As we can notice in Fig. 11, the BFECC method renders severe volume loss, while the particle level set method [Enright et al. 2002a] conserves volume better at the cost of repeatedly re-sampling 16 particles per cell near the interface. Our method manages to produce further improved result over the particle level set method without paying extra computations for particle re-samplings.

Vortex Leapfrogging. In this example (Fig. 12), two clock-wise rotating vortex pairs with identical strength are released from the bottom of a tank. The outer vortex pair is initially separated with twice the distance compared to the inner pair. Our method maintains the vortices over a long term period and meanwhile keeps the sharpness of the density field.

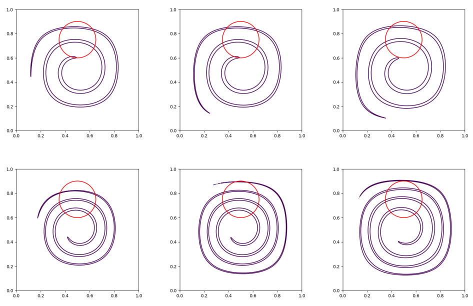

Fig. 11. 2D Vortex in a Box. Comparisons between BFECC (left), particle level set (middle) and our method (right) at $t = 3.2$ seconds (top) and $t = 5$ seconds (bottom). BiMocq^2 can preserve volume over a long time period with small computational cost.

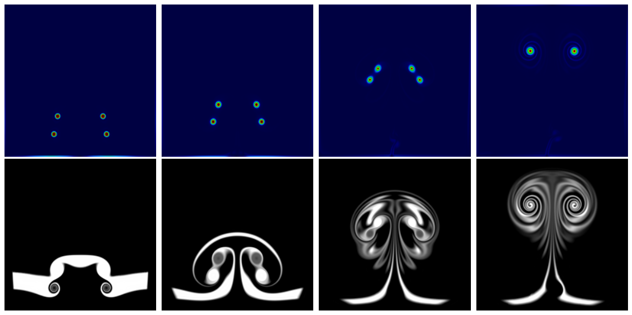

Fig. 12. 2D Vortex Leapfrogging. Top and bottom rows are visualized vorticity and density fields at $t = 6$, $t = 18.5$, $t = 50$ and $t = 70$ seconds.

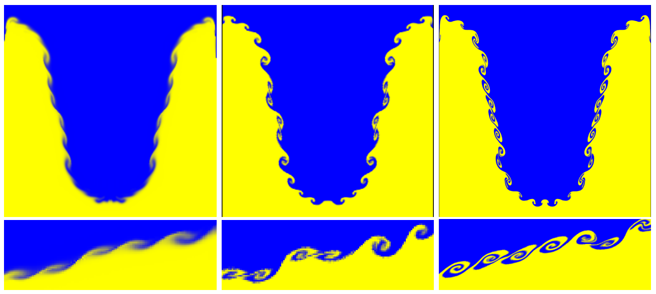

Fig. 13. 2D simulation of Rayleigh-Taylor instability, frame 118. Top left: result from MacCormack Reflection solver with BFECC density advector at a simulation resolution of 256x1280. Top middle: result obtained with BiMocq^2 at a simulation resolution of 256x1280. Top right: reference simulation at a simulation resolution of 768x3840. Bottom row: zoom-in view of the vortex rolling-up structure of the Kelvin-Helmholtz instability from each method.
'

Rayleigh-Taylor Instability . In this example (Fig. 13), the Kevin-Helmholtz-Rayleigh-Taylor instability was studied using a Boussinesq model. Blue (heavier) fluid was initially placed on top of a lighter (yellow) fluid. The density difference produces buoyancy forces that deforms the fluid interface. During this process, the density gradient along the interface of the two flow may introduce strong vorticle accelerations to the flow field, causing the interface (a vortex-sheet) to roll-up. This phenomenon is known as Kelvin-Helmholtz instability (Fig. 13, bottom row). The rolling-up of vortices is highly coupled with the sharpness of the density-interface. Fig. 13 shows that our method outperforms [Zehnder et al. 2018] with more accurate prediction of the vortex rolling-up structures, as can be verified by comparison to the reference simulation at 3× resolution.

4.2 3D simulations and the performance

For 3D simulations, while all the other portions of our pipeline remain in CPU, we specifically ported the kernel in response to the

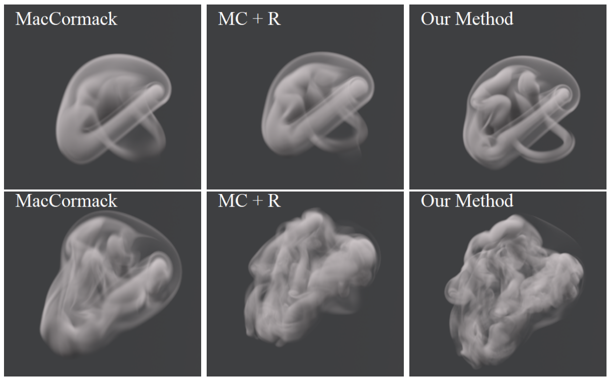

Fig. 14. Smoke ball rising up. Left: simulated with MacCormack advector. Middle: [Zehnder et al. 2018] Reflection solver for velocity field and MacCormack for density field. Right: our method. The ability to preserve thin structure of density field not only results in better visual richness, but also leads to more turbulence.

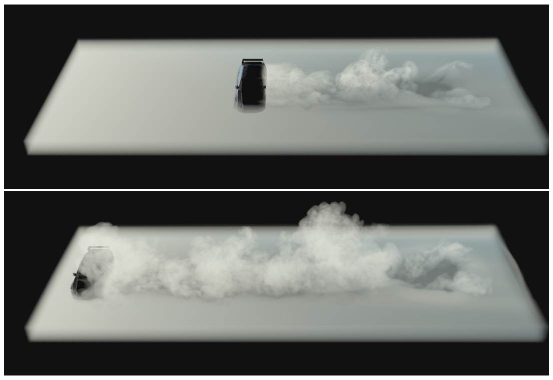

Fig. 15. Frames of a turbulent flow filed introduced by a skidding car. Top: frame 100, bottom: frame 275.

passive advections from CPU to GPU. For integrating the advection trajectory, such design choice, to our experience, usually provides superior efficiency. In practice, modern fluid solvers prefer to take sub-steps for passive advection of quantity; as a result, the advection kernel can easily become the bottleneck of advanced fluid solvers, especially when the pressure projection phase is equipped with an efficient multigrid preconditioner.

In a simulation case with a 300×600×600 domain and $\Delta t > 30$ CFL, our CPU parallel implementation of the pressure projection takes about 40 seconds to converge to the machine precision (and optionally takes another 100 seconds to build the multi-level matrices with moving boundaries); Our CPU parallel passive advection kernel requires more than 15 minutes to finish all the high-order ODE integrations with multiple sub-steps, which is intolerable for a large scale fluid solver. Thus, we ported the advection portion to GPU in a straightforward way to reduce the cost to ~20 seconds, leading to roughly a 45× speedup for the advection and an overall 6× speedup for the whole pipeline.

All the 3D performance timings are collected from a desktop machine with an Intel Core i7-6950X @ 3.00GHz CPU, an 128GM RAM and a nVidia Titan Xp graphics card. All the advectors (MacCormack, Semi-Lagragian, BFECC, DMC) are implemented on GPU while the pressure solver remains on CPU. All our 3D simulations were performed without any vorticity confinement or post-turbulence models.

Simple Smoke. In this test Fig. 14, two smoke spheres with different heat are released from the bottom of the container. Each frame takes ~22 seconds with MacCormack advector and ~28 seconds with our solver; while with MacCormack Reflection solver it takes ~40 seconds. BiMocq^2 drastically increases the sharpness of the smoke, resulting not only in a sharper appearance with passive thin features but also a more turbulent field with better preserved thin smoke structures.

Smoke with Moving Boundaries. In Fig. 15, a car skidding through a density field is simulated, each frame takes ~70 seconds. In Fig. 16, a rising smoke plume colliding with a sphere is simulated. The sphere remains stationary for the first 50 frames and then begins to move up and down. Each frame of the simulation costs ~60 seconds.

With relative motions at obstacle boundaries, mappings quickly become highly distorted near the collision boundary. Notice that flow lines from different origins can be stopped by the obstacle surface and hence the mapping information they carry, resulting in unresolvable mappings near the boundary; while at the regions far away from the collision surface, the mappings remain valid for advection. Hence, for a band of cells near the boundary, we revert the post-advection velocity field to BFECC advected results from last time step, and then compute the change between the reverted values and the values predicted by the long-term mappings. This difference should be considered as a momentum change and then be added to the accumulation buffer accordingly. Since the flow features in the vicinity of the boundary are essentially dominated by the boundary layer dynamics in lieu of the convection, we claim our boundary treatment is highly viable and in general accurate.

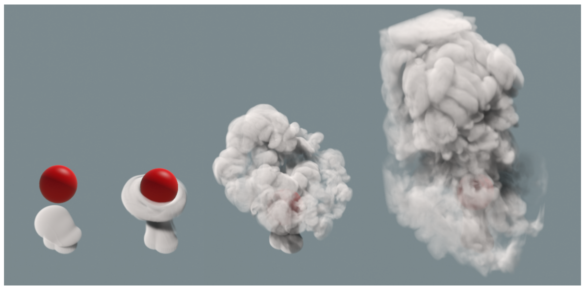

Fig. 16. Simulation of a rising smoke colliding with a moving sphere, from left to right: frame 32, frame 57, frame 116 and frame 232.

Vortex collide. In this example (Fig. 1), ink with different colors (colors don't affect the flow) are injected into the each other at the speed of 0.05 m/sec to form two vortex rings. The sources are 0.12m apart and the container, with resolution $200 \times 400 \times 400$ and cell size 0.001m, is full of fluid with a kinematic viscosity of 1×10^{-6} m^2/s,

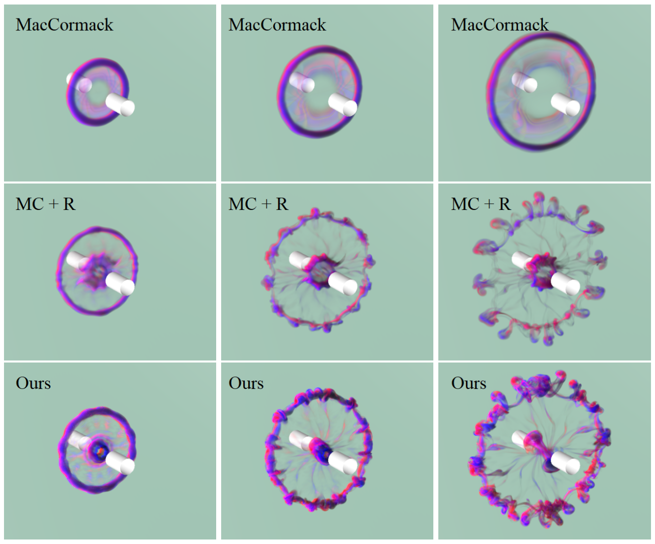

Fig. 17. Vortex rings collide. Top row: frames from MacCormack simulation; both the dye and momentum quickly diffuse out, thus the result diverges from reality. Middle row: [Zehnder et al. 2018] Reflection solver for velocity field and MacCormack for density field; although the momentum is well preserved, the density field still diffuses out due to the numerical dissipation. Bottom row: frames from our solver; with the great power to perform conservative convection, both the dye and velocity energy are well preserved, resulting in good agreement with real observation.

giving a Reynolds number of 2000. Our method shows great agreement with the vortex re-connection phenomena as documented in [Lim and Nickels 1992]. Each frame takes ~50 seconds. In contrast, an equivalent MacCormack simulation fails to reproduce such phenomena, while MacCormack Reflection solver is able to preserve the momentum but still loses sharpness in density field. A comparison can be found in Fig. 17.

4.3 Volumetric combustion

We further apply our BiMocq^2 solver to simulate volumetric combustion. We follow the divergence control strategy [Bridson 2008; Feldman et al. 2003]; while in addition to those solvers, we approximate the full flux of momentum, readily

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \mathbf{u} \nabla \cdot \mathbf{u} = -\frac{1}{\rho} \nabla p + \mathbf{f}$$
$$\nabla \cdot \mathbf{u} = g \tag{28}$$

where g is the target divergence function obtained from combustion reaction calculations.

While the BiMocq^2 solver integrates the advection-acceleration equation at great accuracy, we solve an additional differential equation for the intermediate velocity $\mathbf{u}^*$ to account for the full flux, formally,

$$\dot{\mathbf{u}} = -\mathbf{u} \nabla \cdot \mathbf{u} \tag{29}$$

This equation is solved on a grid-based manner and the change of velocity is accumulated to the accumulation buffer following §3.3. Simulation of 3D explosion can be found in Fig. 18.

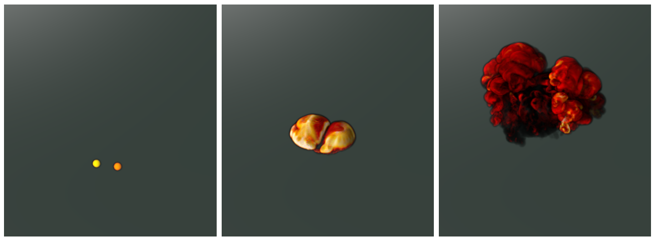

Fig. 18. Our method simulating explosions modeled with divergence control. The explosion was initiated with two burning sources at the bottom.

5 LIMITATION AND FUTURE WORK

Limitations. Although the proposed method is not restricted by CFL condition, the simulation accuracy highly relies on the consistency of the forward and backward mappings. Results become meaningless when the two mappings mismatch with each other to a certain extent. Furthermore, in the vicinity of the solid boundary, the proposed solution works sufficiently well to account for the boundary contact. However in the cases with free-boundaries being the visual focus of the simulation, extrapolation schemes shall be developed to warp the mapping functions around the area.

Future work. Our method inspires and enables us to probe into a few interesting possibilities as the future work.

Liquid. We are curious to see the method applied to free-surface flow simulations. We have already examined the effectiveness of our method for level set advections. We would like to explore the possibility of extending our idea to also improve the traditional fast marching method as well as the velocity extrapolations. We think the method also has great potential for numerically circulation preserving vortex simulations. Imagine the length of a vortex segment can be determined with the forward mapping and thus the temporal vorticity, which can be expressed as a production of a $t = 0$ quantity with the length, such approach could avoid the numerically unstable vortex stretching term in 3D simulations.

Eulerian solid and its coupling with fluid. As Eulerian grid-based solid simulation [Fan et al. 2013; Levin et al. 2011; Teng et al. 2016] becomes popular in graphics community due to its ability to enable easier coupling between fluid and solid and its robustness even with large time steps. We are interested in investigations of applying our method to such problems.

Sparse data structure. Our current pipeline is built upon a traditional dense data structure, which prohibits extremely high-resolution applications as in [Aanjaneya et al. 2017; Setaluri et al. 2014]. Thus, we look forward to exploring possibilities to exploit modern highly efficient sparse data structure for storing Cartesian grids such as OpenVDB [Museth 2013] and SPGrid [Gao 2018; Setaluri et al. 2014] to further accelerate the computation.

Adaptivity. Adaptive methods are usually employed to dedicate limited computational powers to regions of interests [Aanjaneya et al. 2017; Gao et al. 2017; Setaluri et al. 2014], such as the collision interfaces between smoke/liquid and rigid bodies. It would be challenging but meaningful to also apply this strategy to our method: when the flow front (of a plume, or wave) moves, or when vortical motion stretches the flow structure, mappings become highly entangled and shall be resolved with increased resolution.

Other directions. Our mappings can also be combined to study the data science of flow. For example, with the mappings available, we can study the possibility to directly learn from the flow trajectory. Furthermore, our method is a pure Eulerian method with both high accuracy and low dissipation, its performance can possibly be greatly boosted with GPU acceleration. We look forward to seeing such an accurate real-time simulation environment for other graphics researches such as reinforced learning for swimmers.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments. This work was supported in part by 973 Program (2015CB352501), NSF IIS-1755544 and NSF CCF-1813624. We thank Dr. Robert Bridson for insightful discussions. We also thank Haowei Han from Light Chaser Animation studio for his help in rendering.

REFERENCES

Mridul Aanjaneya, Ming Gao, Haixiang Liu, Christopher Batty, and Eftychios Sifakis. 2017. Power diagrams and sparse paged grids for high resolution adaptive liquids. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 140.

Ryoichi Ando, Nils Thuerey, and Chris Wojtan. 2015. A stream function solver for liquid simulations. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 53.

Alexis Angelidis. 2017. Multi-scale Vorticle Fluids. *ACM Trans. Graph.* 36, 4, Article 104 (July 2017), 12 pages. https://doi.org/10.1145/3072959.3073606

R. Bridson. 2008. *Fluid Simulation for Computer Graphics.* Taylor & Francis.

Tyson Brochu, Todd Keeler, and Robert Bridson. 2012. Linear-time smoke animation with vortex sheet meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* Eurographics Association, 87–95.

Chung-Ki Cho, Byungjoon Lee, and Seongjai Kim. 2018. Dual-Mesh Characteristics for Particle-Mesh Methods for the Simulation of Convection-Dominated Flows. *SIAM Journal on Scientific Computing* 40, 3 (2018), A1763–A1783.

Georges-Henri Cottet, Petros D Koumoutsakos, D Petros, et al. 2000. *Vortex methods: theory and practice.* Cambridge university press.

Roger A Crawfis and Nelson Max. 1993. Texture splats for 3D scalar and vector field visualization. In *Proceedings of the 4th conference on Visualization'93.* IEEE Computer Society, 261–266.

S. Elcott, Y. Tong, E. Kanso, P. Schröder, and M. Desbrun. 2007. Stable, Circulation-preserving, Simplicial Fluids. *ACM Trans. Graph.* 26, 1, Article 4 (2007).

Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. 2002a. A hybrid particle level set method for improved interface capturing. *Journal of Computational physics* 183, 1 (2002), 83–116.

Douglas Enright, Stephen Marschner, and Ronald Fedkiw. 2002b. Animation and rendering of complex water surfaces. In *ACM Trans. Graph.*, Vol. 21. 736–744.

Ye Fan, Joshua Litven, David IW Levin, and Dinesh K Pai. 2013. Eulerian-on-lagrangian simulation. *ACM Transactions on Graphics (TOG)* 32, 3 (2013), 22.

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual Simulation of Smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01).* ACM, New York, NY, USA, 15–22.

Bryan E Feldman, James F O'brien, and Okan Arikan. 2003. Animating suspended particle explosions. In *ACM Transactions on Graphics (TOG)*, Vol. 22. ACM, 708–715.

Florian Ferstl, Ryoichi Ando, Chris Wojtan, Rüdiger Westermann, and Nils Thuerey. 2016. Narrow band FLIP for liquid simulations. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 225–232.

Nick Foster and Ronald Fedkiw. 2001. Practical Animation of Liquids. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01).* ACM, New York, NY, USA, 23–30.

C. Fu, Q. Guo, T. Gast, C. Jiang, and J. Teran. 2017. A Polynomial Particle-in-cell Method. *ACM Trans. Graph.* 36, 6, Article 222 (2017), 222:1–222:12 pages.

M. Gao. 2018. *Sparse Paged Grid and its Applications to Adaptivity and Material Point Method in Physics Based Simulations.* Ph.D. Dissertation. University of Wisconsin, Madison.

Ming Gao, Andre Pradhana Tampubolon, Chenfanfu Jiang, and Eftychios Sifakis. 2017. An Adaptive Generalized Interpolation Material Point Method for Simulating Elastoplastic Materials. *ACM Trans. Graph.* 36, 6, Article 223 (Nov. 2017), 12 pages. https://doi.org/10.1145/3130800.3130879

Ming Gao, Xinlei Wang, Kui Wu, Andre Pradhana, Eftychios Sifakis, Cem Yuksel, and Chenfanfu Jiang. 2018. GPU Optimization of Material Point Methods. In *SIGGRAPH Asia 2018 Technical Papers (SIGGRAPH Asia '18).* ACM, New York, NY, USA, Article 254, 12 pages. https://doi.org/10.1145/3272127.3275044

C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. 2015. The Affine Particle-in-cell Method. *ACM Trans. Graph.* 34, 4 (July 2015), 51:1–51:10.

B. Kim, Y. Liu, I. Llamas, and J. Rossignac. 2005. FlowFixer: Using BFECC for Fluid Simulation. In *Eurographics Conference on Natural Phenomena.* 51–56.

Theodore Kim, Nils Thürey, Doug James, and Markus Gross. 2008. Wavelet Turbulence for Fluid Simulation. In *ACM SIGGRAPH 2008 Papers (SIGGRAPH '08).* ACM, New York, NY, USA, Article 50, 6 pages. https://doi.org/10.1145/1399504.1360649

David I. W. Levin, Joshua Litven, Garrett L. Jones, Shinjiro Sueda, and Dinesh K. Pai. 2011. Eulerian Solid Simulation with Contact. In *ACM SIGGRAPH 2011 Papers (SIGGRAPH '11).* Article 36, 36:1–36:10 pages.

T. T. Lim and T. B. Nickels. 1992. Instability and reconnection in the head-on collision of two vortex rings. *Nature* 357 (05 1992), 225–227. https://doi.org/10.1038/357225a0

Nelson Max, Roger Crawfis, and Dean Williams. 1992. Visualizing wind velocities by advecting cloud textures. In *Proceedings Visualization'92.* IEEE, 179–184.

A. McKenzie. 2007. *HOLA: a High-Order Lie Advection of discrete differential forms, with applications in fluid dynamics.* Ph.D. Dissertation. Caltech.

Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. 2009. Energy-preserving Integrators for Fluid Animation. In *ACM SIGGRAPH 2009 Papers (SIGGRAPH '09).* Article 38, 38:1–38:8 pages.

Ken Museth. 2013. VDB: High-resolution Sparse Volumes with Dynamic Topology. *ACM Trans. Graph.* 32, 3, Article 27 (July 2013), 22 pages. https://doi.org/10.1145/2487228.2487235

Duc Quang Nguyen, Ronald Fedkiw, and Henrik Wann Jensen. 2002. Physically Based Modeling and Animation of Fire. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques.* 721–728.

Tobias Pfaff, Nils Thuerey, and Markus Gross. 2012. Lagrangian Vortex Sheets for Animating Fluids. *ACM Trans. Graph.* 31, 4, Article 112 (2012), 112:1–112:8 pages.

N. Rasmussen, D. Nguyen, W. Geiger, and R. Fedkiw. 2003. Smoke simulation for large scale phenomena. In *ACM Trans Graph*, Vol. 22. ACM, 703–707.

Takahiro Sato, Christopher Batty, Takeo Igarashi, and Ryoichi Ando. 2017. A Long-term Semi-lagrangian Method for Accurate Velocity Advection. In *SIGGRAPH Asia 2017 Technical Briefs (SA '17).* ACM, New York, NY, USA, Article 5, 4 pages. https://doi.org/10.1145/3145749.3149443

Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An unconditionally stable MacCormack method. *Journal of Scientific Computing* 35, 2-3 (2008), 350–371.

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. 2014. SPGrid: A Sparse Paged Grid Structure Applied to Adaptive Smoke Simulation. *ACM Trans. Graph.* 33, 6, Article 205 (Nov. 2014), 12 pages.

Karl Sims. 1992. Choreographed image flow. *The Journal Of Visualization And Computer Animation* 3, 1 (1992), 31–43.

Jos Stam. 1999. Stable Fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99).* ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 121–128. https://doi.org/10.1145/311535.311548

Jos Stam and Eugene Fiume. 1995. Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques.* ACM, 129–136.

Yun Teng, David I. W. Levin, and Theodore Kim. 2016. Eulerian Solid-fluid Coupling. *ACM Trans. Graph.* 35, 6, Article 200 (2016), 200:1–200:8 pages.

Jerry Tessendorf. 2015. Advection Solver Performance with Long Time Steps, and Strategies for Fast and Accurate Numerical Implementation.

Jerry Tessendorf and Brandon Pelfrey. 2011. The characteristic map for fast and efficient vfx fluid simulations. In *Computer Graphics International Workshop on VFX, Computer Animation, and Stereo Movies. Ottawa, Canada.*

Steffen Weißmann and Ulrich Pinkall. 2010. Filament-based smoke with vortex shedding and variational reconnection. In *ACM Trans. Graph.*, Vol. 29. 115.

D. C. Wiggert and E. B. Wylie. 1976. Numerical predictions of two-dimensional transient groundwater flow by the method of characteristics. *Water Resources Research* 12, 5 (1976), 971–977. https://doi.org/10.1029/WR012i005p00971

You Xie, Erik Franz, Mengyu Chu, and Nils Thuerey. 2018. tempoGAN: A Temporally Coherent, Volumetric GAN for Super-resolution Fluid Flow. *ACM Trans. Graph.* 37, 4, Article 95 (July 2018), 15 pages. https://doi.org/10.1145/3197517.3201304

Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. 2018. An Advection-reflection Solver for Detail-preserving Fluid Simulation. *ACM Trans. Graph.* 37, 4, Article 85 (July 2018), 8 pages. https://doi.org/10.1145/3197517.3201324

Xinxin Zhang, Robert Bridson, and Chen Greif. 2015. Restoring the Missing Vorticity in Advection-projection Fluid Solvers. *ACM Trans. Graph.* 34, 4, Article 52 (2015), 52:1–52:8 pages.

Yongning Zhu and Robert Bridson. 2005. Animating Sand As a Fluid. In *ACM SIGGRAPH 2005 Papers (SIGGRAPH '05).* ACM, New York, NY, USA, 965–972.