Computational Visual Media
https://doi.org/10.1007/s41095-018-0117-9

Research Article

Spatially adaptive long-term semi-Lagrangian method for accurate velocity advection

Takahiro Sato[1] (✉), Christopher Batty[2], Takeo Igarashi[1], and Ryoichi Ando[3]

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.

Keywords fluid simulation; advection; method of characteristics; spatially adaptive integration; interpolation error correction

1 Introduction

An accurate velocity advection scheme is an essential component for any visually pleasing fluid simulation. Today, the MacCormack scheme [1] has become the state-of-the-art Eulerian scheme in practice, due to its ease of implementation and cost-effective accuracy advantage over first-order semi-Lagrangian schemes [2]. Nevertheless, challenges remain. Artificial (numerical) diffusion still takes place at every step, leading to a significant dissipation of vorticity and energy over time. Naively increasing the resolution does not help, since in general the time step size must also be adjusted according to some CFL (Courant-Friedrichs-Lewy) 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 above persist. Xiu and Karniadakis [3] provide a more comprehensive discussion of accuracy versus grid resolution in semi-Lagrangian schemes. The *characteristic map* scheme [4], based on the method of characteristics, was developed to reduce the accumulation of dissipation. However, application of this method to velocity advection requires non-trivial extensions. This paper presents a feasible solution: 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:

- New equations for advection that effectively minimize numerical dissipation by incorporating the pressure gradient over time.
- Intuitive control of accuracy, allowing a user to trade off quality against increased computational and memory costs.
- A spatially adaptive scheme for long-term semi-Lagrangian backtracing, allowing efficient pressure gradient integration.
- A new error correction scheme to address issues induced by interpolation between grids and tracer particles.
- Our method is easy to implement and parallelize, and it outperforms the MacCormack scheme in preservation of kinetic energy and vorticity.

2 Related work

For a review of grid-based fluid simulation we refer to Bridson's textbook [5]. Since our contribution

1 The University of Tokyo, Tokyo, Japan. E-mail: T. Sato, takahirosato1115@gmail.com (✉); T. Igarashi, takeo@acm.org.

2 University of Waterloo, Waterloo, Canada. E-mail: christopher.batty@uwaterloo.ca.

3 National Institute of Informatics, Japan. E-mail: rand@nii.ac.jp.

Manuscript received: 2018-01-31; accepted: 2018-04-07

is a new Eulerian advection scheme, we focus our discussion around such methods.

2.1 Semi-Lagrangian method

Semi-Lagrangian advection was introduced to graphics by Stam [2], with the key advantage of being unconditionally stable regardless of time step [5]. It works by moving a virtual particle one step back in time through the velocity field and (tri- or 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.

An unconditionally stable semi-Lagrangian MacCormack method [1] reduces error through extra back and forth steps, and thereby achieves second-order accuracy. While this partially mitigates numerical diffusion, some diffusion arising from the grid interpolation nevertheless remains.

2.2 High-order interpolation

Multilinear interpolation can be replaced by high-order schemes. Essentially non-oscillatory (ENO) [6], weighted ENO (WENO) [6], and the cubic-interpolation pseudo-particle (CIP) scheme [7] are popular approaches, and these methods have successfully been applied in graphics [8, 9]. The improvements they offer are due to their increased order of accuracy, whereas our method reduces error introduced by repeated interpolation, separately 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.

2.3 Characteristic map

Our method is similar in spirit to the work of Tessendorf and Pelfrey [4] and that of Hachisuka [10] in the sense that they used the method of characteristics. 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. To apply the characteristic map for velocity advection, Hachisuka [10] proposed to generalize the non-advection terms (e.g., the pressure gradient, and external forces) as the source of change [11] (see Eq. (3.37) for details). Our method shares the same strategy as the work of Hachisuka [10] but differs in that our method consistently fetches the velocity field N steps back after reaching N time steps, while the method of Hachisuka [10] "resets" the total record of velocity at fixed intervals. This brings pros and cons—resetting all previous records in this way may speed up the average simulation time while the effects of dissipation are still reduced by a factor $O(1/N)$ at the cost of (possibly) noticeable temporal artifacts at the time of re-initialization. Our method does not display such artifacts but the performance drag due to backward sampling persists for the entire simulation.

3 Advection scheme

3.1 Overview

For the sake of brevity, we initially omit external forces (e.g., gravity), but they are re-visited in Section 3.8. Firstly, we illustrate how to incorporate temporal information into our advection scheme. We begin with the momentum equation of the incompressible Euler equations:

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

where D/Dt denotes the material derivative, and $p(\boldsymbol{x},t)$ and $\boldsymbol{u}(\boldsymbol{x},t)$ denote pressure and velocity, respectively, at position $\boldsymbol{x}$ and 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

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

where $\boldsymbol{x}(S(t))$ denotes a position on a trajectory S at a time t. For brevity, in the following we use shortened notation: $\boldsymbol{u}_{S,T}$ for $\boldsymbol{u}(\boldsymbol{x}(S(T)),T)$ and $p_{S,T}$ for $p(\boldsymbol{x}(S(T)),T)$. We 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 Simulation loop

```
1: $\boldsymbol{u}^{\star}_{S,T} = \boldsymbol{u}_{S,0} - \int_0^T \frac{1}{\rho}\nabla p_{S,t}\mathrm{d}t$
2: $\boldsymbol{u}^{*}_{S,T} = \boldsymbol{u}^{\star}_{S,T}(\boldsymbol{x} - \Delta t \boldsymbol{u}_{S,T})$
3: $\boldsymbol{u}_{S,T+\Delta t} = \mathrm{project}(\boldsymbol{u}^{*}_{S,T})$
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. We then compute a middle velocity $\boldsymbol{u}^{*}_{S,T}$ in a similar way to the regular semi-Lagrangian method [2]. Note 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) method [12]. We 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 [5] to get the new velocity for the next time step (line 3 of Algorithm 1).

3.2 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 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 line integration. For example:

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

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

3.3 Seeding integration tracers

In the above, we assumed we were backtracing only a single point, but the velocity field values sampled on the regular grid 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 simply seed multiple points (integration tracers) per cell, inspired by a Gaussian quadrature rule.

We place seeds in a uniform grid pattern over each cell, using four tracers per cell in 2D and eight in 3D. The initial velocity of each tracer particle is interpolated from the velocity field on grid faces. The tracer particles are then backtraced in parallel. Finally, their 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.

3.4 Interpolation error correction

When applied, the algorithm above introduces an additional numerical diffusion step associated with back and forth velocity interpolation between grids and tracer particles. This issue can be understood as follows: firstly, we seed tracer particles with velocities interpolated from grids. When we interpolate velocity from tracer particles back to grids (as we do at the end of our advection scheme), the grid velocity is smeared, leading to numerical diffusion as in the *particle-in-cell* (PIC) method. We overcome this issue by predicting this loss of information as Δu, and injecting it back into $\boldsymbol{u}^{\star}_{S,T}$. Our error correction scheme is summarized in Algorithm 2.

Algorithm 2 Our interpolation error estimation

```
1: $u_p \leftarrow$ interpolate($\boldsymbol{u}_{S,T}$)
2: $\boldsymbol{u}_g \leftarrow$ average($u_p$)
3: $\Delta u = \boldsymbol{u}_{S,T} - \boldsymbol{u}_g$
```

We note that when we naively perform this error correction, the kinetic energy can slightly increase in some specific scenarios, e.g., in the 2D Taylor–Green vortex test. Thus, we only correct 90% of the estimated error in all of our examples, except for the comparison test on various ratios of estimated interpolation error (see Fig. 4). This strategy is similar to the work of Zhu and Bridson [12] in the sense that the PIC/FLIP method suggests linearly combining FLIP and PIC where the blending factor is heavily biased towards FLIP.

3.5 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 have a fixed computational cost regardless of running time. Let N be a target bound on the number of time steps' data to be stored. Equation (2) can then be re-written as

$$\boldsymbol{u}_{S,T}=\left(\boldsymbol{u}_{S,0}-\int_0^{T-N\Delta t}\frac{1}{\rho}\nabla p_{S,t}\mathrm{d}t\right)-\int_{T-N\Delta t}^{T}\frac{1}{\rho}\nabla p_{S,t}\mathrm{d}t \tag{4}$$

Note that Eq. (4) is equivalent to

$$\boldsymbol{u}_{S,T}=\boldsymbol{u}^{\star}_{S,T-N\Delta t}-\int_{T-N\Delta t}^{T}\frac{1}{\rho}\nabla p_{S,t}\mathrm{d}t \tag{5}$$

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 the point. When T is smaller than $N\Delta t$, we just stop the backtracing at the beginning.

Although this reintroduces some numerical diffusion, the amount is $O(1/N)$ compared to the standard semi-Lagrangian method. For completeness, we assume that $\boldsymbol{u}^{\star}_{S,0}=\boldsymbol{u}_{S,0}$ and $N\Delta t\leqslant T$. Algorithm 3 lays out one step of our modified algorithm.

Algorithm 3 Simulation loop (modified)

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

3.6 Spatially adaptive integration

Our multi-sampled integration scheme is essential for accurate long-term integration of the pressure gradient, but the effect of such accuracy may not be noticeable where the velocity magnitude is negligibly small. We exploit this observation and apply the following two-level adaptive scheme: we seed a single tracer particle per cell if the velocity magnitude of a cell is less than 0.5, and eight tracers everywhere else. We assign a weight of 1 to the former case, and 1/8 to the latter case, and use these weights later in computing the average velocity on faces. This technique allows us to significantly speed up the integration calculation, since typically the majority of the simulation domain contains velocity values of small magnitude. Where smoke is present, we also seed 8 tracers if the density exceeds a small threshold (0.01 in our tests).

3.7 Temporal filtering

When applied as described, our method can display temporal flickering artifacts: because we always fetch the velocity from only the frame N steps back, this 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 velocities from multiple sources and blend them together. Our blending recipe is as follows:

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

where $W=\sum_i w_i$ and $N_i=N-i$. To accommodate the effect of temporal filtering, we replace $\boldsymbol{u}^{\star}_{S,T}$ with $\boldsymbol{u}^{\bullet}_{S,T}$ in Algorithm 3. In our examples, we pick $w_i=\alpha^{i-1}$ where $\alpha<1$ is a user-specified parameter which we set to $\alpha=0.9$.

3.8 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 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

All examples in Figs. 1–3 were run on a Linux machine with a 10-core Intel Core i7-6950X CPU at 3.00 GHz. We applied interpolation error correction (see Section 3.4) and spatially adaptive integration (see Section 3.6) in all cases except as noted.

Figure 1 demonstrates how simulation quality improves as we increase N. This simulation was run on a 128^3 grid. Our modified advection scheme using $N=16$ took approximately 5.4 s per time step, corresponding to roughly 46% of the simulation time. The right bottom of Fig. 1 shows an example without error correction. Without this correction, the velocity field tends to smooth out more quickly due to

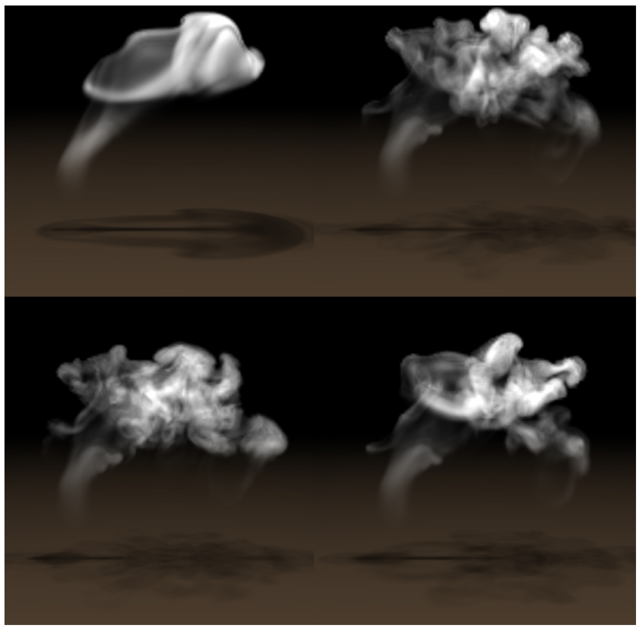

Fig. 1 Three dimensional rising smoke. Left to right, top to bottom: semi-Lagrangian advection, and our method (for $N = 4$, $N = 16$, and $N = 16$ without interpolation error correction); N is the number of preceding time steps used by our method.

numerical diffusion arising from interpolation between grids and tracer particles. Note that in the video in the Electronic Supplementary Material (ESM), some Mach-band-like artifacts are noticeable, but this is solely due to insufficient sample rays in our ray marching algorithms.

Figure 2 shows a spiral maze experiment as also performed by Mullen et al. [13]. We set up the same experiment with semi-Lagrangian advection, MacCormack advection with WENO interpolation, and our method with variable N. When N reached 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 results for other schemes in the ESM.

The bottom of Fig. 2 visualizes the spatial adaptivity used in our method. When applied to Fig. 1, our spatial adaptivity speeds up the backtrace calculation 2.8 times on average.

Finally, Fig. 4 shows kinetic energy plots with various ratios of estimated interpolation error on a 2D Taylor–Green vortex test. When we did not apply our correction, we observed that the kinetic energy decreased significantly. On the other hand, when we corrected 100% of the estimated error, we observed that the kinetic energy increased slightly.

Figure 3 plots the observed kinetic energy on a 2D Taylor–Green vortex test. As expected, our method retains kinetic energy for a longer duration than other schemes.

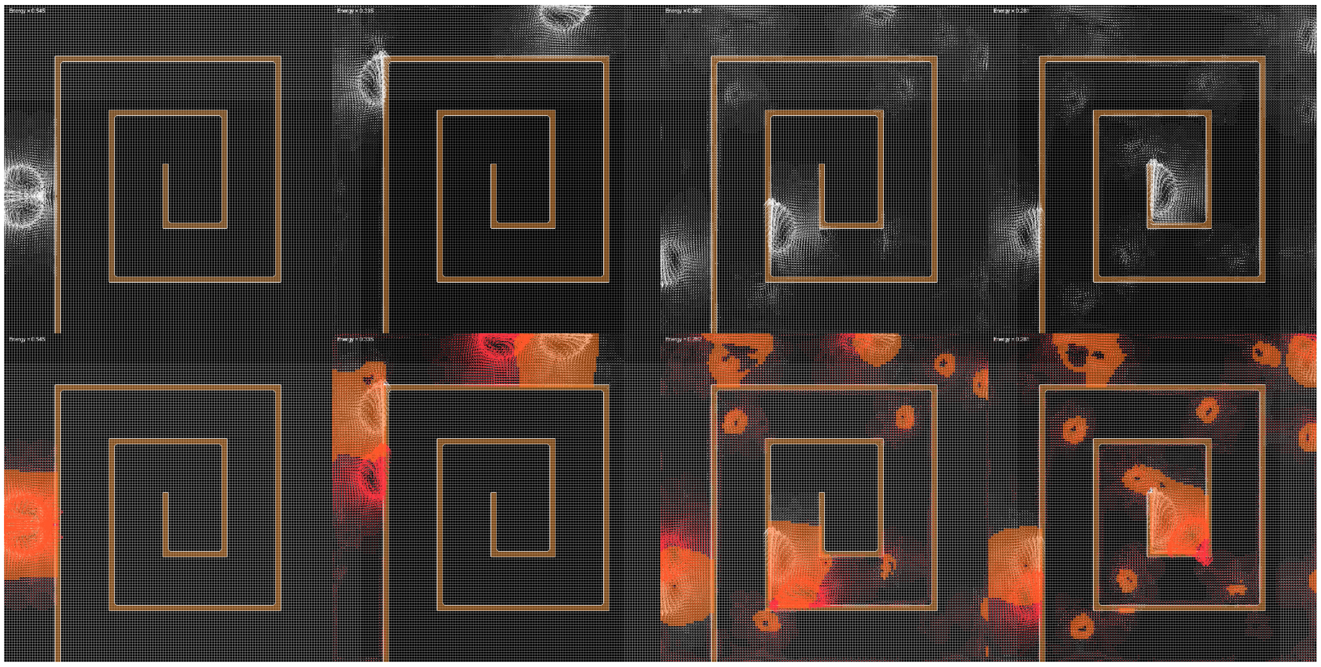

Fig. 2 Top: a crawling vorticity experiment using our method ($N = 32$). Vorticity is initiated on the left wall and is allowed to crawl along the spiral walls, ultimately reaching the center of the maze (far right). Bottom: our adaptivity approach is visualized for a velocity field traced 32 steps back. We seed 4 tracer particles per cell in cells highlighted with red, and use only a single tracer particle everywhere else. The overlaid velocities in red indicate the velocity field 32 steps back.

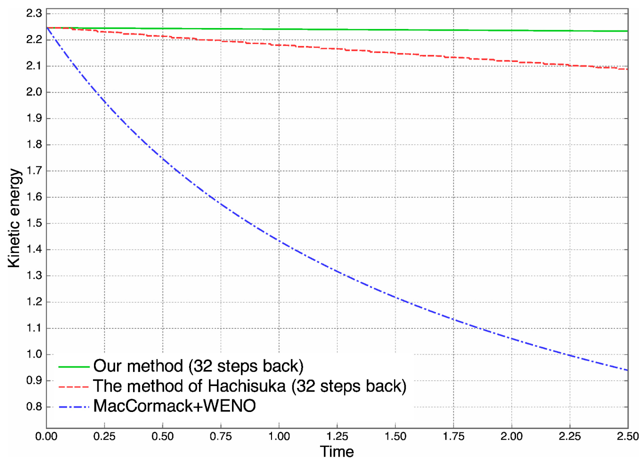

Fig. 3 Kinetic energy in the 2D Taylor–Green vortex test.

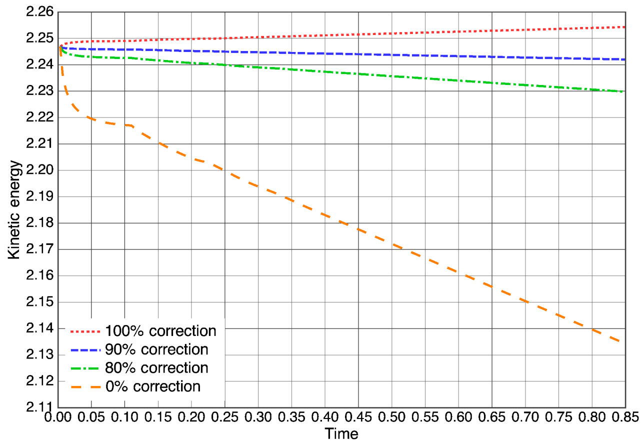

Fig. 4 Kinetic energy in the 2D Taylor–Green vortex test for various ratios of estimated interpolation error.

5 Discussion

5.1 Observations

In practice, the choice of an effective value for N depends critically 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 further than this induced numerical instabilities, such as velocity fluctuations: when N exceeds some tolerable number, our 8- or 4-point integration scheme may not be able to accurately calculate the gradient integral due to significant deformation of grid cells.

In our preliminary tests we tried to adaptively change the maximum backtrace count over space depending on the flow complexity. However, we often fell into the situation that either kinetic energy quickly decreased or numerical diffusion excessively took place, and found it difficult to control the number.

We also applied our method to 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. [14].

We explored use of two different interpolation schemes in our method: tri- or bi-linear interpolation, and sixth-order WENO interpolation. Although WENO interpolation showed slightly superior accuracy, we felt that the increased runtime was not worth the cost. In the 3D rising smoke example (see Fig. 1), the same setup with WENO interpolation took about 19 times longer on average.

Note that although our method devises an advection operator to better retain kinetic energy over a long duration, it does not offer exact preservation. If this was desired, one might prefer to use a strictly energy-preserving integrator [13].

We observed that our interpolation error correction can increase the kinetic energy in some scenarios. Although we were unable to identify the source of the energy increase, it only takes place for a short duration and it eventually decreases in dynamically changing scenarios.

Our correction scheme may introduce an additional step, but we note that its cost is negligible when compared to that of our whole backtracing phase.

5.2 Limitations

The primary drawback of our method is the added computational cost and memory requirements 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 solution step can often dominate the simulation cost (e.g., taking 90% for smoke [15]) by a factor $O(N_g^2)$ if a preconditioned conjugate gradient method is used, for N_g grid cells. Since the semi-Lagrangian method has $O(N_g)$ and our method has $O(NN_g)$, our method scales better than the pressure solution if $N < N_g$.

6 Conclusions and future work

This paper has introduced a reduced-dissipation velocity advection scheme for fluid animation. The key attribute of our method is 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, we would like to extend our method to minimize the drift of plasticity for Eulerian solid simulation (e.g., for the *material point method*), and thus better preserve elasticity.

Acknowledgements

This work was supported by NSERC (Grant RGPIN-04360-2014) and JSPS KAKENHI (Grant 17H00752). The authors thank Toshiya Hachisuka for insightful discussions.

Electronic Supplementary Material Supplementary material is available in the online version of this article at https://doi.org/10.1007/s41095-018-0117-9.

References

[1] Selle, A.; Fedkiw, R.; Kim, B.; Liu, Y.; Rossignac, J. An unconditionally stable MacCormack method. *Journal of Scientific Computing* Vol. 35, Nos. 2–3, 350–371, 2008.

[2] Stam, J. Stable fluids. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, 121–128, 1999.

[3] Xiu, D.; Karniadakis, G. E. A semi-Lagrangian high-order method for Navier–Stokes equations. *Journal of Computational Physics* Vol. 172, No. 2, 658–684, 2001.

[4] Tessendorf, J.; Pelfrey, B. The characteristic map for fast and efficient VFX fluid simulations. In: Proceedings of the Computer Graphics International Workshop on VFX, Computer Animation, and Stereo Movies, 2011.

[5] Bridson, R. *Fluid Simulation for Computer Graphics*, 2nd edn. Taylor & Francis, 2015.

[6] Shu, C. W. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In: *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations. Lecture Notes in Mathematics, Vol. 1697.* Quarteroni, A. Ed. Springer Berlin Heidelberg, 325–432, 1998.

[7] Takewaki, H.; Yabe, T. The cubic-interpolated pseudo particle (CIP) method: Application to nonlinear and multi-dimensional hyperbolic equations. *Journal of Computational Physics* Vol. 70, No. 2, 355–372, 1987.

[8] Foster, N.; Fedkiw, R. Practical animation of liquids. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 23–30, 2001.

[9] Heo, N.; Ko, H.-S. Detail-preserving fully-Eulerian interface tracking framework. In: Proceedings of the ACM SIGGRAPH Asia 2010 Papers, Article No. 176, 2010.

[10] Hachisuka, T. Combined Lagrangian–Eulerian approach for accurate advection. In: Proceedings of the ACM SIGGRAPH 2005 Posters, Article No. 114, 2005.

[11] Hachisuka, T. Advection equation solver using mapping functions. Thesis for Bachelor of Engineering. 2006. Available at https://www.ci.i.u-tokyo.ac.jp/~hachisuka/bt.pdf.

[12] Zhu, Y.; Bridson, R. Animating sand as a fluid. In: Proceedings of the ACM SIGGRAPH 2005 Papers, 965–972, 2005.

[13] Mullen, P.; Crane, K.; Pavlov, D.; Tong, Y.; Desbrun, M. Energy-preserving integrators for fluid animation. In: Proceedings of the ACM SIGGRAPH 2009 Papers, Article No. 38, 2009.

[14] Zhang, X.; Bridson, R.; Greif, C. Restoring the missing vorticity in advection-projection fluid solvers. *ACM Transactions on Graphics* Vol. 34, No. 4, Article No. 52, 2015.

[15] Lentine, M.; Zheng, W.; Fedkiw, R. A novel algorithm for incompressible flow using only a coarse grid projection. In: Proceedings of the ACM SIGGRAPH 2010 Papers, Article No. 114, 2010.

Takahiro Sato is an M.S. student at Computer Science Department, the University of Tokyo. His research interest is focused on physics simulation for applications in computer graphics.

Christopher Batty is an assistant professor of computer science at the University of Waterloo. His research is focused on the development of novel physical simulation techniques for applications in computer graphics and computational physics, with an emphasis on the diverse behaviors of fluids.

Takeo Igarashi is a professor at Computer Science Department, the University of Tokyo. He received his Ph.D. degree from Information Engineering Department, the University of Tokyo, in 2000. His research interest is in user interface in general and current focus is on interaction techniques for 3D graphics.

Ryoichi Ando is an assistant professor at National Institute of Informatics. His research interest is focused on physics simulation for computer graphics, with a strong emphasis on fluid animation.