A Long-Term Semi-Lagrangian Method for Accurate Velocity Advection

Takahiro Sato
The University of Tokyo

Christopher Batty
University of Waterloo

Takeo Igarashi
The University of Tokyo

Ryoichi Ando
National Institute of Informatics

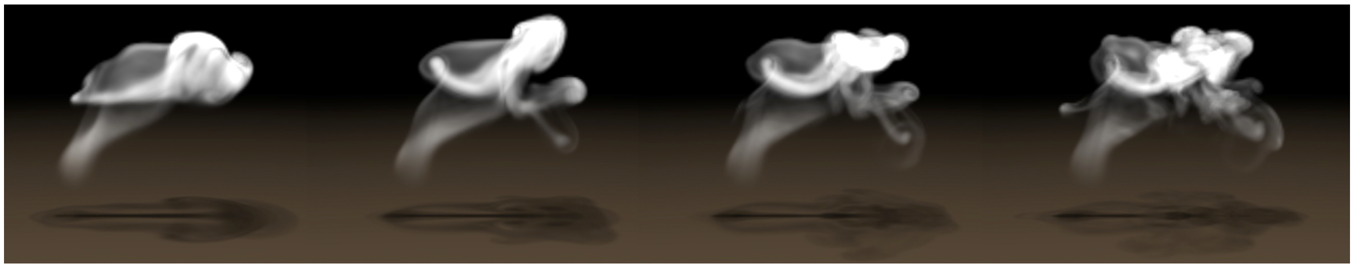

Figure 1: Our method extends the semi-Lagrangian method and corrects the velocity field of the current time step via the integration of the pressure gradient over time. From left to right, semi-Lagrangian advection, our method (for $N = 4$, $N = 8$, $N = 16$) where N denotes the number of preceding time steps used by our method.

ABSTRACT

We introduce a new advection scheme for fluid animation. Our main contribution is the use of long-term temporal changes in pressure to extend the commonly used semi-Lagrangian scheme further back along the time axis. Our algorithm starts by tracing sample points along a trajectory following the velocity field backwards in time for many steps. During this backtracing process, the pressure gradient along the path is integrated to correct the velocity of the current time step. We show that our method effectively suppresses numerical diffusion, retains small-scale vorticity, and provides better long-term kinetic energy preservation.

CCS CONCEPTS

• **Computing methodologies → Physical simulation**;

KEYWORDS

fluid simulation, advection, method of characteristics

ACM Reference format:
Takahiro Sato, Christopher Batty, Takeo Igarashi, and Ryoichi Ando. 2017. A Long-Term Semi-Lagrangian Method for Accurate Velocity Advection. In *Proceedings of SIGGRAPH Asia 2017 Technical Briefs, Bangkok, Thailand, November 27–30, 2017 (SA '17 Technical Briefs),* 4 pages.
https://doi.org/10.1145/3145749.3149443

1 INTRODUCTION

An accurate velocity advection scheme is an essential component for any visually pleasing fluid simulation. Today, the MacCormack scheme [Selle et al. 2008] has become the state-of-the-art Eulerian scheme in practice due to its ease of implementation and cost-effective accuracy over first-order semi-Lagrangian schemes [Stam 1999]. Nevertheless, challenges remain. Artificial (numerical) diffusion still takes place at every step, leading to a significant dissipation of vorticity and energy over time. Naïvely increasing the resolution does not help, since in general the time step size must also be adjusted according to some CFL number, and the increased resolution leads to significantly larger computational costs. High-order interpolation schemes (e.g., ENO or WENO) can improve accuracy, but involve larger stencils, and the issues listed above persist. Xiu and Karniadakis [2001] provide a more comprehensive discussion of accuracy versus grid resolution in semi-Lagrangian schemes. The Characteristic Map scheme [Tessendorf and Pelfrey 2011], based on the method of characteristics, was developed to reduce the accumulation of dissipation, but thus far it has been limited to essentially passive scalar fields (e.g., smoke density). This paper presents a new alternative: we leverage the time-varying pressure field data retained from previous frames to significantly reduce the detrimental effects of numerical dissipation. In summary, this paper offers the following contributions:

- We derive new equations for advection that effectively minimize numerical dissipation by incorporating the pressure gradient over time.
- Our algorithm offers intuitive control of accuracy, allowing a user to trade off quality against increased computational and memory costs.

- Our method is easy to implement and parallelize, and it outperforms the MacCormack scheme at preserving kinetic energy and vorticity.

2 RELATED WORK

For a review of grid-based fluid simulation we refer to Bridson's textbook [Bridson 2015]. Since our contribution is a new Eulerian advection scheme, we focus our discussion around such methods.

Semi-Lagrangian Method. Semi-Lagrangian advection was introduced to graphics by Stam [1999], with the key advantage of being unconditionally stable regardless of time step [Bridson 2015]. It works by moving a virtual particle one step back in time through the velocity field and (tri-/bi-)linearly interpolating a value at the resulting position. Indeed, for CFL numbers less than one the method is equivalent to a first-order upwind advection scheme. As we show later, this interpolation is the primary source of numerical diffusion.

Selle et al. an unconditionally stable semi-Lagrangian MacCormack method [Selle et al. 2008] that reduces error through extra back and forth steps, and thereby achieving second-order accuracy. While this partially mitigates numerical diffusion, some diffusion arising from the grid interpolation nevertheless remains.

High-order Interpolation. Multilinear interpolation can be replaced with high-order schemes. Essentially non-oscillatory (ENO) [Chi-Shu 1997], weighted ENO (WENO) [Chi-Shu 1997] and the cubic-interpolation pseudo-particle (CIP) scheme [Takewaki and Yabe 1987] are popular approaches, and these methods have successfully been applied in graphics [Foster and Fedkiw 2001; Heo and Ko 2010]. The improvements they offer are due to their increased order of accuracy, whereas our method reduces error introduced by repeated interpolations, separate from the particular interpolation method used. Our results demonstrate that our method with linear interpolation provides qualitatively superior results to the MacCormack method with sixth-order WENO interpolation.

Method of Characteristics. Our method is similar in spirit to the work of Tessendorf and Pelfrey [2011] and that of Hachisuka [2005], in the sense that they used the method of characteristics, albeit to advect a density field rather than the velocity field. These approaches follow a streamline of a virtual particle through the velocity field in a Lagrangian manner, much like the (single-step) semi-Lagrangian method. However, the current density at a position is formulated as the density at the *original* position of a particle at the beginning of a simulation, plus the total density source which the particle has received along the streamline. By tracing back to the starting time, repeated interpolation (and dissipation) are avoided.

We similarly consider material transport along the streamlines of velocity; however, by further considering the role of the pressure field over time, we are able to extend this framework to correctly handle velocity advection, rather than scalar advection.

3 OUR ADVECTION SCHEME

First, we illustrate how to incorporate temporal information into our advection scheme. We begin with the momentum equation of the incompressible Euler equations,

$$\frac{D\boldsymbol{u}(\boldsymbol{x},t)}{Dt} = -\frac{1}{\rho}\nabla p(\boldsymbol{x},t), \tag{1}$$

where D/Dt denotes the material derivative, and $p(\boldsymbol{x},t)$ and $\boldsymbol{u}(\boldsymbol{x},t)$ denote pressure and velocity, respectively, at a position $\boldsymbol{x}$ and a time t. Let S be the trajectory of a particle passively advected by the time-varying velocity field from the beginning of a simulation to a time $t = T$, parameterized by time. Integrating both sides of Eq. (1) over time gives

$$\boldsymbol{u}(\boldsymbol{x}(S(T)),T) = \boldsymbol{u}(\boldsymbol{x}(S(0)),0) - \int_0^T \frac{1}{\rho}\nabla p(\boldsymbol{x}(S(t)),t)dt, \tag{2}$$

where $\boldsymbol{x}(S(t))$ denotes a position on a trajectory S at a time t. For brevity, in the following we use short notations: $\boldsymbol{u}_{S,T} \equiv \boldsymbol{u}(\boldsymbol{x}(S(T)),T)$ and $p_{S,T} \equiv p(\boldsymbol{x}(S(T)),T)$. We will aim to solve Eq. (2) and show that this effectively lessens the numerical dissipation. We outline one step of our simulation in Algorithm 1.

Algorithm 1: Our Simulation Loop

```
1 $\boldsymbol{u}^\star_{S,T} = \boldsymbol{u}_{S,0} - \int_0^T \frac{1}{\rho}\nabla p_{S,t}dt$
2 $\boldsymbol{u}^*_{S,T} = \boldsymbol{u}^\star_{S,T}(\boldsymbol{x} - \Delta t\boldsymbol{u}_{S,T})$
3 $\boldsymbol{u}_{S,T+\Delta t} = \text{project}(\boldsymbol{u}^*_{S,T})$
4 Save $p$ and $\boldsymbol{u}_{S,T+\Delta t}$
```

In the basic semi-Lagrangian method, significant numerical diffusion arises because the velocity is resampled at every time step. We circumvent this issue by reconstructing $\boldsymbol{u}^\star_{S,T}$ from the velocity field at the *beginning* of a simulation (Line 1 of Algorithm 1). This way, our approach does not accumulate numerical diffusion over time. Notice that unlike $\boldsymbol{u}_{S,T}$, the reconstructed velocity $\boldsymbol{u}^\star_{S,T}$ is not exactly divergence-free in the limit of numerical approximation. Therefore, we choose $\boldsymbol{u}_{S,T}$ for backtracing positions to preserve mass conservation in the same spirit as the Fluid-Implicit Particle (FLIP) [Zhu and Bridson 2005] and use $\boldsymbol{u}^\star_{S,T}$ for sampling the intermediate velocity after advection (Line 2 of Algorithm 1) because $\boldsymbol{u}^\star_{S,T}$ need not necessarily be divergence-free. Finally, $\boldsymbol{u}^*_{S,T}$ is projected to be incompressible though the regular pressure projection routine [Bridson 2015] to get the new velocity for the next time step (Line 3 of Algorithm 1).

3.1 Integrating The Pressure Gradient

We compute the integral of the pressure gradient in Eq. (2) by repeating the semi-Lagrangian backtrace until we reach the beginning of a simulation. Hence, we must record both the velocity and pressure fields for all previous time steps. We will later show that this limitation can be partially alleviated, in exchange for some reduction in accuracy. In our examples, we employ second-order accurate Runge-Kutta for backtracing, and choose single point quadrature for the line integration. For example:

$$\int_{T-\Delta t}^{T} \frac{1}{\rho}\nabla p_{S,t}dt \approx \Delta t\nabla p_{S,T-\frac{1}{2}\Delta t}. \tag{3}$$

Like before, we use the divergence-free velocity field $\boldsymbol{u}_{S,T}$ for backtracing positions. At the end of the backtrace we can locate S_0, and substitute into Eq.(2) to complete the calculation of $\boldsymbol{u}^\star_{S,T}$.

3.2 Seeding Integration Tracers

In the above exposition, we assumed we were backtracing only a single point, but the velocity field values sampled on the regular grid

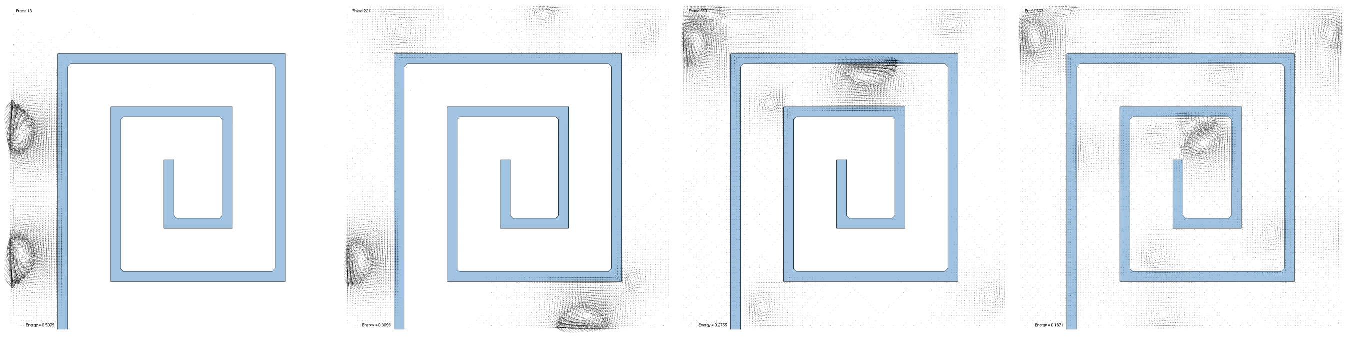

Figure 2: A crawling vorticity experiment using our method ($N = 32$). Vorticity is initiated on the left wall and is allowed to crawl along the spiral walls, ultimately reaching the center of the maze (far right).

are properly interpreted as the average of the velocity over a small cell. Therefore, we should backtrace not a single point but rather a small volume around the sample point. Since true backtracing of a volumetric region would lead to severe geometric tangling, we instead propose to simply seed multiple points (integration tracers) per cell, inspired by a Gaussian quadrature rule.

We seed tracer particles in a uniform grid pattern over each cell, using four tracers per cell in 2D and eight in 3D. Those tracers are separately backtraced and the averaged value is used to compute $\boldsymbol{u}^{\star}_{S,T}$ per cell. This setup is straightforward to extend to staggered configurations, as we do in our examples.

3.3 Reusing The Reconstructed Velocity

As the simulation proceeds, the total number of previous velocity and pressure fields stored continually increases. This eventually leads to a tremendous memory footprint and for practical purposes it becomes infeasible to fetch a velocity from the beginning of the simulation. To overcome this issue, we propose an amendment to allow our method to run at a fixed computational cost regardless of running time. Let N be a target bound on the number of time steps' data to be stored. Eq.(2) can then be re-written as

$$\boldsymbol{u}_{S,T} = \left(\boldsymbol{u}_{S,0} - \int_{0}^{T-N\Delta t} \frac{1}{\rho}\nabla p_{S,t}dt\right) - \int_{T-N\Delta t}^{T} \frac{1}{\rho}\nabla p_{S,t}dt. \quad (4)$$

Notice that Eq.(4) is equivalent to

$$\boldsymbol{u}_{S,T} = \boldsymbol{u}^{\star}_{S,T-N\Delta t} - \int_{T-N\Delta t}^{T} \frac{1}{\rho}\nabla p_{S,t}dt. \quad (5)$$

This way, we can resort to the previously reconstructed $\boldsymbol{u}^{\star}_{S,T-N\Delta t}$ instead of tracing all the way back to $\boldsymbol{u}_{S,0}$. To this end, we additionally store $\boldsymbol{u}^{\star}_{S,T}$ at every time step. When computing Eq.(2), we backtrace at most N steps and fetch $\boldsymbol{u}^{\star}_{S,T-N\Delta t}$ instead of $\boldsymbol{u}_{S,0}$ at

Algorithm 2: Our Simulation Loop (Updated)

```
1 $\boldsymbol{u}^{\star}_{S,T} = \boldsymbol{u}^{\star}_{S,T-N\Delta t} - \int_{T-N\Delta t}^{T} \frac{1}{\rho}\nabla p_{S,t}dt$
2 $\boldsymbol{u}^{*}_{S,T} = \boldsymbol{u}^{\star}_{S,T}(\boldsymbol{x} - \Delta t\boldsymbol{u}_{S,T})$
3 $\boldsymbol{u}_{S,T+\Delta t} = \text{project}(\boldsymbol{u}^{*}_{S,T})$
4 Save $p$ ,$\boldsymbol{u}_{S,T+\Delta t}$ and $\boldsymbol{u}^{\star}_{S,T}$.
```

the point. Although this reintroduces some numerical diffusion, the amount is $O(\frac{1}{N})$ compared to the standard semi-Lagrangian method. For completeness, we assume that $\boldsymbol{u}^{\star}_{S,0} = \boldsymbol{u}_{S,0}$ and $N\Delta t \leq T$. Algorithm 2 lays out one step of our modified algorithm.

3.4 Temporal Filtering

When applied as described, our method can display temporal flickering artifacts; this is because we always fetch the velocity from only the frame N steps back, which allows partial decoupling between sets of frames separated by N steps (e.g., for $N = 4$, frame 5 interpolates its starting velocity from frame 1, whereas frame 6 starts from frame 2, allowing the two sequences to gradually deviate over time). We introduce a temporal filtering technique to mitigate this issue. Instead of sampling velocity from a single frame, we fetch the velocity from multiple sources and blend them together. Our blending recipe is as follows:

$$\boldsymbol{u}^{\bullet}_{S,T} = \frac{1}{W}\sum_{i=1}^{N}\left\langle w_i\boldsymbol{u}^{\star}_{S,T-N_i\Delta t} - w_i\int_{T-N_i\Delta t}^{T}\frac{1}{\rho}\nabla p_{S,t}dt\right\rangle, \quad (6)$$

where $W = \sum_i w_i$ and $N_i = N - i$. To accommodate the effect of our temporal filtering, we replace $\boldsymbol{u}^{\star}_{S,T}$ with $\boldsymbol{u}^{\bullet}_{S,T}$ in Algorithm 2. In our examples, we pick $w_i = \alpha^{i-1}$ where $\alpha < 1$ is a user-specified parameter which we set to $\alpha = 0.9$.

3.5 Static Solids, Liquids and External Forces

To straightforwardly extend our method to support solid boundaries and liquids, rather than explicitly storing pressure, we store the change in velocity due to the pressure projection: $\boldsymbol{u}_{S,T+\Delta t} - \boldsymbol{u}^{*}_{S,T}$. Although this increases the memory consumption, it provides the benefit that we can automatically account for the extrapolated velocity without special care. External forces $\boldsymbol{f}$, such as gravity, buoyancy, or user interaction, can likewise be added to the change: $\boldsymbol{u}_{S,T+\Delta t} - \boldsymbol{u}^{*}_{S,T} + \boldsymbol{f}$.

4 RESULTS

Figure 1 demonstrates how the simulation quality improves as we increase N. This simulation was run with a 128^3 grid on a 10-core Intel(R) Core(TM) i7-6950X CPU 3.00GHz running Linux. Our modified advection scheme took approximately 10 seconds per time step, which corresponds to roughly 60% of the simulation time.

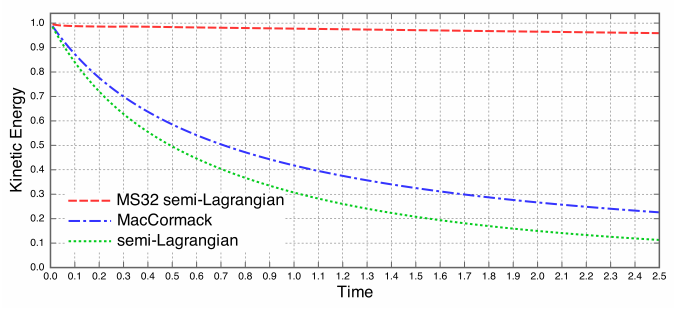

Figure 3: Kinetic energy plot of the 2D Taylor-Green vortex test.

Figure 2 shows a spiral maze experiment as also performed by Mullen et al. [2009]. We set up the same experiment with semi-Lagrangian advection, MacCormack advection with WENO interpolation, and our method with variable N. When N reaches 32, we observed that our method successfully passed the test, in that an initial vortex propagates all the way to the maze's center. We provide the results of other schemes in the supplemental video.

Finally, Figure 3 plots the observed kinetic energy on a 2D Taylor-Green vortex test [Cummins and Rudman 1999]. As expected, our method retains kinetic energy for a longer duration compared to other schemes.

5 DISCUSSION

In practice, the choice of an effective N highly also depends on the accuracy of the integration scheme used. We observed that in two dimensions, our four-point sampling technique typically allowed us to step backwards at most 32 time steps without apparent artifacts. Stepping back more than this induced numerical instabilities, such as velocity fluctuations. We also applied our method for liquids, but found that the visual improvement was subtle. We suspect that this is because interior vorticity does not play a dominant role in many liquid scenes, as also suggested by Zhang et al. [2015].

We explored our method with two different interpolation schemes: bilinear interpolation and sixth-order WENO interpolation. Although WENO interpolation showed a slightly superior accuracy, we felt that the increased runtime did not pay the cost.

Note that although our method devises an advection operator to better retain kinetic energy for a long duration, it does not offer exact preservation. If this is desired, one may prefer to use a strictly energy-preserving integrator [Mullen et al. 2009].

5.1 Limitations

The primary drawback of our method is the added computational cost and memory storage compared to basic semi-Lagrangian advection. These are approximately N times larger, because we must repeat a semi-Lagrangian-style backtracing step N times. Fortunately, our method is fully parallelizable and portable to modern GPUs, which suggests a strong potential for acceleration. Also, the pressure solve step can often dominate the simulation cost (e.g., 90% for smoke [Lentine et al. 2010]) by $O(N_g^2)$ if a preconditioned conjugate gradient method is used, for N_g grid cells. Since the semi-Lagrangian method consumes $O(N_g)$ and our method runs at $O(NN_g)$, our method scales better than the pressure solve if $N < N_g$.

6 CONCLUSION AND FUTURE WORK

This paper introduced a reduced-dissipation velocity advection scheme for fluid animation. The key attribute of our method was to integrate the time-varying pressure gradient along the trajectory to avoid dissipation from resampling the velocity at every time step. Our approach is easy to implement and successfully suppresses numerical diffusion, allowing us to better preserve small-scale turbulence and kinetic energy over the alternative MacCormack advection scheme. In future work we would like to extend our method to minimize the drift of plasticity for Eulerian solid simulation (e.g., Material Point Method), and thus preserve better elasticity.

ACKNOWLEDGEMENTS

This work was supported by NSERC (Grant RGPIN-04360-2014) and JSPS KAKENHI Grant 17H00752.

REFERENCES

R. Bridson. 2015. *Fluid Simulation for Computer Graphics, Second Edition.* Taylor & Francis.

Wang Chi-Shu. 1997. *Essentially Non-Oscillatory and Weighted Essentially Non-Oscillatory Schemes for Hyperbolic Cservation Laws.* Technical Report.

Sharen J Cummins and Murray Rudman. 1999. An SPH Projection Method. *J. Comput. Phys.* 152, 2 (July 1999), 584–607. https://doi.org/10.1006/jcph.1999.6246

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. https://doi.org/10.1145/383259.383261

Toshiya Hachisuka. 2005. Combined Lagrangian-Eulerian Approach for Accurate Advection. In *ACM SIGGRAPH 2005 Posters (SIGGRAPH '05).* ACM, New York, NY, USA, Article 114. https://doi.org/10.1145/1186954.1187084

Nambin Heo and Hyeong-Seok Ko. 2010. Detail-preserving fully-Eulerian Interface Tracking Framework. In *ACM SIGGRAPH Asia 2010 Papers (SIGGRAPH ASIA '10).* ACM, New York, NY, USA, Article 176, 8 pages. https://doi.org/10.1145/1866158.1866198

Michael Lentine, Wen Zheng, and Ronald Fedkiw. 2010. A Novel Algorithm for Incompressible Flow Using Only a Coarse Grid Projection. In *ACM SIGGRAPH 2010 Papers (SIGGRAPH '10).* ACM, New York, NY, USA, Article 114, 9 pages. https://doi.org/10.1145/1833349.1778851

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).* ACM, New York, NY, USA, Article 38, 8 pages. https://doi.org/10.1145/1576246.1531344

Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An Unconditionally Stable MacCormack Method. *J. Sci. Comput.* 35, 2-3 (June 2008), 350–371. https://doi.org/10.1007/s10915-007-9166-4

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

Hideaki Takewaki and Takasi Yabe. 1987. The Cubic-interpolated Pseudo Particle (CIP) Method: Application to Nonlinear and Multi-dimensional Hyperbolic Equations. *J. Comput. Phys.* 70, 2 (June 1987), 355–372. https://doi.org/10.1016/0021-9991(87)90187-2

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.*

Dongbin Xiu and George Em Karniadakis. 2001. A Semi-Lagrangian High-Order Method for Navier-Stokes Equations. *J. Comput. Phys.* 172, 2 (2001), 658 – 684. https://doi.org/10.1006/jcph.2001.6847

Xinxin Zhang, Robert Bridson, and Chen Greif. 2015. Restoring the Missing Vorticity in Advection-projection Fluid Solvers. *ACM Trans. Graph.* 34, 4, Article 52 (July 2015), 8 pages. https://doi.org/10.1145/2766982

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. https://doi.org/10.1145/1186822.1073298