DOI: 10.1111/cgf.14627
ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2022
D. L. Michels and S. Pirk
(Guest Editors)

Volume 41 (*2022*), *Number 8*

A Second-Order Explicit Pressure Projection Method for Eulerian Fluid Simulation

J. Jiang[1], X. Shen[1], Y. Gong[1], Z. Fan[1], Y. Liu[1], G. Xing[1], X. Ren[2] and Y. Zhang[1]

[1]College of Computer Science, Sichuan University, China
[2]WXG, Tencent, China

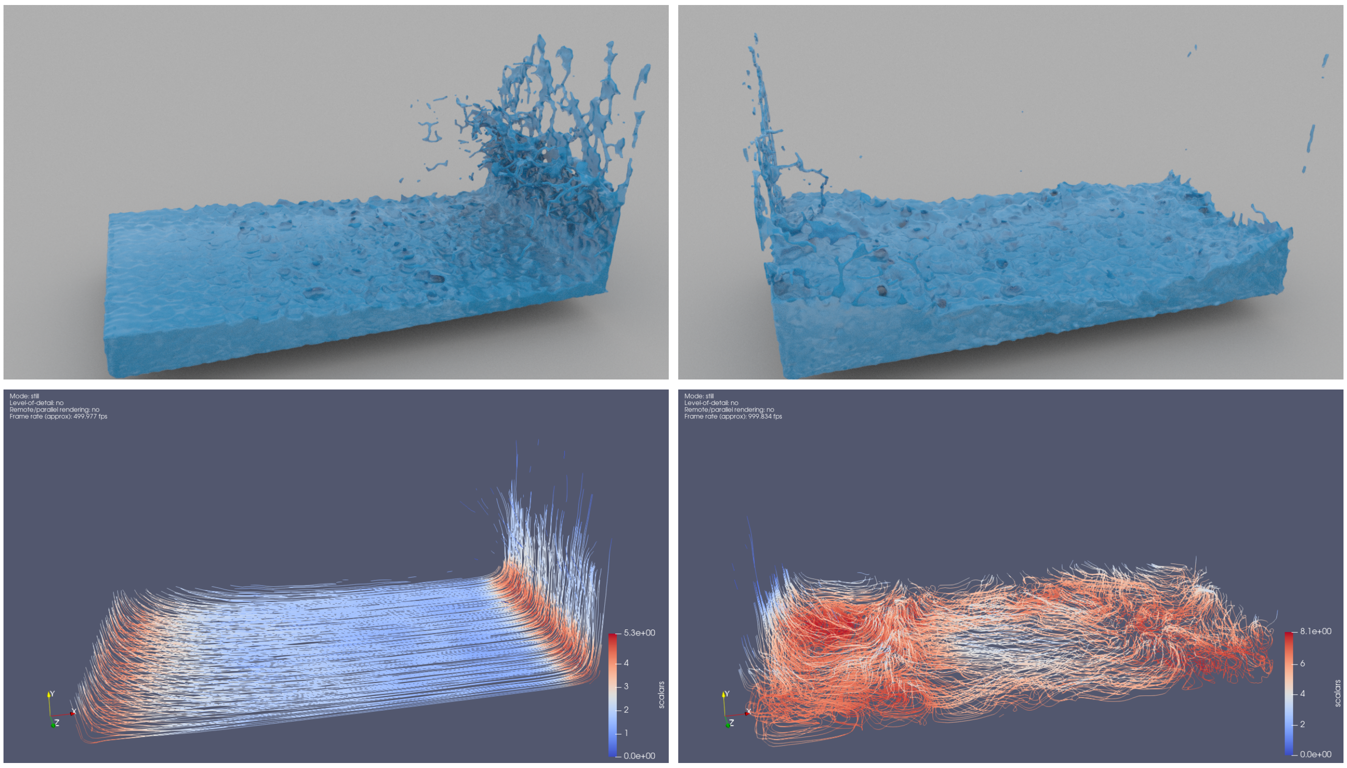

Figure 1: *The Dam Break scenario simulated by our scheme, realistic rendering(top), streamline(down).*

Abstract

In this paper, we propose a novel second-order explicit midpoint method to address the issue of energy loss and vorticity dissipation in Eulerian fluid simulation. The basic idea is to explicitly compute the pressure gradient at the middle time of each time step and apply it to the velocity field after advection. Theoretically, our solver can achieve higher accuracy than the first-order solvers at similar computational cost. On the other hand, our method is twice and even faster than the implicit second-order solvers at the cost of a small loss of accuracy. We have carried out a large number of 2D, 3D and numerical experiments to verify the effectiveness and availability of our algorithm.

CCS Concepts

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

1. Introduction

Benefited from the pioneering work of [FM97] and [Sta99], Eulerian based fluid simulation has achieved great success in the graphics community. They introduced the advection-projection scheme of [CMM90], and used the semi-Lagrangian method to solve the nonlinear advection term, resulting in an unconditionally stable fluid solver. But this also introduces two serious problems, one is the numerical dissipation caused by the semi-Lagrangian advection, and the other is the energy loss caused by the pressure projection.

In the past few years, a lot of work [KLLR05] [SFK*08] [QZG*19] has been proposed to address the first issue. But with respect to the second problem, even if an accurate advection scheme is used, the overall solution accuracy is not good enough[MCP*09] [ZBG15], which manifests as the rapid dissipation of vorticity and the artificial viscosity of fluid motion. The key reason is that all these methods actually employ a first-order treatment of the pressure gradient term.

In order to solve the issue mentioned above, [ZNT18] proposed the reflection solver, which can be interpreted as an implicit midpoint method for the pressure gradient term[NZT19]. But it needs an additional Poisson solution at the middle time, resulting in twice the time cost of traditional first-order methods.

In this work, we propose a velocity solution scheme with full second-order time accuracy and low computational overhead. This scheme accurately considers the influence of pressure gradient on fluid motion, and combines with the advanced advection scheme to obtain a full second-order result. In short, we simplify the reflection solver and consider the influence of pressure gradient on velocity with the idea of second-order explicit midpoint time integration, as shown in Fig. 2. Theoretically, doing this can avoid the expensive multiple calls of Poisson solver while retaining the second-order time accuracy.

To sum up, our paper has the following contributions:

- We propose a second-order explicit midpoint method to exert the influence of pressure gradient on the velocity field to reduce the energy loss caused by projection process. The computational overhead of our method is comparable to the first-order methods and twice and even faster than the implicit second-order solver.
- In order to get more accurate pressure value, we introduce the incremental pressure projection methods in CFD to explicitly calculate and save the pressure, advance it with time, and correct it at the end of each time step;

At the same time, our scheme is purely Eulerian, which is completely parallel. And it is orthogonal to the existing framework, hence can be integrated into it with minor modifications.

2. Related Work

Reviewing the Eulerian fluid simulation in the graphics community can not ignore the pioneering work of [FM97] and [Sta99], who introduced 1) staggered grid [HW65] 2) Chorin-style advection-projection scheme[Cho68] 3) semi-Lagrangian advection [SC91] into the graphics community. This brings stability, but also introduces the famous and huge numerical dissipation, which will lead to the reduction of fluid energy and the rapid loss of fluid details such as rotational motion. As mentioned above, there are two main reasons: one is the numerical dissipation caused by the semi-Lagrangian advection, and the other is the energy loss caused by the pressure projection. In the past 20 years, a lot of work has been done to improve these defects, we will make a brief review here.

Higher-order advection methods. The error caused by semi-Lagrangian advection is generally considered to be the main reason of the above phenomenon. The improvement of the advection term is mainly divided into two parts over time. On the one hand, many people continue to improve the grid based advection. For example, [KLLR05] introduced the idea of back and forth error compensation and correction methods (BFECC)[DL03] into advection, and expect to use the first-order semi-Lagrangian construction module to construct the second-order advection algorithm; [SFK*08] improved the efficiency of BFECC to reduce the computational overhead without losing its accuracy. [QZG*19] introduced the method of characteristic mapping (MCM) and showed high advection accuracy in their experiments. On the other hand, a lot of work has turned to the hybrid grid particle method, which has more advantages in dealing with advection. According to [Bri15], this can be traced back to the work of Los Alamos National Laboratory in the early 1950s, it uses particles to deal with the kinematic properties of fluid, and hands over the mechanical calculation to the grid, which named as particle-in-cell (PIC). Many subsequent works continue this idea. [ZB05] improved the transmission process from direct sampling to incremental updating, which greatly reduced the numerical dissipation of PIC, but also introduced some noise. The work of [JSS*15] proved that even incremental updating greatly lost the motion information of fluid, so they developed a series of methods to solve this problem, such as APIC[JSS*15] and PolyPIC[FGG*17].

Energy conservation. Independent of the improvement of the advection algorithm, a lot of work turn to the projection process. This is due to the discovery of [ETK*07] and [ZBG15] that even if the high-order advection scheme is used, the projection step will still removes energy from fluid. Around this problem, [SU94] and [FSJ01] calculated a force that does not exist in practice to confine the vorticity. [MCP*09] applied the influence of pressure on fluid motion in the form of implicit Crank-Nicolson-style, which need an expensive nonlinear Newton solver. In contrast, [ZNT18] approximate their idea in a simpler way, they consider the effect of pressure on the fluid by using a reflection operation at the middle time, which is more like the idea of implicit midpoint method[NZT19].

Vortex methods. In addition to the fluid simulation based on velocity, the vortex method is very useful in maintaining the rotating motion of the fluid. But it uses vorticity instead of velocity to represent the fluid, which brings difficulties to the large-scale application of the industry. Since our method does not belong to the vortex method, we will only list some interesting work here. Such methods include [SRF05]; [PK05]; [ZB14]; [Ang17]; [PCK*19]; [YXZ*21] and so on.

3. Theory

In this part, we will first make a simple review of the first-order advection-projection solver and second-order reflection solver in

Table 1: *Physical quantities and symbols*

Symbols	Meaning
$u^n, u^{n+1/2}, u^{n+1}$	Fluid velocity field
$\tilde{u}^{n+1/2}, \tilde{u}^{n+1}$	Intermediate fluid velocity field after advection
$\hat{u}^{n+1/2}$	Intermediate fluid velocity field after reflection
p^n, p^{n+1}	Fluid pressure field
$\tilde{p}^{n+1}$	Intermediate fluid pressure field after advection
$\nabla \tilde{p}_{exp}^{n+1/2}$	Intermediate pressure gradient field by **explicitly** estimate
$\nabla \tilde{p}_{imp}^{n+1/2}$	Intermediate pressure gradient field by **implicitly** calculate
ϕ^{n+1}	Pressure increment field obtained by projection

Sec. 3.1 and Sec. 3.2 respectively. And then the basic idea of our second-order explicit solver is presented in Sec. 3.3. Finally, more details of our algorithm are discussed in Sec. 3.4 - 3.5. For the sake of clarity, the physical quantities used in this paper and their symbols are as shown in Tab. 1.

In the following analysis, we use Taylor vortex as an intuitive case to help illustrating the basic ideas of various solvers, including ours, as shown in Fig. 2. Considering a specific point x (red dot) at time step $n+1$ in the flow field, its position at last time step n can be achieved by a backward mapping $\Psi(x;u^n,\Delta t)$(blue dot), as illustrated in Fig. 2(b).

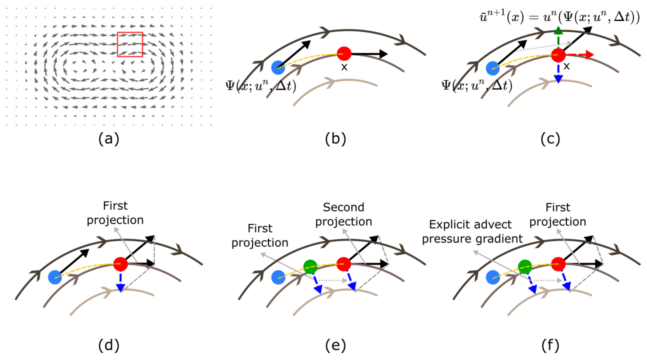
$\tilde{u}^{n+1}(x) = u^n(\Psi(x;u^n,\Delta t))$
$\Psi(x;u^n,\Delta t)$
x
(a)
(b)
(c)
First projection
Second projection
Explicit advect pressure gradient
(d)
(e)
(f)

Figure 2: *An intuitive interpretation of various solvers: (a) Velocity field of Taylor vortex; (b) Take position x at time step n+1 as an example; (c) Advection; (d) An implicit first-order treatment of the pressure gradient term; (e) The reflection solver is actually an implicit second-order method which requires an additional call of Poisson solver; (f) Our method greatly reduces the computational overhead while maintaining the second-order accuracy.*

3.1. Advection-Projection Solver

For the incompressible Navier-Stokes equations, if we do not consider the fluid viscosity and external force, they degenerate into the following form:

$$\frac{\partial u}{\partial t} + u \cdot \nabla u = -\frac{1}{\rho}\nabla p \qquad (1)$$
$$\nabla \cdot u = 0$$

where u, p, ρ represent velocity, pressure and density respectively. Spatially, we use the standard MAC grid discretization, and in time, the advection-projection method [Sta99] discretizes Equ. 1 into two main steps: advection $\mathcal{A}$ and projection $\mathcal{P}$.

The advection step $\mathcal{A}$ solves the advection term in the Navier-Stokes equation, but ignores the non divergence constraint to obtain an intermediate velocity field $\tilde{u}^{n+1}(x) = u^n(\Psi(x;u^n,\Delta t))$, as shown in Fig. 2 (c). This step usually pulls the velocity field into a divergent space. Then the projection step $\mathcal{P}$ is to enforce the non divergence constraint and maintain mass conservation, as shown in Fig. 2 (d), which is actually a Helmholtz-Hodge decomposition.

$$\frac{u^{n+1} - \tilde{u}^{n+1}}{\Delta t} = -\frac{1}{\rho}\nabla p^{n+1} \qquad (2)$$

Note that in this solver, the effect of pressure gradient on velocity is only considered in the backward Euler integral scheme, like Equ. 2, which is only first-order time accuracy. As [ETK*07] and [ZBG15] found, doing this significantly reduces the kinetic energy of the fluid, resulting in the rapid loss of fluid details such as vortex motion.

3.2. Second-Order Implicit Solver

To address above issue, [ZNT18] proposed a reflection solver, which can be interpreted as an implicit midpoint time integration scheme[NZT19]. That is, for the pressure gradient term, replace ∇p^{n+1} in Equ. 2 with $\nabla \tilde{p}_{imp}^{n+1/2}$ to get Equ. 3:

$$\frac{u^{n+1} - \tilde{u}^{n+1}}{\Delta t} = -\frac{1}{\rho}\nabla \tilde{p}_{imp}^{n+1/2} \qquad (3)$$

Algorithm 1: Reflection solver (one time step)

```
Input: Original velocity: $u^n$
Output: Updated velocity: $u^{n+1}$
1 $\tilde{u}^{n+1/2} = \mathcal{A}(u^n; u^n, \Delta t/2)$          ▷ Advect velocity
2 $u^{n+1/2} = \mathcal{P}(\tilde{u}^{n+1/2})$                     ▷ First projection
3 $\hat{u}^{n+1/2} = 2u^{n+1/2} - \tilde{u}^{n+1/2}$                ▷ Reflection
4 $\tilde{u}^{n+1} = \mathcal{A}(\hat{u}^{n+1/2}; u^{n+1/2}, \Delta t/2)$  ▷ Advect velocity
5 $u^{n+1} = \mathcal{P}(\tilde{u}^{n+1})$                         ▷ Second projection
```

The pseudo code of reflection solver is as shown in Alg. 1, the first projection step implicitly calculates the pressure gradient approximation at the middle time (line 2), and the reflection operation applies it to the velocity field for the whole time step (line 3). At the same time, in order to strictly meet the incompressible condition at the end of each time step, an additional projection operation is performed at the end (line 5).

However, as mentioned above, the reflection solver needs to call the Poisson solver twice (line 2 and 5 in Alg. 1), which almost double the time cost of the traditional first-order scheme.

3.3. Our Second-Order Explicit Solver

In order to avoid the extra computational overhead caused by the implicit method and maintain the second-order time accuracy, we choose to estimate the pressure gradient at the middle time explicitly, and obtain our second-order explicit midpoint time integration scheme for the pressure gradient term, as shown in Equ. 4.

$$\frac{u^{n+1}-\tilde{u}^{n+1}}{\Delta t} = -\frac{1}{\rho}\nabla\tilde{p}_{exp}^{n+1/2} \tag{4}$$

So we need to find a way to explicitly estimate $\nabla\tilde{p}_{exp}^{n+1/2}$. In numerical calculations, if we want to advance a fluid variable (∇p) over time explicitly, just find its time derivative $\frac{\partial(\nabla p)}{\partial t}$. In the Eulerian fluid simulation, for each grid point, the change of pressure gradient with time is composed of two parts: one is caused by advection, the other is the pressure increase caused by the current grid fluid inflow or the pressure decrease caused by fluid outflow. The latter is expressed by $f(x,t)$, which is a function of spatial position x and time t, so the above change can be written in the form of Equ.5:

$$\frac{\partial(\nabla p)}{\partial t} = -u\cdot\nabla(\nabla p) + f(x,t) \tag{5}$$

For explicit advancing, we fit the first term in Equ.5 as the approximate rate of change of pressure gradient, that is, advect ∇p^n to $n+1/2$ to get our simple approximation of $\nabla\tilde{p}_{exp}^{n+1/2}$. The pseudo code of the above process is shown in Alg. 2($\mathcal{G}$ represents calculate gradient). Compared with the implicit(reflection) method, this will cost some accuracy, which is caused by the existence of the second term and the error of the advection operation itself, but our numerical experiments (Sec. 4.4) show that it is small enough and can be exchanged for twice or more efficiency improvements (Sec. 4.1).

Algorithm 2: Estimate $\nabla\tilde{p}_{exp}^{n+1/2}$ with right order

```
Input: Original pressure: $p^n$
Output: Intermediate pressure gradient: $\nabla\tilde{p}_{exp}^{n+1/2}$
1 $\nabla p^n = \mathcal{G}(p^n)$                          ▷ Calculate gradient
2 $\nabla\tilde{p}_{exp}^{n+1/2} = \mathcal{A}(\nabla p^n; u^n, \Delta t/2)$   ▷ Advect gradient
```

Algorithm 3: Estimate $\nabla\tilde{p}_{exp}^{n+1/2}$ with wrong order

```
Input: Original pressure: $p^n$
Output: Intermediate pressure gradient: $\nabla\tilde{p}_{exp}^{n+1/2}$
1 $\tilde{p}^{n+1/2} = \mathcal{A}(p^n; u^n, \Delta t/2)$          ▷ Advect pressure
2 $\nabla\tilde{p}_{exp}^{n+1/2} = \mathcal{G}(\tilde{p}^{n+1/2})$             ▷ Calculate gradient
```

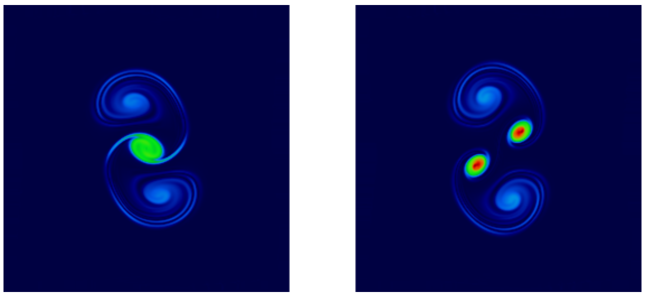

Figure 3: *Estimate $\nabla\tilde{p}_{exp}^{n+1/2}$ with wrong order(left), the results obtained have only first-order accuracy, our scheme can obtain second-order accuracy(right).*

It should be emphasized that if first advect $\Delta t/2$ to get $p^{n+1/2}$ and then calculate its gradient, it is different from the above process. The latter (Alg. 3) is hardly helpful to improve the velocity accuracy, e.g. Fig. 3. This is because the pressure gradient after advection carries direction information, which is very important to improve the accuracy of the solution, as shown in Fig. 4.

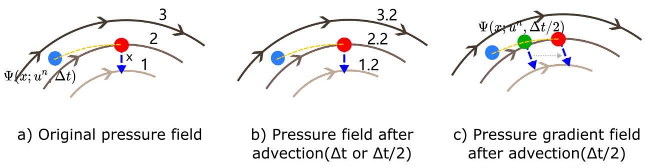
3
2
1
x
$\Psi(x; u^n, \Delta t)$
3.2
2.2
1.2
$\Psi(x; u^n, \Delta t/2)$
a) Original pressure field
b) Pressure field after advection(Δt or Δt/2)
c) Pressure gradient field after advection(Δt/2)

Figure 4: *(a) A simplified original pressure field conforming to pressure gradient distribution(the number represents the pressure distribution); (b) Pressure field after advection, if we first advect the pressure and then calculate its gradient, it will lose the key direction information; (c) We apply the pressure gradient at n+1/2 ($\Psi(x;u^n,\Delta t/2)$, marked by the green particle) to the velocity field. Equivalent to advect original pressure gradient field to n+1/2.*

To sum up, we get our algorithm as in Alg. 4. And we will explain line 4 and line 7 of Alg. 4 in Sec. 3.4.

Algorithm 4: Our solver(one time step)

```
Input: Original velocity and pressure: $u^n$, $p^n$
Output: Updated velocity and pressure: $u^{n+1}$, $p^{n+1}$
1 $\tilde{u}^{n+1} = \mathcal{A}(u^n; u^n, \Delta t)$                    ▷ Advect velocity
2 $\nabla p^n = \mathcal{G}(p^n)$                          ▷ Calculate pressure gradient
3 $\nabla\tilde{p}_{exp}^{n+1/2} = \mathcal{A}(\nabla p^n; u^n, \Delta t/2)$   ▷ Advect pressure gradient
4 $\tilde{p}^{n+1} = \mathcal{A}(p^n; u^n, \Delta t)$                    ▷ Advect pressure
5 $\tilde{u}^{n+1} = \tilde{u}^{n+1} - \Delta t\nabla\tilde{p}_{exp}^{n+1/2}$          ▷ Apply pressure gradient
6 $u^{n+1} = \mathcal{P}(\tilde{u}^{n+1})$                             ▷ Projection
7 $p^{n+1} = \tilde{p}^{n+1} + \phi^{n+1}$                    ▷ Pressure correction
```

3.4. Pressure Incremental Correction

The premise of the discussion in Sec. 3.3 is that we have the pressure field p^n at the beginning of time step. In the advection-projection framework, the pressure will not be saved, but an accessory generated by Helmholtz-Hodge decomposition, the purpose is to make $\nabla \cdot u^{n+1} = 0$. What makes the matter worse is that since we need to explicitly calculate the effect of the gradient of pressure on the velocity, in order to obtain a more accurate velocity, a more accurate pressure is necessary. But the pressure obtained by advection-projection framework has only the first-order accuracy in time. Therefore, we introduce the pressure incremental projection scheme[BCM01] to explicitly save and calculate the more accurate pressure, while the potential field ϕ^{n+1} generated by Helmholtz-Hodge decomposition just correct the pressure field we got before(Alg. 5).

Algorithm 5: Calculate p^{n+1} with lagging pressure p^n

```
Input: Original pressure: $p^n$, potential field: $\phi^{n+1}$ obtained
       by Helmholtz-Hodge decomposition
Output: Updated pressure ($p^{n+1}$)
1 $p^{n+1} = p^n + \phi^{n+1}$                ▷ Pressure correction
```

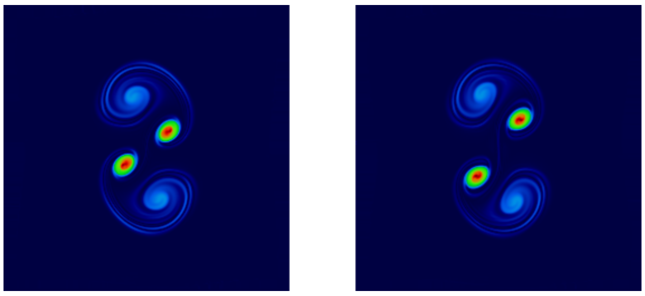
Figure 5: *The influence of advect pressure or not in small scale rotational motion. In the case of small Δt, and the fluid is mainly rotating, the pressure field changes little with time, the error introduced by advecting pressure(left) is greater than that by using lagging pressure field(right).*

Note that in Eulerian fluid simulation, all physical quantities need to be advected, but in the paper of [BCM01], they directly correct the lagging pressure field p^n at the previous time step. This is because the CFD field always aims at accuracy and will not pursue large Δt, therefore, for each grid point, there is little pressure difference between the two time steps. Our experiments confirm that this has better accuracy only in the case of small step size and insignificant flow translational motion(see Fig. 5). But in the case of large Δt and more complex fluid motion, pressure advection is necessary(see Fig. 6). So we choose to correct the advected pressure $\tilde{p}^{n+1}$ at the end of each time step instead of directly correcting the lagging pressure field p^n[BCG87] [BCM01], as Alg. 4 line 4 and line 7, even if this introduces another error: the error caused by the inaccuracy of advection operation, and this error can be eliminated as much as possible by using high-order advection scheme.

For pressure increment projection method, we have one more thing to do: get our initial pressure(p^0). In the paper of [BCG87], it is suggested to use an iterative method to obtain the initial pressure field at the beginning of the first time step. For convenience, we

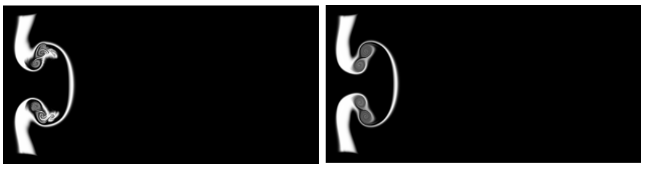
Figure 6: *The influence of advect pressure or not in large scale advective motion. In the scenario with large Δt and obvious fluid advection motion, the local pressure $(\partial p/\partial t)$ at the grid point changes greatly with time, the error introduced by using lagging pressure field(left) is greater than that by advecting pressure(right).*

choose to execute a reflection solver at the first time step and save the final pressure field generated by two Helmholtz-Hodge decomposition as our initial pressure.

3.5. Additional Algorithm Details

In this part, we explain some specific algorithm details.

Boundary Conditions. Because we introduce explicit pressure and make use of it, we set simple boundary conditions for it in consideration of stability. For the empty grid, we set its pressure to zero; for the fluid grid, the pressure refers to Alg. 4 to advect and update; for the grid in the solid boundary, we set it as the average value of the fluid grid pressure around it. It is worth mentioning that many CFD literatures(like [BCG89]) believe that only velocity boundary conditions is needed, otherwise the system may be in an over determined state. However, in our experiments, instability may occur at the boundary. Therefore, we set simple pressure boundary conditions to solve such problems. We will leave more detailed research for future work.

MacCormack Limiter. According to the description of [SFK*08], the MacCormack method needs to use an extrema limiter to prevent its numerical oscillations. In our implementation, the advection result of the MacCormack method is limited by the result values of its first backward tracking.

4. Results

We have implemented our GPU solver based on [Bri15]. The following solvers are also implemented for comparison: stable fluids with semi-Lagrangian advection (SF)[Sta99], stable fluids with MacCormack advection (MC)[SFK*08], MacCormack advection with reflection solver (MC + R)[ZNT18], MacCormack advection with second-order reflection solver (MC + R^2)[NZT19], MacCormack advection with BDF2 (MC + BDF2)[NZT19], and MacCormack advection with our method(MC + Ours). For some experimental scenarios, we use higher-order backtrace mapping (RK3) in advection, for these cases, we have made special notes in the figure and in the text, such as (MCRK3). All the experiments are conducted on a desktop machine with a Ryzen 7 5800X @ 3.80GHz CPU, 32GM RAM and a nVidia GeForce RTX 2080 SUPER graphics card. In order to visualize the simulation results, we use the configuration of [QZG*19] for 2D cases and use Houdini to render the 3D effects.

Table 2: *Simulation configuration and performance.*

Scene	Domain/m^2,m^3	Resolution	CFL	Δt/s	Solver	Avg. comp. time/ms
Taylor Vortex(Fig. 7 and Fig. 8)	$2\pi \times 2\pi$	256×256	∞	0.025	SF	83
				0.025	MC	85
				0.025	MC+BDF2	82
				0.050	MC+R	156
				0.050	MC+R^2	155
				0.025	**MC+Ours**	**68**
2D Vortex Leapfrogging(Fig. 9)	$4\pi \times 2\pi$	512×256	∞	0.1	MC+BDF2	147
				0.2	MC+R	273
				0.2	MC+R^2	275
				0.1	**MC+Ours**	**130**
Vortex Sheet(Fig. 10)	$2\pi \times 2\pi$	256×256	∞	0.025	SF	78
				0.025	MC	79
				0.025	MC+BDF2	79
				0.050	MC+R	141
				0.050	MC+R^2	146
				0.025	**MC+Ours**	**60**
3D Smoke Plumes(Fig. 11)	$5 \times 20 \times 5$	$128 \times 512 \times 128$	∞	0.04	SF	3023
				0.04	MC	3144
				0.04	MC+BDF2	3053
				0.04	MC+R	3531
				0.04	MC+R^2	3578
				0.04	**MC+Ours**	**1216**
Dam Break(Fig. 12)	$32 \times 16 \times 16$	$128 \times 64 \times 64$	∞	0.05	FLIP99	52
				0.05	**FLIP99+Ours**	**54**
Taylor Green Vortex(Fig. 13 and Fig . 14)	$2\pi \times 2\pi$	256×256	∞	/	/	/

In order to validate the algorithm proposed in this paper, we have conducted the following experiments. Firstly, we compare the computational overhead of different solvers in different experimental scenarios to verify the efficiency of our algorithm, as shown in Sec. 4.1. Secondly, in order to test the preservation of fluid energy and vorticity, we test our algorithm in some 2D scenes, as shown in Sec. 4.2. For fairness, reflection(R) and second-order reflection(R^2) solver use twice the time step size of other algorithms[ZNT18] in 2D scenes. Then, we test some 3D scenes (smoke and water) to verify the practicability of our algorithm, as shown in Sec. 4.3. Finally, we conduct rigorous numerical experiments to verify the second-order time accuracy of our algorithm, as shown in Sec. 4.4.

4.1. Performance

In order to compare the computational overhead of various solvers, we test these solvers on multiple data set with different configurations. The performance data as shown in Tab. 2.

In most scenarios, our algorithm shows its theoretical efficiency: we obtains the effect of second-order accuracy with the same (or even less) computational overhead as the first-order advection-projection algorithm. In contrast, as mentioned in the paper of [ZNT18], the computational overhead of R and R^2 solver is twice that of other algorithms due to its additional advection and projection operations.

In addition, we find that our algorithm showed dramatic performance improvements in 3D smoke scene(high resolution scene). This is because our method has applied the nearly correct pressure gradient to the velocity field before performing the most time-consuming pressure projection, resulting in the input velocity field of Helmholtz-Hodge decomposition is almost non divergence. Therefore, the conjugate gradient solver only needs a few iterations to meet the convergence requirements. At the same time, this is also the reason why the computational cost of the R and R^2 solver in this scene is not twice as high as that of the traditional algorithm. The reflection operation makes the input velocity field of the second Helmholtz-Hodge decomposition accurate enough and only needs a small number of iterations to exit. Therefore, the main computational cost of the R and R^2 solver in this scene comes from the first pressure projection process, resulting in an efficiency close to that of the traditional algorithm. This result proves that our algorithm has potential performance improvement beyond theory.

4.2. 2D Results

Taylor Vortex. We follow the settings in [McK07]: two vortices are placed close to each other, and the initial distance between them will cause the two vortices to merge or separate after a period of time. [QZG*19] and [McK07] set this distance to 0.81 and 0.8 respectively so that their algorithm can separate two vortices. Our algorithm can accept the a even smaller distance 0.79, which is closer to the critical separation distance. The experimental results are shown in Fig. 7. It is worth noting that BDF2 and R^2 have achieved amazing results, and our algorithm is slightly worse than

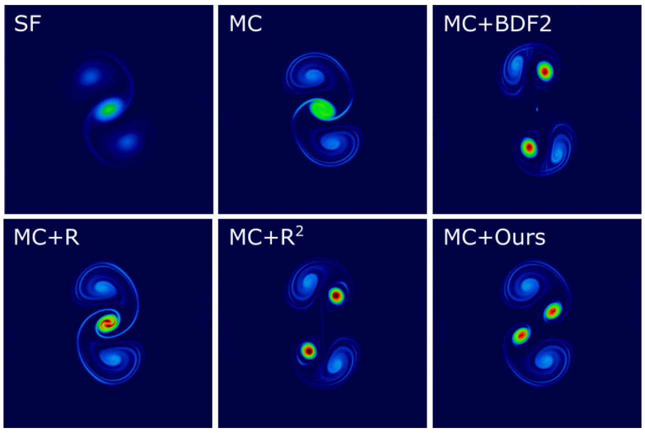

Figure 7: *Taylor Vortices results at t = 7.5 seconds, with initial distance 0.79. Color indicates the vorticity magnitude.*

the above two, but better than other algorithms. As analyzed in Sec. 3.4, this is due to the introduction of large errors into the pressure field by using low-precision advection. In this scenario, better results can be obtained by directly using the lagging pressure field.

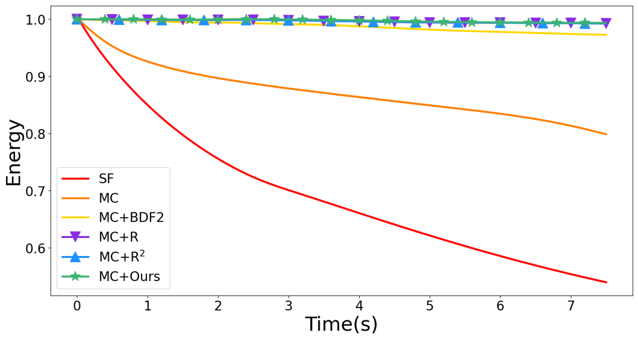

Figure 8: *Evolution of total kinetic energy of fluid with time on the Taylor Vortex scenario, normalized by the initial energy.*

At the same time, we also calculated the change of the total kinetic energy of the fluid with the advance of time, as shown in Fig. 8. Since no energy is injected into the scene after initialization, a good kinetic energy conservation algorithm can maintain the initial energy. We find that both the R, R^2 solver and our method can achieve this goal.

2D Vortex Leapfrogging. We follow the settings in [QZG*19], two vortex pairs are initialized with the same energy intensity on the left side of the scene and advance them over time. The experimental results are shown in the Fig. 9 first column. It can be seen that BDF2 produces wrong velocity field prediction, and our method produces similar results as the R solver. In contrast, the R^2 solver performs better, which is due to its modification of the advection term. In other words, compared with the other schemes, its treatment of the advection term is more accurate. In order to eliminate the influence of advection accuracy, we uniformly improve the accuracy of the advection algorithm (change the backtrace module of MC from RK1 to RK3), then the advantage of our algorithm in correcting the pressure gradient term can be shown. As in Fig. 9 second column, our method maintains the vortex motion to the greatest extent and has no asymmetry and other errors. Therefore, the computational cost saved by our method can be used for more accurate advection, so as to achieve the optimal compromise between computational cost and effect.

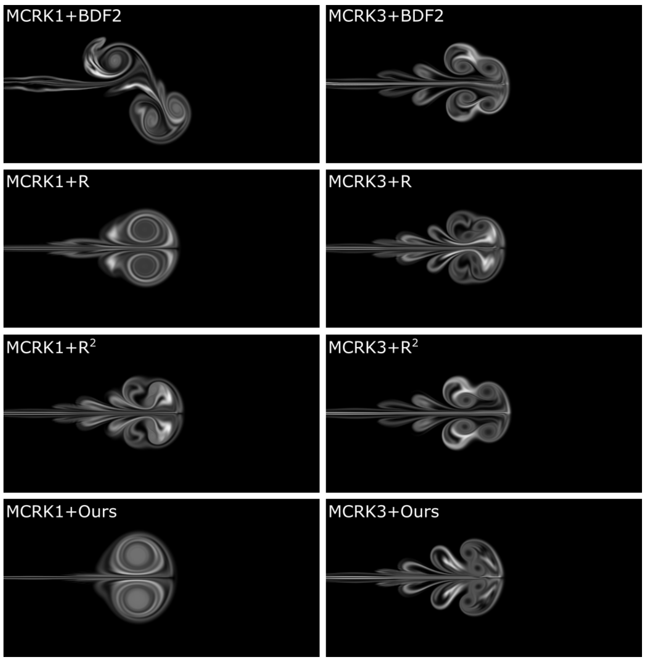

Figure 9: *Two vortex pairs leapfrogging results at t = 70 seconds, the density field is visualized.*

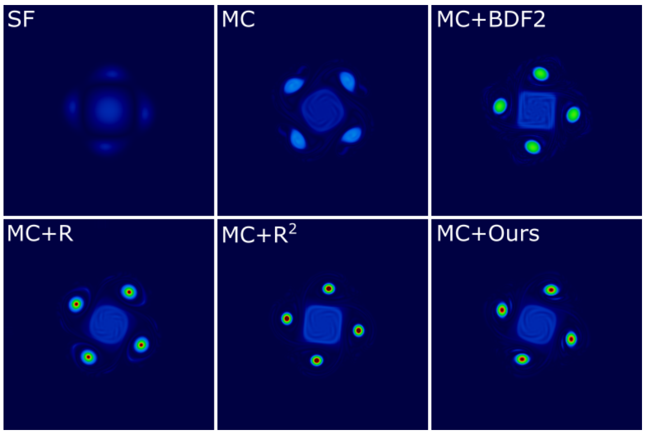

Figure 10: *Vortex sheet results at t = 15 seconds, color indicates the vorticity magnitude.*

Vortex Sheet. A disc-shaped region in the center of the scene is initialized with a rigid rotation. As time goes on, it will produce

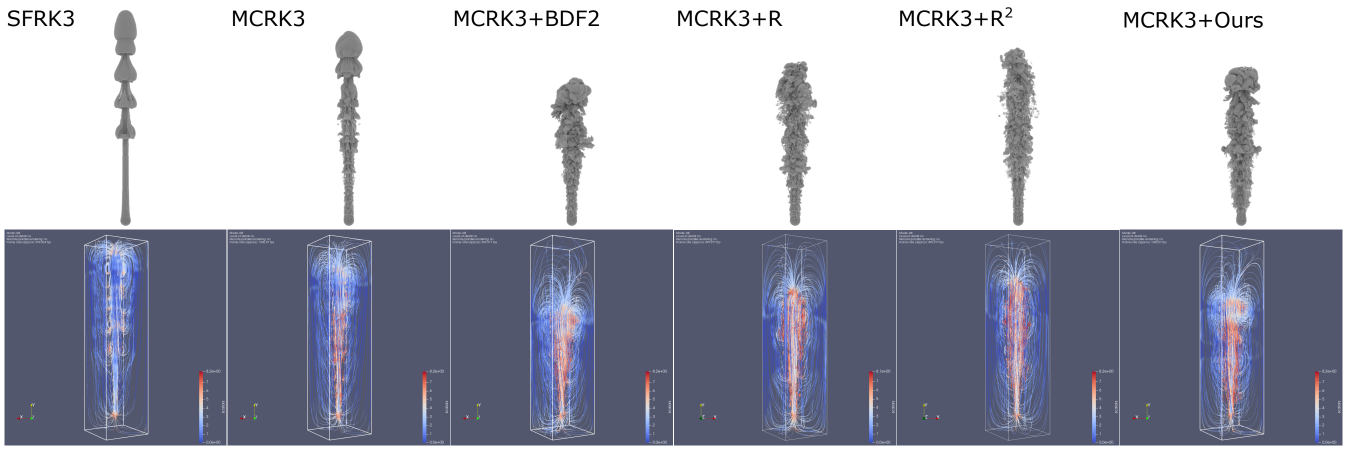

Figure 11: *Smoke plumes scenario with different solver at t = 6.0 seconds, a hot smoke rising with buoyancy, visualized with density(top) and Shannon's entropy(down).*

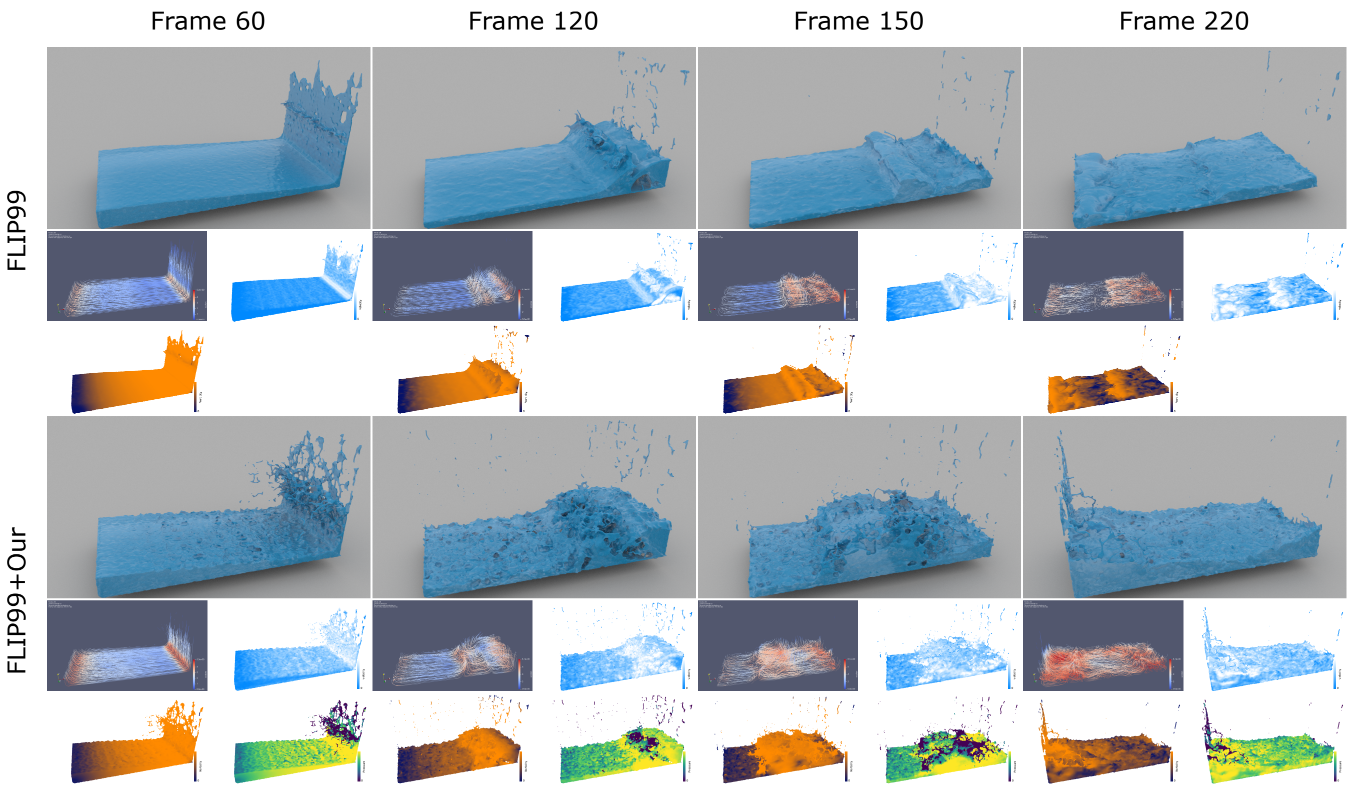

Figure 12: *Dam Break scenario with different solver at frame 60, 120, 150, 220, visualized with realistic rendering, streamline, fluid velocity, fluid vorticity and fluid pressure.*

some vortex motion at the edge and gradually merge to produce interesting phenomena. The experimental results are shown in the Fig. 10. The R solver, R^2 solver and our solver have better saved the vorticity and achieved satisfactory results.

4.3. 3D Results

Smoke Plumes. In order to verify the effectiveness and usability of the algorithm, we designed a 3D smoke plumes scene. We set up a spherical smoke emitter which emits hot smoke into the scene. We give the smoke entering the scene a buoyancy acceleration, and the smoke will rise rapidly over time and produce complex and interesting fluid phenomena.

The experimental results are shown in Fig. 11. We can see that the traditional algorithms (SF, MC) produce much less details than the second-order methods(MC+BDF2, MC+R, MC+ R^2, MC+Ours). And our method can capture the most vortical structure. In order to make a better comparative observation, we visualized the streamline of the scene, and Shannon's entropy is used for coloring. The results are shown in the second row of Fig. 11.

Dam Break. We initialize a hexahedral initial water area on the left of the scene and let it develop freely under the action of gravity and solid wall. When the wave head hits another wall, it will produce an interesting wave like effect. The experimental results are shown in Fig. 12(in order to better simulate the free surface fluid, we use FLIP99[ZB05] as the advection scheme of current scene).

In order to better show the advantages of our method, we visualize the scene in different ways in Fig. 12: realistic rendering, streamline, fluid velocity, fluid vorticity and fluid pressure. The results demonstrate that first-order advection-projection framework produces serious numerical viscosity, and our method can better capture and retain the energy and details of the fluid.

4.4. Numerical Results

Taylor-Green Vortex. In order to verify that our method has second-order accuracy in time, we choose Taylor-Green Vortex which has analytical solution. Similar to many other papers, we ran it to 1s, and computed the RMS error in velocity with respect to the analytical solution. Fig. 13 shows the log-log plot of this error as a function of time. For more intuitive comparison, we visualize the experimental data when $\Delta t = 0.1$ (MC+R=0.2), as shown in Fig. 14.

The experimental results show that BDF2, R solver and our method all have second-order time accuracy, which is consistent with the theory. In terms of error scale, our method is slightly smaller than BDF2, and slightly larger than the reflection solver (the two error sources are described in Sec. 3.3 and Sec. 3.4. To sum up, firstly, we ignore some changes when explicitly advancing the pressure gradient, and secondly, the inaccurate advection operation on pressure and pressure gradient introduce another error). However, compared with the R solver, we save half the time overhead, and BDF2 has wrong and asymmetric calculation results in many cases, so we think our method will have great application potential.

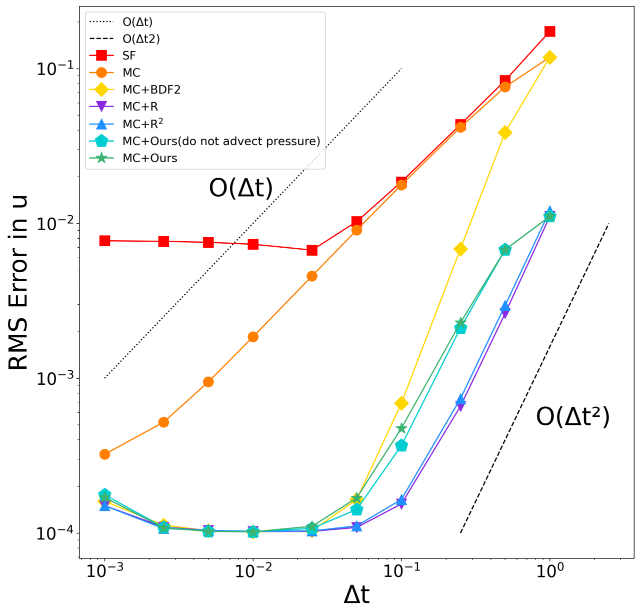

Figure 13: *The log-log plot of RMS error in velocity as a function of time.*

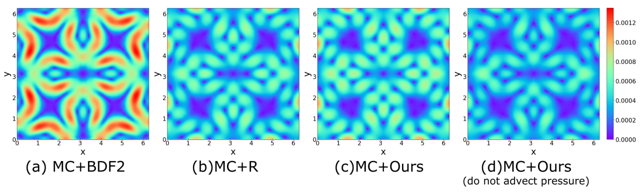

Figure 14: *The visualization plot of RMS error in velocity.*

5. Conclusions

By modifying the reflection solver with the idea of explicit midpoint method, we get a new algorithm that does not lose its accuracy too much and reduces the computational overhead to half or even more.

In addition, our scheme does not modify the advection term and projection process in the original framework, so can be seamlessly integrated into any existing advection-projection fluid simulation program. Our scheme is also completely parallel. We have designed a large number of 2D and 3D effects and numerical experiments, and compared some mainstream methods to verify the accuracy of our algorithm. The results show that our algorithm greatly improves the time accuracy of advection-projection method under the premise of small space-time overhead, and produces convincing results.

5.1. Limitations

Even if we consider the influence of pressure gradient on fluid motion with second-order time accuracy, the error of advection operation still exists, and because we need to explicitly advect the pressure and its gradient, the selection of advection algorithm is more important. In order to achieve better visual effect in more general scenes, we suggest to use the computational power saved by our algorithm for high-order advection schemes, to get the best compromise between performance and effect.

5.2. Future Work

There are still many interesting works about our algorithm.

Boundary Conditions. As mentioned in Sec. 3.5, the introduction of explicit pressure makes us have to consider additional pressure boundary conditions. In this paper, we simply specify it without in-depth research. We think it may be one of the potential research directions.

Combination with other methods. Our method only makes a simple modification to the original advection-projection method, and therefore we can enjoy its benefits. For example, we can combine some interesting schemes (artificially injected vorticity) with our method to obtain more interesting simulation results.

Improvement of advection term. As mentioned in the article, we only make a second-order correction to the pressure gradient term in the Navier-Stokes equation, and the choice of advection term is arbitrary. The combination of a BDF2 type advection with our method may have a potential accuracy improvement.

6. Acknowledgement

This work was supported by the National Key Project of China (Project Number GJXM92579) and National Natural Science Foundation of China (Project Number 61972271).

References

[Ang17] ANGELIDIS, ALEXIS. "Multi-scale vorticle fluids". *ACM Transactions on Graphics (TOG)* 36.4 (2017), 1–12 2.

[BCG87] BELL, JOHN, COLELLA, PHILLIP, and GLAZ, HARLAND. "A second-order projection method for viscous, incompressible flow". *8th Computational Fluid Dynamics Conference*. 1987, 1176 5.

[BCG89] BELL, JOHN B, COLELLA, PHILLIP, and GLAZ, HARLAND M. "A second-order projection method for the incompressible Navier-Stokes equations". *Journal of computational physics* 85.2 (1989), 257–283 5.

[BCM01] BROWN, DAVID L, CORTEZ, RICARDO, and MINION, MICHAEL L. "Accurate projection methods for the incompressible Navier–Stokes equations". *Journal of computational physics* 168.2 (2001), 464–499 5.

[Bri15] BRIDSON, ROBERT. *Fluid simulation for computer graphics*. AK Peters/CRC Press, 2015 2, 5.

[Cho68] CHORIN, ALEXANDRE JOEL. "Numerical solution of the Navier-Stokes equations". *Mathematics of computation* 22.104 (1968), 745–762 2.

[CMM90] CHORIN, ALEXANDRE JOEL, MARSDEN, JERROLD E, and MARSDEN, JERROLD E. *A mathematical introduction to fluid mechanics*. Vol. 3. Springer, 1990 2.

[DL03] DUPONT, TODD F and LIU, YINGJIE. "Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function". *Journal of Computational Physics* 190.1 (2003), 311–324 2.

[ETK*07] ELCOTT, SHARIF, TONG, YIYING, KANSO, EVA, et al. "Stable, circulation-preserving, simplicial fluids". *ACM Transactions on Graphics (TOG)* 26.1 (2007), 4–es 2, 3.

[FGG*17] FU, CHUYUAN, GUO, QI, GAST, THEODORE, et al. "A polynomial particle-in-cell method". *ACM Transactions on Graphics (TOG)* 36.6 (2017), 1–12 2.

[FM97] FOSTER, NICK and METAXAS, DIMITRIS. "Modeling the motion of a hot, turbulent gas". *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 1997, 181–188 2.

[FSJ01] FEDKIW, RONALD, STAM, JOS, and JENSEN, HENRIK WANN. "Visual simulation of smoke". *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001, 15–22 2.

[HW65] HARLOW, FRANCIS H and WELCH, J EDDIE. "Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface". *The physics of fluids* 8.12 (1965), 2182–2189 2.

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

[KLLR05] KIM, BYUNGMOON, LIU, YINGJIE, LLAMAS, IGNACIO, and ROSSIGNAC, JAROSLAW R. *Flowfixer: Using bfecc for fluid simulation*. Tech. rep. Georgia Institute of Technology, 2005 2.

[McK07] MCKENZIE, ALEXANDER GEORGE. "HOLA: a high-order Lie advection of discrete differential forms with applications in Fluid Dynamics". PhD thesis. California Institute of Technology, 2007 6.

[MCP*09] MULLEN, PATRICK, CRANE, KEENAN, PAVLOV, DMITRY, et al. "Energy-preserving integrators for fluid animation". *ACM Transactions on Graphics (TOG)* 28.3 (2009), 1–8 2.

[NZT19] NARAIN, RAHUL, ZEHNDER, JONAS, and THOMASZEWSKI, BERNHARD. "A Second-order advection-reflection solver". *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 2.2 (2019), 1–14 2, 3, 5.

[PCK*19] PADILLA, MARCEL, CHERN, ALBERT, KNÖPPEL, FELIX, et al. "On bubble rings and ink chandeliers". *ACM Transactions on Graphics (TOG)* 38.4 (2019), 1–14 2.

[PK05] PARK, SANG IL and KIM, MYOUNG JUN. "Vortex fluid for gaseous phenomena". *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 2005, 261–270 2.

[QZG*19] QU, ZIYIN, ZHANG, XINXIN, GAO, MING, et al. "Efficient and conservative fluids using bidirectional mapping". *ACM Transactions on Graphics (TOG)* 38.4 (2019), 1–12 2, 5–7.

[SC91] STANIFORTH, ANDREW and CÔTÉ, JEAN. "Semi-Lagrangian integration schemes for atmospheric models—A review". *Monthly weather review* 119.9 (1991), 2206–2223 2.

[SFK*08] SELLE, ANDREW, FEDKIW, RONALD, KIM, BYUNGMOON, et al. "An unconditionally stable MacCormack method". *Journal of Scientific Computing* 35.2 (2008), 350–371 2, 5.

[SRF05] SELLE, ANDREW, RASMUSSEN, NICK, and FEDKIW, RONALD. "A vortex particle method for smoke, water and explosions". *ACM SIGGRAPH 2005 Papers*. 2005, 910–914 2.

[Sta99] STAM, JOS. "Stable fluids". *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 1999, 121–128 2, 3, 5.

[SU94] STEINHOFF, JOHN and UNDERHILL, DAVID. "Modification of the Euler equations for "vorticity confinement": Application to the computation of interacting vortex rings". *Physics of Fluids* 6.8 (1994), 2738–2744 2.

[YXZ*21] YANG, SHUQI, XIONG, SHIYING, ZHANG, YAORUI, et al. "Clebsch gauge fluid". *ACM Transactions on Graphics (TOG)* 40.4 (2021), 1–11 2.

[ZB05] ZHU, YONGNING and BRIDSON, ROBERT. "Animating sand as a fluid". *ACM Transactions on Graphics (TOG)* 24.3 (2005), 965–972 2, 9.

[ZB14] ZHANG, XINXIN and BRIDSON, ROBERT. "A PPPM fast summation method for fluids and beyond". *ACM Transactions on Graphics (TOG)* 33.6 (2014), 1–11 2.

[ZBG15] ZHANG, XINXIN, BRIDSON, ROBERT, and GREIF, CHEN. "Restoring the missing vorticity in advection-projection fluid solvers". *ACM Transactions on Graphics (TOG)* 34.4 (2015), 1–8 2, 3.

[ZNT18] ZEHNDER, JONAS, NARAIN, RAHUL, and THOMASZEWSKI, BERNHARD. "An advection-reflection solver for detail-preserving fluid simulation". *ACM Transactions on Graphics (TOG)* 37.4 (2018), 1–8 2, 3, 5, 6.