Affine particle in cell method for MAC grids and fluid simulation

Ounan Ding, Tamar Shinar, Craig Schroeder[1]

Department of Computer Science and Engineering
University of California Riverside

Abstract

We present a new technique for transferring momentum and velocity between particles and MAC grids based on the Affine-Particle-In-Cell (APIC) framework [1, 2] previously developed for co-located grids. APIC represents particle velocities as locally affine, rather than locally constant as in traditional PIC. These previous APIC schemes were designed primarily as an improvement on Particle-in-Cell (PIC) transfers, which tend to be heavily dissipative, and as an alternative to Fluid Implicit Particle (FLIP) transfers, which tend to be noisy.

The original APIC paper [1] proposed APIC-style transfers for MAC grids, based on a limit for multilinear interpolation. We extend these to the case of smooth basis functions and show that the proposed transfers satisfy all of the original APIC properties. In particular, we achieve conservation of angular momentum across our transfers, something which eluded [1].

Early indications in [1] suggested that APIC might be suitable for simulating high Reynolds fluids due to favorable retention of vortices, but these properties were not studied further and the question was left unresolved. One significant drawback of APIC relative to FLIP is that energy is dissipated even when $\Delta t = 0$. We use two dimensional Fourier analysis to investigate dissipation in this important limit. We investigate dissipation and vortex retention numerically in the general case to quantify the effectiveness of APIC compared with PIC, FLIP, and XPIC.

Keywords: APIC, PIC, FLIP, MPM, MAC grids, hybrid Lagrangian/Eulerian, particle-grid, computational fluid dynamics

1. Introduction

Hybrid particle/grid methods have been used for decades to simulate many different physical phenomena, including compressible flow, incompressible flow, plasma physics, computational solids, granular materials, and many more [3]. The original hybrid scheme was Particle In Cell (PIC) [4], which was originally devised for fluids. PIC worked by mapping particle state to a fixed Eulerian grid, on which forces are computed.

[1] craigs@cs.ucr.edu

The updated grid state is then mapped back to particles. In the original method, these mapping steps were done using linear interpolation and nearest-point interpolation. These low-order interpolation strategies were critical to the original method, since using smoother interpolation for both transfers produces excessive dissipation. However, smoother interpolation strategies are important for eliminating cell-crossing instabilities [5] and avoiding discontinuities in the flow map derivatives [6]. As a result, the original PIC method was not widely adopted.

A major improvement came with the introduction of Fluid Implicit Particle (FLIP) [7, 8], which mapped *changes* in velocities from the grid to the particles. This broke the cycle of repeated velocity interpolation, allowing smoother interpolation kernels to be used while avoiding excessive dissipation. This has led to a number of new interpolation kernels, including GIMP [5, 9], CPDI [10, 11]. B-spline interpolation has also been shown to work well [6].

This also greatly improved the angular momentum conservation properties of the particle/grid transfers [12, 13]. Indeed, FLIP transfers can be used to implement schemes that conserve momentum, angular momentum, and total energy [14, 15]. Another major advance in hybrid methods came with the introduction of the Material Point Method (MPM), which extended hybrid methods to handle viscoelastic solids [16, 17].

Although most hybrid methods today are based on FLIP transfers, such schemes are known to suffer from noise caused by numerical instabilities. While all hybrid particle-grid approaches suffer to some degree from the finite grid instability [18, 19] (or the ringing instability [20, 21]), these errors are quite prominent when FLIP transfers are used. This is particularly true when using MPM [16, 17] for simulating history dependent materials. The finite grid instability may be understood as a mismatch between the modes that can be represented on particles and the modes that can be represented on the mesh.

Some explanation and intuition for the causes of these instabilities, as well as ideas for reducing them, may also be gleaned by examining the way transfers interact with grid forces. In a PIC-like scheme, particle velocities are transferred (interpolated) to the grid, then transferred (interpolated) back to particles at the end of the time step. The effect of this repeated interpolation is significant dissipation. If noise is added to the particle velocities, the noise is filtered out during the interpolations and also to some degree by the physics and grid-based numerical scheme. This makes PIC-like schemes very stable. FLIP transfers back velocity differences instead, which avoids the dissipation. If no changes are made to the grid velocities, then the particle velocities are unmodified. However, this also makes the scheme respond differently to particle noise. If the particle-to-grid transfer operator has a nullspace, then any noise in the nullspace would not be transferred to the grid. Since this noise component will not be damped on the grid, no corresponding correction will be transferred back to the particles. The noise component is not damped. Typically there are many more particles than grid nodes, which means the transfer operator must have a large nullspace in which particle noise may persist without damping. Failing to efficiently damp errors introduced during

	PIC	FLIP	APIC
1	$m_i^n = \sum_p w_{ip}^n m_p$	$m_i^n = \sum_p w_{ip}^n m_p$	$m_i^n = \sum_p w_{ip}^n m_p$
2a			$\mathbf{D}_p^n = \sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n)(\mathbf{x}_i^n - \mathbf{x}_p^n)^T$
2b			$\mathbf{C}_p^n = \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1}$
2c	$m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p \mathbf{v}_p^n$	$m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p \mathbf{v}_p^n$	$m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p (\mathbf{v}_p^n + \mathbf{C}_p^n (\mathbf{x}_i^n - \mathbf{x}_p^n))$
3	Grid evolution: $\mathbf{v}_i^n \to \tilde{\mathbf{v}}_i^{n+1}$	Grid evolution: $\mathbf{v}_i^n \to \tilde{\mathbf{v}}_i^{n+1}$	Grid evolution: $\mathbf{v}_i^n \to \tilde{\mathbf{v}}_i^{n+1}$
4a	$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1}$	$\mathbf{v}_p^{n+1} = \mathbf{v}_p^n + \sum_i w_{ip}^n (\tilde{\mathbf{v}}_i^{n+1} - \mathbf{v}_i^n)$	$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1}$
4b			$\mathbf{B}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n)^T$
5	$\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + \Delta t \tilde{\mathbf{v}}_i^{n+1}$	$\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + \Delta t \tilde{\mathbf{v}}_i^{n+1}$	$\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + \Delta t \tilde{\mathbf{v}}_i^{n+1}$
6	$\mathbf{x}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1}$	$\mathbf{x}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1}$	$\mathbf{x}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1}$

Figure 1: Representative time integration schemes for PIC, FLIP, and APIC. Details of the grid evolution are identical for each. Note that the transfers are very similar, though APIC includes a few extra steps related to the additional particle state. Subscripts indicate where quantities like ($\mathbf{v}_i$ is velocity on the grid; $\mathbf{v}_p$ is velocity on particles).

integration leads to numerical instability.

When using FLIP transfers, the particle velocities are not used to move the particle positions; rather, particle positions are directly interpolated from the grid. This is equivalent to using an interpolated, PIC velocity for position updates. This greatly limits the negative impact of the spurious particle velocities. This also means it is possible for a simulation to come to rest with nonzero velocities (See Figure 19). As long as the final velocity field is in the transfer operator's nullspace, the grid velocity will be zero and the particles will not move. Despite these issues, FLIP transfers are still most commonly used, particularly for MPM. Blends between PIC and FLIP transfers are also a viable alternative [22, 23, 24], using a small amount of the PIC solution to dissipate noise that might otherwise accumulate in the FLIP solution.

Recently, a new transfer called Affine Particle In Cell (APIC) was developed as a PIC-like alternative to FLIP [1, 2]. These transfers interpolate information from particles to grid and also from grid to particles, as in the original PIC (See Figure 1). To reduce dissipation, the rowspace of the transfer operator is enriched by storing velocities and a measure of velocity gradients on particles. (It is worth pointing out that [25] did something similar in the context of FLIP transfers, but the benefit in this case was noticeable but relatively modest. FLIP is already not very dissipative, and the modifications to the transfers do not reduce the finite grid instability.) In doing so, fewer velocity modes are filtered out by the transfers, dramatically reducing dissipation. In particular, it is possible to develop APIC schemes that conserve both linear and angular momentum in the case of co-located grids [2].

	XPIC(r)	XPIC(1)
1	$m_i^n = \sum_p w_{ip}^n m_p$	$m_i^n = \sum_p w_{ip}^n m_p$
2	$m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p \mathbf{v}_p^n$	$m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p \mathbf{v}_p^n$
3	Grid evolution: $\mathbf{v}_i^n \to \tilde{\mathbf{v}}_i^{n+1}$	Grid evolution: $\mathbf{v}_i^n \to \tilde{\mathbf{v}}_i^{n+1}$
4a	$\mathbf{v}_i^{n,1} = r\mathbf{v}_i^n$	
4b	$\mathbf{v}_i^{n,k} = \frac{r-k+1}{k} \sum_p \sum_j \frac{m_p w_{ip}^n w_{jp}^n}{m_i^n} \mathbf{v}_j^{n,k-1}$	
4c	$\mathbf{v}_i^{n*} = \sum_{k=1}^{r} (-1)^{k+1} \mathbf{v}_i^{n,k}$	
4d	$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n (\mathbf{v}_i^{n*} + \tilde{\mathbf{v}}_i^{n+1} - \mathbf{v}_i^n)$	$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1}$
5	$\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + (\mathbf{v}_i^n + \tilde{\mathbf{v}}_i^{n+1}) \frac{\Delta t}{2}$	$\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + (\mathbf{v}_i^n + \tilde{\mathbf{v}}_i^{n+1}) \frac{\Delta t}{2}$
6	$\mathbf{x}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1} + \sum_i w_{ip}^n (\mathbf{v}_i^{n*} - \mathbf{v}_p^n) \frac{\Delta t}{2}$	$\mathbf{x}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1} + \sum_i w_{ip}^n (\mathbf{v}_i^n - \mathbf{v}_p^n) \frac{\Delta t}{2}$

Figure 2: Time integration scheme for XPIC of order r. XPIC(1) is equivalent to PIC except for the more accurate grid and particle position update. Step 4b is repeated for $2 \le k \le r$.

Since the development of APIC, another scheme called XPIC [26] was developed as another compromise between the dissipation of PIC and the noise of FLIP. XPIC is a family of schemes that is in many ways a blend of PIC and FLIP (and includes PIC as a member). XPIC uses PIC transfers to filter out noise from the FLIP solution, unlike a simple blend which merely damps it. XPIC significantly reduces FLIP-style noise, but unlike APIC does not eliminate it. In the special case of XPIC(1), the transfer of velocity from grid to particle is equivalent to PIC method, but XPIC(1) differs in the position update (See Figure 2). XPIC(1) may be considered as an improved PIC. Other regularization strategies have also been employed to mitigate the noise caused by FLIP [27].

Another transfer strategy is moving least squares (MLS) [27], which computes a polynomial best fit to transfer velocity information to particles. MLS is capable of high order accuracy but is rather expensive due to the need to solve a system of equations per particle for the transfers. Indeed, some variants of APIC may be formulated as a PIC-style MLS with polynomial degree one [28, 29].

Although the focus on APIC transfers has mostly been in the context of MPM, [1] also explored a limited extension to MAC grids. *In this work, we show that an APIC scheme can be developed for MAC grids that also conserves both linear and angular momentum.*

Being a new method, there are many questions about the behavior and utility of APIC that have not been explored. In the context of fluids, one important concern is the suitability of PIC for high Reynolds number flows. *Shedding light on this question in the primary goal of this work.*

We show how a two dimensional Fourier transform can be used to study the dissipation of transfers for MAC-grid-based incompressible fluids, in a similar way to how one dimensional Fourier transforms are currently being used for one dimensional MPM. We also show that two dimensional Fourier transform can be used to study the interaction between transfers and pressure projection. This makes it possible to compare APIC, PIC, FLIP and XPIC in terms of dissipation of two dimensional incompressible flows.

2. Numerical method

2.1. Notation

In this document, we use notation to give hints as to the meaning of symbols. As a general rule, bold lowercase symbols ($\mathbf{x}_p^n$, $\mathbf{p}^{P,n}$, $\mathbf{e}_a$) are vectors, bold uppercase symbols ($\mathbf{D}_{pa}^n$, $\mathbf{I}$) are matrices, and non-bold symbols (w_{ipa}^n, m_p, Δt, $\tilde{v}_{ia}^{n+1}$) are scalars. We follow the convention that all vector quantities are considered to be column vectors unless explicitly transposed. Thus, quantities like ∇p will be treated as column vectors.

Many symbols use a combination of subscripts and superscripts. Subscripts are used to index grid nodes (i, j), particles (p), and spatial dimensions (a, b). We index MAC faces by treating each axis direction as a regular grid, which is indexed with ia. The axis direction is denoted as $\mathbf{e}_a$. Quantities associated with both grid and particle indices have both indices (w_{ipa}^n). A superscript of n indicates a quantity near the beginning of the time step (before forces are applied), and a superscript of $n+1$ indicates a quantity computed later in the time step. Other adornments are used to distinguish quantities that would otherwise get the same name ($\tilde{\mathbf{x}}_{ia}^{n+1}$ vs $\mathbf{x}_{ia}^{n+1}$) or to denote intermediates (v_{ia}^*). The superscripts P and G indicate global particle-based or grid-based quantities ($\mathbf{l}^{P,n}, \mathbf{l}^{G,n}$). To avoid confusion, we will never use the summation convention in this document; all summation is specified explicitly.

Note that the indices can be used to unambiguously distinguish quantities on MAC grids (w_{ipa}^n) from those on co-located grids (w_{ip}^n). Such quantities have the same meaning and differ only in the grid layout chosen.

2.2. Weights

Hybrid schemes are notable for requiring information to be transferred between particles and a grid. These transfers are defined using an interpolating kernel, which is assumed to satisfy the partition of unity and interpolation properties

$$\sum_i N(\mathbf{x}_p^n - \mathbf{x}_i^n) = 1 \qquad \sum_i \mathbf{x}_i^n N(\mathbf{x} - \mathbf{x}_i^n) = \mathbf{x} \tag{1}$$

for any $\mathbf{x}$. The kernel $N(\mathbf{x})$ is used to define interpolation weights and weight gradients as $w_{ip}^n = N(\mathbf{x}_p^n - \mathbf{x}_i^n)$ and $\nabla w_{ip}^n = \nabla N(\mathbf{x}_p^n - \mathbf{x}_i^n)$. The properties of $N(\mathbf{x})$ lead to properties for w_{ip}^n and ∇w_{ip}^n:

$$\sum_i w_{ip}^n = 1 \qquad \sum_i w_{ip}^n \mathbf{x}_i^n = \mathbf{x}_p^n \qquad \sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n) = \mathbf{0} \qquad \sum_i \mathbf{x}_i^n (\nabla w_{ip}^n)^T = \mathbf{I}. \tag{2}$$

In the case of MAC grids and the proposed time integration, weights are defined independently for each axis with $\bar{w}^n_{ipa} = N(\mathbf{x}^n_p - \mathbf{x}^n_{ia})$ and $w^n_{ipa} = N(\mathbf{x}^{n+1}_p - \mathbf{x}^n_{ia})$. In our discretization we advance positions *before* transferring particle information to the grid, and the algorithm needs two sets of weights when using FLIP or XPIC. We use $\bar{w}^n_{ipa}$ to denote weights before moving particles and w^n_{ipa} to denote weights after. It is not actually necessary to compute two sets of weights, since $w^n_{ipa} = \bar{w}^{n+1}_{ipa}$.

The x, y, and z faces form regular Cartesian grids that are offset from one another. The same properties hold independently per axis:

$$\sum_i w^n_{ipa} = 1 \qquad \sum_i w^n_{ipa}\mathbf{x}^n_{ia} = \mathbf{x}^{n+1}_p \qquad \sum_i w^n_{ipa}(\mathbf{x}^n_{ia} - \mathbf{x}^{n+1}_p) = \mathbf{0} \qquad \sum_i \mathbf{x}^n_{ia}(\nabla w^n_{ipa})^T = \mathbf{I}. \tag{3}$$

For completeness, the linear, quadratic, and cubic splines we investigate for the kernel $\hat{N}(x)$ are:

linear	quadratic	cubic
$\begin{cases} 1 - \lvert x\rvert & 0 \le \lvert x\rvert < 1 \\ 0 & 1 \le \lvert x\rvert \end{cases}$	$\begin{cases} \frac{3}{4} - x^2 & 0 \le \lvert x\rvert < \frac{1}{2} \\ \frac{1}{2}(\frac{3}{2} - \lvert x\rvert)^2 & \frac{1}{2} \le \lvert x\rvert < \frac{3}{2} \\ 0 & \frac{3}{2} \le \lvert x\rvert \end{cases}$	$\begin{cases} \frac{2}{3} - \frac{1}{2}x^2(2 - \lvert x\rvert) & 0 \le \lvert x\rvert < 1 \\ \frac{1}{6}(2 - \lvert x\rvert)^3 & 1 \le \lvert x\rvert < 2 \\ 0 & 2 \le \lvert x\rvert \end{cases}$

From this, $N(\mathbf{x}) = \hat{N}(x)\hat{N}(y)$ in 2D and $N(\mathbf{x}) = \hat{N}(x)\hat{N}(y)\hat{N}(z)$ in 3D.

2.3. Overview of co-located transfers

General outlines for a sample algorithm for PIC, FLIP, and APIC with a co-located grid are provided in Figure 1. A corresponding outline for XPIC is provided in Figure 2. In each case, the time step begins by transferring particle mass m_p and momentum $m_p\mathbf{v}^n_p$ to the grid to produce mass m^n_i and momentum $m^n_i\mathbf{v}^n_i$ on the grid (steps 1-2). Velocity is computed by dividing the mass from the momentum. This velocity is updated on the grid in some way, such as by applying MPM finite element forces (step 3). This results in an updated grid velocity $\tilde{\mathbf{v}}^{n+1}_i$, which is then transferred back to particles, resulting in the new particle velocity $\mathbf{v}^{n+1}_p$ (step 4). Finally, particles are updated to new locations $\mathbf{x}^{n+1}_p$ by interpolating them from moving grid positions $\tilde{\mathbf{x}}^{n+1}_i$ (steps 5-6). Additional steps may be required depending on the specifics of the grid evolution algorithm, such as maintaining a deformation gradient ($\mathbf{F}^n_p \to \mathbf{F}^{n+1}_p$).

This simple outline is flexible, and many variations have been considered. For example, steps 4-6 may be moved to the beginning of the time step; in this case, the transfers may be interpreted as a type of semi-Lagrangian advection for an Eulerian algorithm [30]. PIC and FLIP differ only in step 4a, and a PIC/FLIP blend may be constructed by interpolating these two velocity updates [22]. A more general form for steps 4b and 6 in APIC is considered in [2], which allows them to do midpoint rule for their time integration and achieve conservation of angular momentum. Viewed as an advection scheme, APIC may be compared to [31], which also stores derivative information to reduce diffusion.

2.4. APIC for MAC grids

We begin our treatment of the MAC case by laying out the full time integration scheme. Since we will be using our MAC grid for fluids, we discretized the Navier-Stokes equations. We implement a projection method, as is typically done with FLIP [22, 30].

2.4.1. Advection

In a standard standard MPM discretization, particle positions are updated at the end of the time step. If this is done with Chorin splitting, a pressure projection will be used to make the fluid velocity divergence-free, after which this velocity will be transferred to particles and moved. The resulting particle velocities are not divergence free, and we have observed convergence problems in the resulting scheme.

Instead, we note that [22] moves particles at the beginning of the time step. This is in line with Chorin splitting, which projects the advected velocity field, resulting in a divergence-free velocity field at the end of the time step. To implement this, we delay the position update until the beginning of the next time step but otherwise compute the update in exactly the same way.

$$\mathbf{x}_p^{n+1} = \sum_{ia} \bar{w}_{ipa}^n \mathbf{e}_a \mathbf{e}_a^T \tilde{\mathbf{x}}_{ia}^n = \mathbf{x}_p^n + \Delta t \mathbf{v}_p^n. \tag{4}$$

Here we use $\bar{w}_{ipa}^n = N(\mathbf{x}_p^n - \mathbf{x}_{ia}^n)$ to denote the weights *before* moving particles, reserving $w_{ipa}^n = N(\mathbf{x}_p^{n+1} - \mathbf{x}_{ia}^n)$ for the weights *after* moving particles. $\mathbf{e}_a$ are axis directions. Note that $\tilde{\mathbf{x}}_{ia}^n$ is the quantity $\tilde{\mathbf{x}}_{ia}^{n+1}$ from the previous time step. In the case of APIC (or PIC), the second equality holds, and the particles can be moved using particle velocities without referencing $\bar{w}_{ipa}^n$. Although the XPIC position update is more accurate than the PIC update, its use would prevent our APIC transfers from satisfying the APIC properties. Generalized APIC transfers [2] could likely be adapted to MAC grids to achieve similar benefits. Since these are more complex than the original transfers, we base our transfers on the original APIC transfers and use PIC position updates.

In the case of FLIP and XPIC, we store $\tilde{v}_{ia}^{n+1}$ from the previous time step, which allows us to compute new positions using the summation. Since this information in not available for the first frame, we perform a particle-to-grid transfer to obtain initial velocities for $\tilde{v}_{ia}^{n+1}$. The PIC/APIC update could be used for the first time step instead.

2.4.2. Particle to grid

The next step is transferring particle mass m_p, velocity $\mathbf{v}_p^n$, and information about velocity derivatives $\mathbf{b}_{pa}^n$ from particles to the grid.

$$m_{ia}^n = \sum_p w_{ipa}^n m_p \tag{5}$$

$$\mathbf{D}_{pa}^n = \sum_i w_{ipa}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})(\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})^T \tag{6}$$

$$m_{ia}^n v_{ia}^n = \sum_p w_{ipa}^n m_p \mathbf{e}_a^T \mathbf{v}_p^n + \sum_p w_{ipa}^n m_p (\mathbf{b}_{pa}^n)^T (\mathbf{D}_{pa}^n)^{-1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \tag{7}$$

This looks very similar to the co-located case, but there are some subtle differences. There is one matrix $\mathbf{D}_{pa}^n$ defined per axis, and $\mathbf{b}_{pa}^n$ is a vector per axis. These vectors may be interpreted as the columns of the matrix $\mathbf{B}_p^n$ that is used in the co-located case. Being a MAC layout, velocities are staggered, and the scalar components of velocity v_{ia}^n are stored at separate locations. Recall that w_{ipa}^n is computed using the new particle positions, which explains the unexpected use of $\mathbf{x}_p^{n+1}$ in these equations.

2.4.3. Grid evolution

We split the grid evolution step into two parts: gravity and pressure. Gravity is applied explicitly: $v_{ia}^* = v_{ia}^n + \Delta t g_a$. We use finite differences to discretize the Poisson equation and perform a velocity projection on a MAC grid layout to obtain an incompressible velocity field $v_{ia}^* \to \tilde{\mathbf{v}}_{ia}^{n+1}$. We assume a constant density ρ for the pressure discretization. These steps are performed in exactly the same way as for an Eulerian MAC discretization. We assume inviscid Euler, so we do not apply viscosity. Although we have masses on the grid, we do not use them for the pressure projection, since doing so causes *boiling* in the fluid. That is, an initially stationary pool of water would develop currents in it.

2.5. Grid to particle

After updating grid velocities, we update our final particle data following essentially the same algorithm as in the co-located case.

$$\mathbf{v}_p^{n+1} = \sum_{ia} w_{ipa}^n \tilde{v}_{ia}^{n+1} \mathbf{e}_a \tag{8}$$

$$\mathbf{b}_{pa}^{n+1} = \sum_i w_{ipa}^n \tilde{v}_{ia}^{n+1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \tag{9}$$

$$\tilde{\mathbf{x}}_{ia}^{n+1} = \mathbf{x}_{ia}^n + \Delta t \tilde{v}_{ia}^{n+1} \mathbf{e}_a \tag{10}$$

Unlike the co-located algorithm, we do not advance positions; we delay this step until the beginning of the next time step. In Section 7 we show that these transfers satisfy the original APIC properties.

3. Fourier analysis of transfers

We use Fourier analysis to characterize the dissipation of APIC transfers compared to PIC, FLIP, and XPIC transfers. There are two main settings in which this can be explored. The first is by considering a round trip from grid to particle and then back to grid. In between these transfers, particles are moved, and this step must be ignored to retain linearity. This approach is relatively simple, but it is unable to analyze methods like FLIP, which retain information on particles between time steps. The second approach is to consider transfers from particles to grid and then back to particles. In this case, other grid steps (pressure projection in our case) lie between these two transfers. Since these steps are in general linear, they can be included in the analysis. This analysis approach is compatible with FLIP and XPIC, and we use it to draw a strong contrast between FLIP and APIC.

To facilitate Fourier analysis, we must add two assumptions: (a) the domain is periodic and (b) all cells have the same particle distribution. Note that (b) does *not* imply that the particle distribution is regular. Particles may be positioned quite irregularly within a cell, but that irregular layout must be the same for all cells. Because of (a), it is convenient to treat grid indices as periodic.

3.1. Grid to particle to grid

Linear transfer matrix. While the transfers themselves are linear (as functions of velocities), the advection step is nonlinear due to the movement of the particles and corresponding changes in interpolation weights. For the purposes of analysis, we can eliminate the nonlinearity by considering the limit $\Delta t \to 0$. This corresponds to transferring from grid to particles and immediately back to the grid and approximates the dissipation that results when small time steps are taken. We can then express the grid-to-particle-to-grid transfer as a matrix $v_{ia}^{n+1} = \sum_{jb} M_{ia,jb} \tilde{v}_{jb}^{n+1}$.

Axis-independence. Inspecting the definition of the transfers carefully, we see that the axis components decouple. We can instead write $v_{ia}^{n+1} = \sum_j M_{ij}^a \tilde{v}_{ja}^{n+1}$, where $M_{ia,jb} = M_{ij}^a \delta_{ab}$. That is, each face axis has its own separate transfer matrix, but faces in different direction do not affect each other. This is not surprising; an object translating in the x direction should not begin moving in the y direction. Since each dimension is independent, we can focus on one axis arbitrarily and drop the axis indices. The grid layout is now just a regular grid. As such, the analysis in this section applies equally to transfers with a MAC layout and transfers with a collocated layout. We can now write $v_i^{n+1} = \sum_j M_{ij} \tilde{v}_j^{n+1}$, where M_{ij} is a matrix with as many rows and columns as grid nodes.

Analyzing dissipation with eigenvalues. When we analyze the dissipation of the transfers, it is helpful to isolate the dissipation caused by transfers from the dissipation caused by other parts of the evolution, such as the pressure projection. For this reason, we consider the consequences of repeating these transfers.

The eigenvalues λ of M_{ij} tell us how dissipative the transfers are. Eigenvectors corresponding to $\lambda = 1$ are preserved across the transfer without dissipation. Velocity eigenvectors with $0 < \lambda < 1$ decay due to dissipation. Eigenvectors with $\lambda = 0$ are eliminated entirely. $|\lambda| > 1$ would indicate instability. Within this periodic setup, we observe that M_{ij} is symmetric (each cell affects its left neighbor by the same amount as its right neighbor), which leads to real eigenvalues. In practice, we also observe that $0 \leq \lambda \leq 1$ for all of the schemes we consider. $\lambda \approx 1$ is ideal. A velocity mode corresponding an eigenvector with eigenvalue λ decays by factor λ^k, where the exponent k is the number of grid-to-particle-to-grid transfers.

3.1.1. Fourier analysis

Circulant. With the periodicity assumptions, the transfer matrix M_{ij} will be *tensor product* circulant. That is, if $i = (r, s)$ and $j = (u, v)$ are grid indices in 2D, then $M_{(r,s),(u,v)} = M_{(r+k,s+m),(u+k,v+m)}$ for any k and m, where we make use of the convention of treating the indices as periodic. This just says that the transfer operator appears the same for all cells, which is expected since all cells are indistinguishable. From this, we conclude that M_{ij} has the special structure $M_{(r,s),(u,v)} = c_{r-u,s-v}$, where c_i is the 0^{th} column of M_{ij}. This is the multidimensional analog of a circulant matrix.

Eigenvalues from Fourier transform. A tensor product circulant matrix is diagonalized by a multidimensional Fourier transform in exactly the same way that a circulant matrix is diagonalized by a Fourier transform. The eigenvalues of M_{ij} can be computed as the multidimensional Fourier transform of its column c_i. The Fourier transform conveniently identifies one eigenvalue λ_i with each grid node i, which provides a convenient visual representation of the eigenvalues in 2D. This is the basis for Figure 3. Fourier analysis has been used to analyze hybrid schemes before in 1D (e.g., in [26]), but as far as we are aware we are the first to adopt it as a tool in higher dimensions. This generalization is important, since incompressible flow is trivial in 1D and vortices do not exist in 1D.

Computing c_i. While it is possible to work out c_i and M_{ij} analytically (and we have done so in a few cases), the results are not enlightening. Instead, we compute c_i numerically by performing the transfers on a velocity field containing a single nonzero entry. The multidimensional Fourier transform of the result gives us the eigenvalues.

3.1.2. Eigenvalue images

Sparsity of c_i. The next useful observation is that c_i is very sparse; the nonzero entries in c_i are at most a few cells away from the nonzero entry in the velocity field. That is, $c_i = c_{rs} = 0$ for $|r| > w$ or $|s| > w$, where w here is the width of the stencil. Note that we are using $c_{rs} = c_{(r,s)}$ as a convenient shorthand. $w \leq 3$ for all of our splines (linear, quadratic, cubic) and transfers (APIC, FLIP, PIC, XPIC).

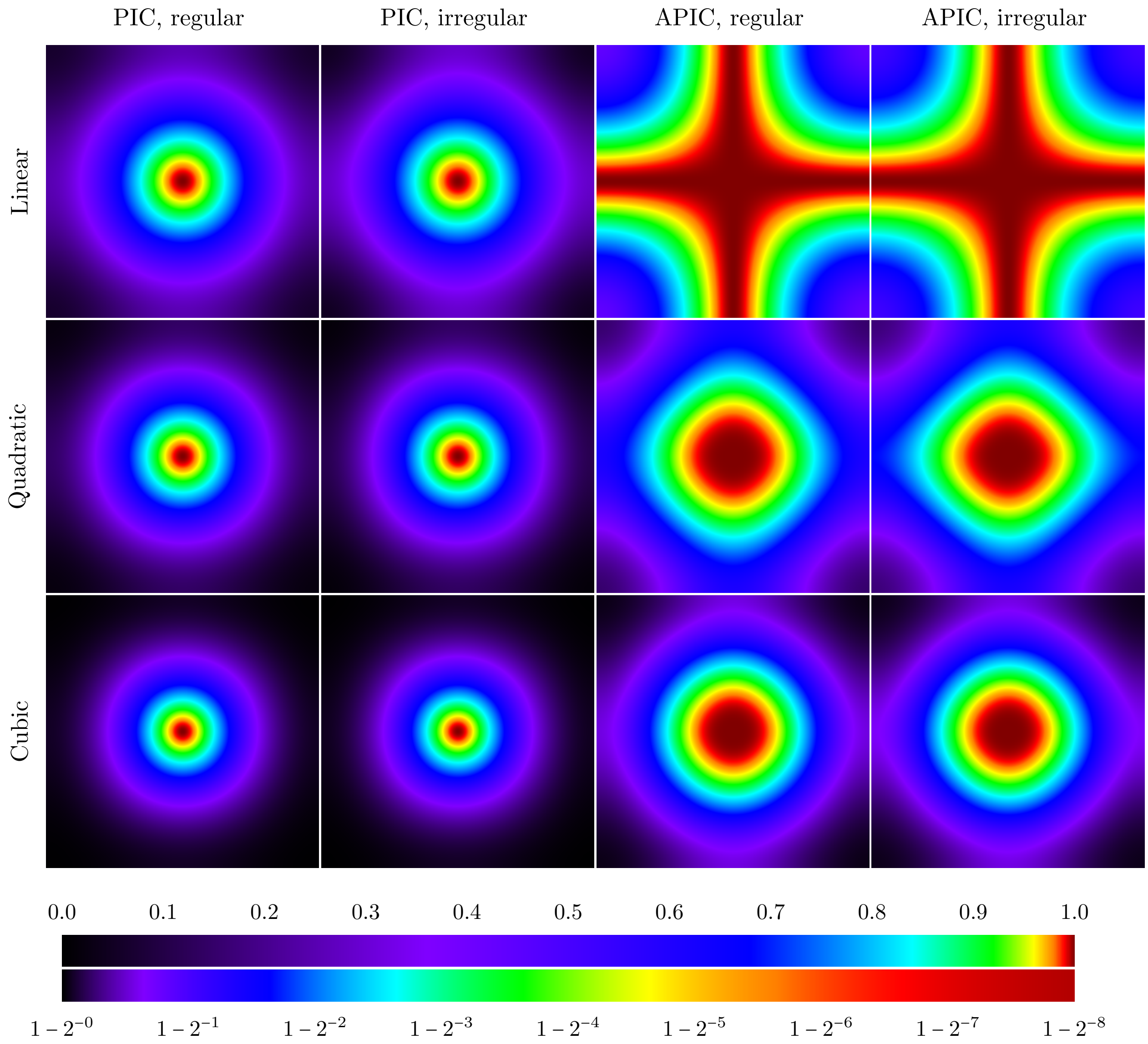
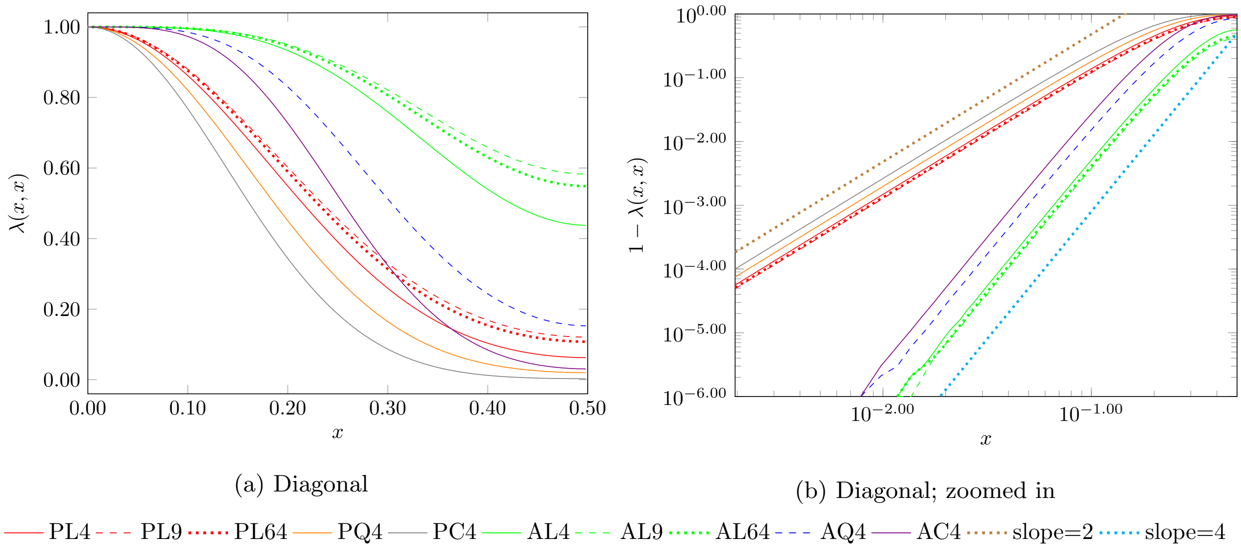


Figure 3: Eigenvalue images for PIC and APIC using 4 particles per cell and linear, quadratic, or cubic splines.

Resolution-independence of c_i. Next, we observe that the nonzero entries of c_i do not depend on the resolution provided the grid size $m \times n$ is large enough $(m, n \geq 2w+1)$. The transfers are local, so the nonzero entries of c_i cannot depend on the *number* of grid cells. Since Δx has units of length but the entries of c_i are dimensionless, the entries of c_i must be independent of Δx. This resolution-independence means that high-resolution images are cheap to compute, since they only require a moderate number of very low resolution transfers to deduce c_i followed by a full-resolution multidimensional Fourier transform.



(a) Diagonal

(b) Diagonal; zoomed in

Figure 4: Eigenvalues for PIC (P) and APIC (A), with 4, 9, or 64 particles per cell and linear (L), quadratic (Q), and cubic (C) interpolation splines. Only the 4 particles per cell case is shown for quadratic and cubic splines; the others are indistinguishable. The most important part of the plot is the top left near $(0,1)$, which corresponds to scale factor for larger scale vortices. For reference, $\lambda(0.10, 0.10)$ is the decay factor for a vortex 5 grid cells in diameter. $\lambda(0.05, 0.05)$ is the decay factor for a vortex 10 grid cells in diameter. Values closer to one are dissipated less. In the right plot, the region near $(x,\lambda) = (0,1)$ is magnified with and $(x, 1-\lambda)$ plotted using logarithmic scales. In this plot, the order of falloff in dissipation as a function of vortex size manifests as the slope of the curve. For reference, lines with slopes 2 and 4 are included in the plot. Note that the orders are 2 for PIC and 4 for APIC, consistent with the analysis in Section 3.1.6.

Explicit form of eigenvalues. The eigenvalues $\lambda_i = \lambda_{rs}$ are given by the Fourier transform

$$\lambda_{rs} = \sum_{u=0}^{m} \sum_{v=0}^{n} c_{uv} e^{\frac{2\pi i r u}{m}} e^{\frac{2\pi i s v}{n}} = \sum_{u=-w}^{w} \sum_{v=-w}^{w} c_{uv} e^{\frac{2\pi i r u}{m}} e^{\frac{2\pi i s v}{n}} \tag{11}$$

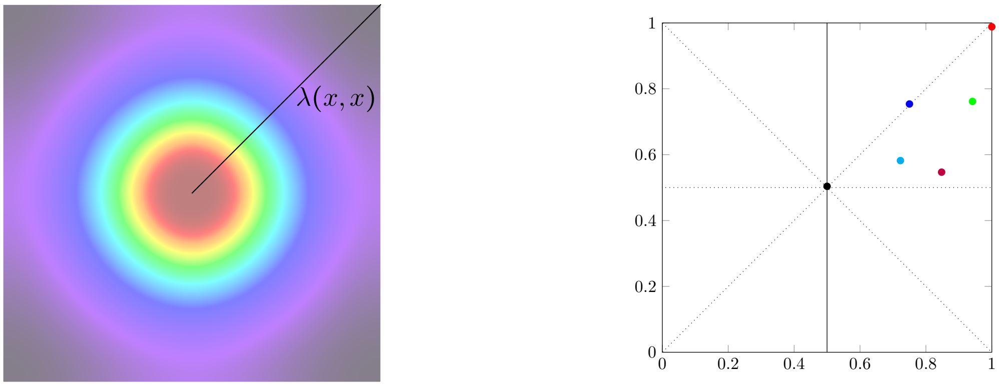
This gives us the eigenvalues for *any* size grid. If we index $\lambda(x,y)$ instead with rational numbers in the range $-\frac{1}{2} \le x, y < \frac{1}{2}$, where $x = \frac{r}{m}$ and $y = \frac{s}{n}$.

$$\lambda(x,y) = \sum_{u=-w}^{w} \sum_{v=-w}^{w} c_{uv} e^{2\pi i x u} e^{2\pi i y v} \tag{12}$$

Observe that this does not depend on the resolution $m \times n$ of the grid. Indeed, we can treat this as a continuous function for eigenvalues. The eigenvalues for any finite resolution are obtained by sampling the appropriate location within the continuous map. The map is symmetric, with $\lambda(x,y) = \lambda(-x,y) = \lambda(x,-y) = \lambda(-x,-y)$. The constant mode corresponds to $\lambda(0,0)$ and lies in the middle of the image. This is how the images shown in Figures 3 are constructed.

3.1.3. Numerical study - regular seeding

In the first study, we compute eigenvalue images corresponding to a range of parameters. We test PIC and APIC transfers using linear, quadratic, and cubic splines (Figure 3). In each ease, we seed with 4



(a) Location of cross sections.

(b) Single particle seed locations.

Figure 5: Guides for Figures 4 and 6. The particle seed locations indicate where the particles were located for Figure 6. The center black dot corresponds to placing the particle at the location of the grid degree of freedom.

particles per cell using either regular seeding (particles seeding in a regular grid pattern) or irregular seeding (4 particles in a blue noise pattern, with all cells having the same arrangement of particles). In each case, low-frequency modes lie in the center of the image. Dark red modes are near 1, and black modes are near zero, with colors assigned on a logarithmic scale according to the color bars shown with the images. (All of the eigenvalue images are shown with the same color scale, which is repeated for convenience.) Images with larger red regions near the middle are less dissipative. From the images, we see that APIC is dramatically less dissipative than PIC. Comparing across splines, we see that dissipation increases with spline order. This is not particularly surprising, since higher order splines interpolate over a larger range. The difference between cubic and quadratic splines is quite modest, and we do not see a strong motivation to prefer one over the other on grounds of dissipation.

To make the differences between the methods easier to see, 1D cross sections from these eigenvalue images are shown in Figures 4. The location of the cross sections are illustrated in Figure 5a. For reference, the results are shown for linear splines with 4, 9, and 64 particles per cell to illustrate the sampling dependence for linear splines. Quadratic and cubic splines are not sensitive to sampling density. The curves for these splines at higher sampling density have been omitted since they overlap the corresponding curve at 4 particles per cell. Note that $\lambda(x,0) = \lambda(0,x)$ due to symmetry of the particle distribution. The difference in damping is especially visible in the zoomed-in versions of these plots. Observe that APIC remains very close to 1 for much larger x than PIC due to the extra zero derivatives at the origin.

3.1.4. Numerical study - irregular seeding

Our second eigenvalue image study shows the sensitivity of transfers to particle positioning within a cell. In these tests, we use one particle per cell placed in one of the positions shown in Figure 5b. The

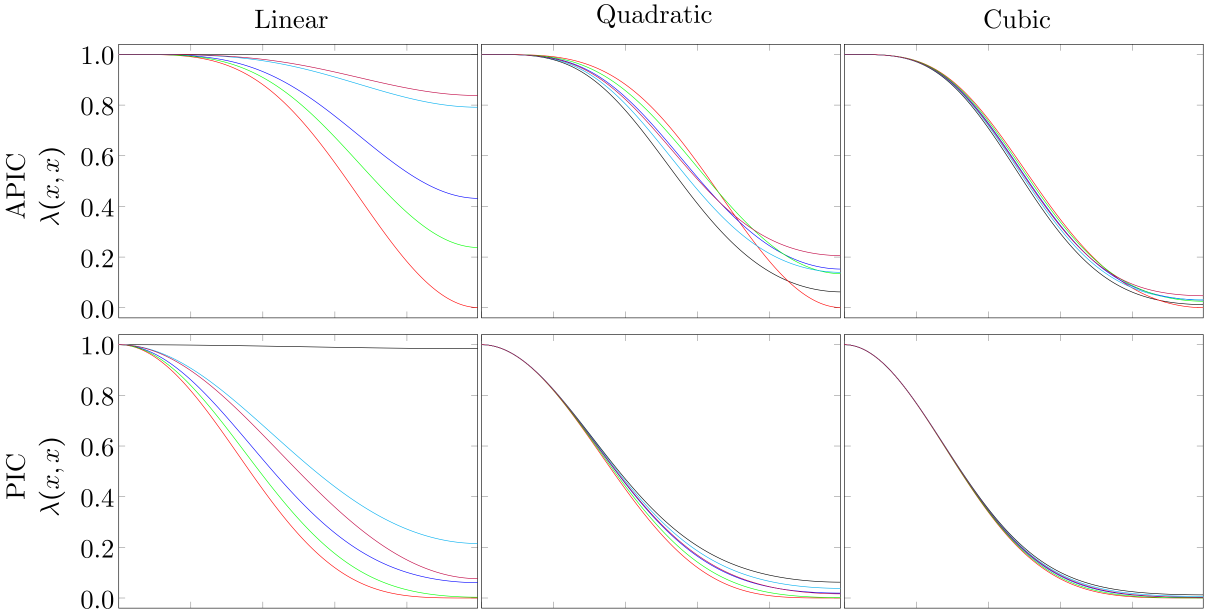


Figure 6: Eigenvalues for APIC and PIC, with linear, quadratic, and cubic splines and one particle per cell. The curve colors indicate the location of the particle in a cell, which are shown in Figure 5b. The particle position dependence decreases as the interpolation order increases, and this dependence is somewhat more pronounced for APIC.

center (black) point corresponds to putting the particle at the grid degree of freedom location. The red location corresponds to offsetting half a grid cell diagonally away from the degrees of freedom. Because of symmetry, we can restrict our samples to half of one quadrant. Each cell has one particle in the same location. Results are shown in Figure 6. As one might expect, one generally observes that particle position dependence decreases as interpolation order increases. Consistent with the above analysis, eigenvalues are larger for APIC than PIC, reflecting its reduced dissipation. One may regard these curves as representing (approximately) the *range* of possible eigenvalues that may result from a particular type of transfer (PIC, APIC) with a particular interpolation kernel (linear, quadratic, cubic). The eigenvalue plots for multiple particles are an average of the plots for each particle individually. Thus, it is possible to place bounds on how much dissipation is possible *independent of the particle distribution*. Of course, the distribution is still the same in each cell, so this should be taken as a guide rather than as a hard bound.

This approach to analyzing the effects irregular seeding, though informative, is still quite limited. Every cell has the same number of particles in the same locations, so the overall particle distribution is still very regular. Indeed, Fourier analysis is only possible because of this overall regularity. A globally irregular seeding may still yield eigenvalues (and thus produce dissipation) that is different from what is seen in this analysis. Tiling the particle distribution over a larger block size (for example, having a fixed irregular particle distribution within each 2 × 2 block of grid cells) would provide a tradeoff between sampling a more irregular particle distribution and the ability to apply Fourier analysis. This would produce 8 eigenvalue images with half the resolution of the original since Fourier analysis must now be performed over blocks. We do not purse this strategy here.

3.1.5. Taylor-Green vortex

In this analysis, we are particularly interested in understanding the tendency for transfers to damp out vorticity. Our model for a vortex is the Taylor-Green vortex, which has a convenient representation in terms of Fourier basis modes. They are also an analytic solution to the Navier-Stokes equations. This makes it an ideal model for studying dissipation.

Let the physical dimensions of our domain be $[-\pi, \pi] \times [-\pi, \pi]$ and assume that the resolution is square, with $m = n$. The Taylor-Green vortex is given by

$$\mathbf{v}(x, y) = \langle -\sin(ax)\cos(ay), \cos(ax)\sin(ay)\rangle, \tag{13}$$

where a is an integer that determines the scale of the vortex. This represents a $2a \times 2a$ checkerboard pattern of vortices which alternate between rotating clockwise and counterclockwise. An example of a Taylor-Green vortex is shown in Figure 7a. Larger a correspond to larger numbers of smaller vortices. Stretched vortices may also be considered,

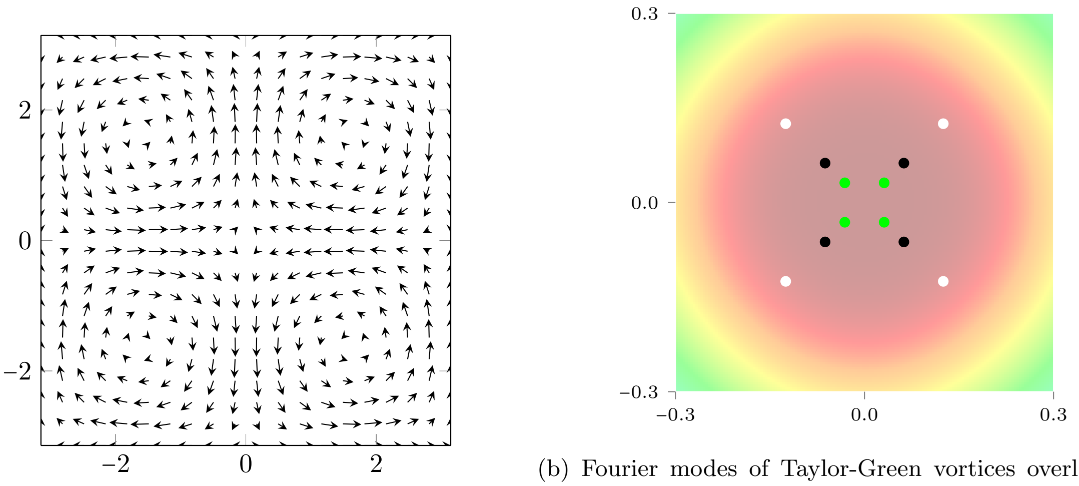
$$\mathbf{v}(x, y) = \langle -\sin(ax)\cos(by), \cos(ax)\sin(by)\rangle, \tag{14}$$

with different scales on x and y, but these are not solutions to the Navier-Stokes equations. They are, however, conveniently expressed in Fourier modes.

Location of Taylor-Green eigenvalues in image. The Fourier mode with wavenumber (k_1, k_2) is $e^{2\pi i x k_1} e^{2\pi i y k_2}$. Observe that the Taylor-Green vortex $\mathbf{v}(x, y)$ is a linear combination of the four modes $(\pm a, \pm a)$. The corresponding eigenvalues are $\lambda(x, y) = \lambda(\pm\frac{a}{m}, \pm\frac{a}{n})$. The location of Taylor-Green modes is shown in Figure 7b for a few sample values of a and resolutions.

Scaling of Taylor-Green. Due to symmetry, these four eigenvalues are all equal. Thus, the Taylor-Green vortex is transferred into a scaled copy of itself. The factor by which it is reduced is $\lambda(\frac{a}{m}, \frac{a}{n})$. Note that as the resolution $m \times n$ is increased, $(\frac{a}{m}, \frac{a}{n}) \to (0, 0)$. The key to understanding the behavior of vortices under refinement is thus understood by examining the behavior of $\lambda(x, y)$ near $(0, 0)$. We will use the differentiability of $\lambda(x, y)$ to characterize $\lambda(x, y)$ near $(0, 0)$ in Section 3.1.6.

On the other hand, a Taylor-Green vortex of a fixed resolution (e.g., 8 grid cells across) corresponds to a fixed place in the eigenvalue image ($\lambda(\pm\frac{1}{16}, \pm\frac{1}{16})$ in this case). The eigenvalue image thus gives a direct indication of the number of pixels required to resolve a Taylor-Green vortex with a specified amount of dissipation. This is not surprising, since the local dissipation of a vortex should not depend on how large the



(a) An example of Taylor-Green vortex on $[-\pi, \pi]^2$ with $a = 1$ and $b = 1$

(b) Fourier modes of Taylor-Green vortices overlaid on transfer eigenvalues. The location and color of the dots indicate the size (8 pixels: white, 16 pixels: black, 32 pixels: green) of Taylor-Green vortex.

Figure 7: Taylor-Green vortex and its Fourier modes

overall computational domain is. Some Fourier modes corresponding to Taylor-Green vortices are shown in Figure 7b.

3.1.6. Dissipation under refinement

In this section, we analyze how the dissipation of Taylor-Green vortices (and stretched vortices) changes under refinement. Fix an initial resolution $m \times n$ and a stretched Taylor-Green vortex (a, b). Each round trip transfer scales this vortex by $\lambda(\frac{a}{m}, \frac{b}{n})$. Next, lets scale the resolution to $qm \times qn$. As noted in Section 3.1.5, for $\lambda(\frac{a}{qn}, \frac{a}{qn}) \to \lambda(0, 0)$ as $q \to \infty$. Since all of the transfers under consideration preserve constant velocity fields, $\lambda(0, 0) = 1$. Approximating $\lambda(x, y)$ by a Taylor series and noting $0 \le \lambda \le 1$, we have $\lambda(\frac{a}{qn}, \frac{a}{qn}) \approx 1 - cq^{-\gamma}$. This approximation corresponds to the first $\gamma - 1$ mixed partial derivatives of $\lambda(x, y)$ vanishing at $(0, 0)$. It can be shown that $\gamma = 2$ for PIC and $3 \le \gamma \le 4$ for APIC. APIC achieves $\gamma = 4$ for quadratic and cubic splines due to the special properties of those splines. The constant c depends on many things, including the layout of the particles.

Under spatial refinement (as $q \to \infty$), we must take more time steps. Assume that $\Delta t = c_2 \Delta x^{\kappa}$, where likely values include $\kappa \in \{1, \frac{3}{2}, 2\}$. Thus, even though dissipation is less at higher resolution, the transfers must be repeated more often. The net dissipation λ is thus

$$\lambda \approx (1 - cq^{-\gamma})^{Tq^{\kappa}} \approx 1 - c_3 q^{\kappa - \gamma} = 1 - c_4 \Delta x^{\gamma - \kappa} \tag{15}$$

Thus, for first order convergence, we must have $\gamma \ge \kappa + 1$, since otherwise dissipation alone creates an error greater than first order. For PIC, this limits us to $\kappa = 1$. This rules out explicit surface tension ($\kappa = \frac{3}{2}$).

On the other other extreme, APIC with quadratic or cubic splines gives $4 - \kappa$, which is compatible with third order accuracy with $\Delta t = O(\Delta x)$ and with second order accuracy with $\Delta t = O(\Delta x^2)$. An interesting consequence of this appears to be that APIC cannot be the basis of a method that is higher than third order accurate in space. The predicted dissipation orders of 2 for PIC and 4 for APIC are observed numerically in Figure 4.

3.2. Particle to grid to particle

Limitations of grid-particle-grid. The grid-particle-grid view of transfers is convenient since its results are concisely described by a single image. This image tells us how dissipative transfers are in the limit $\Delta t \to 0$. Unfortunately, the grid-particle-grid path is not linear for FLIP, since new particle velocities ultimately depend on old particle velocities.

Particle-grid-particle. An alternative way to examine dissipation is to start with information on particles and simulate the steps that occur until we transfer velocities back to particles. Since none of the methods store state on the grid, this avoids the problem with the grid-particle-grid view for FLIP. The particle-grid-particle view, however, is quite a bit more complicated. This path includes the pressure projection step, which mixes velocity components in different directions. When each cell gets the same particle distribution, all cells are indistinguishable; the particles within a cell will all be distinguishable unless the particle distribution within a cell is highly symmetrical. In the case of APIC, the $\mathbf{B}_p^n$ matrices must also be considered as degrees of freedom. In 2D with p particles per cell, each cell will contribute $d = 2p$ degrees of freedom for PIC an FLIP but $d = 6p$ for APIC. Rather than a transfer operator M_{ij} that is a scalar per grid node, we must consider our operator to be $M_{(ia)(jb)}$, where a and b run over the d degrees of freedom per cell.

Pressure step is required. If we ignore the complications introduced by pressure projection and just do the transfers as we did before, then we run into a different problem. If we make no changes to the grid velocity before transferring back to particles, then FLIP will map back a zero difference. The resulting particle-grid-particle map is an identity map, which does not tell us anything interesting about the transfers. To get any useful insight into FLIP, we must include pressures.

Including pressure. We note that pressure projection is linear and is also conveniently diagonalized by the Fourier transform. These properties make it compatible with our analysis. (Viscosity also shares these properties, and one could include it in the analysis. Since our interest is in the inviscid case, we do not do this.) The pressure projection introduces another complication: the particle-grid-particle map is no longer sparse. This simply means the transfers and Poisson solve must be done at the resolution of the final image, which is not a significant problem. Although pressure projection is very convenient to perform directly in Fourier space, we apply pressure using the same central difference discretization we use for simulation.

3.2.1. Analysis procedure

Block tensor product circulant. In the grid-particle-grid case, we were able to reduce the problem to one degree of freedom per cell. The resulting transfer M_{ij} was tensor product circulant and could be diagonalized with a multidimensional Fourier transform. The situation is now more complicated, since each cell has d degrees of freedom associated with it. The matrix $M_{(ia)(jb)}$ is still *block tensor product circulant* in the indices i, j, but not in a, b. That is, if $i = (r, s)$ and $j = (u, v)$ are grid indices in 2D, then $M_{(r,s,a),(u,v,b)} = M_{(r+k,s+m,a),(u+k,v+m,b)}$ for any k and m. Rather than diagonalizing the matrix, the Fourier transform will bring the matrix into a block-diagonal structure.

Constructing and representing the operator. The first column of this operator no longer suffices to represent the entire operator, but the first d columns do. That is, $M_{(r,s,a),(u,v,b)} = c_{r-u,s-v,a,b}$. The columns c_{iab} are obtained by initializing all particle velocity degrees of freedom to zero, except particle velocity degree of freedom b in the first grid cell. This is repeated for each of the d degrees of freedom b in the first cell.

Diagonalization procedure. Although the matrix $M_{(ia)(jb)}$ is no longer fully diagonalizable by Fourier transforms, Fourier transforms can still be used to render it block diagonal with $d \times d$ blocks. By computing d^2 multidimensional Fourier transforms, we have

$$\beta_{rsab} = \sum_{u=0}^{m} \sum_{v=0}^{n} c_{uvab} e^{\frac{2\pi i r u}{m}} e^{\frac{2\pi i s v}{n}} \tag{16}$$

The β_{rsab} are the diagonal blocks. It associates to every grid cell $i = (r, s)$ a matrix $N_{ab} = \beta_{rsab}$. The eigenvalues of the $d \times d$ matrices N_{ab} are the eigenvalues of $M_{(ia)(jb)}$. Since d is relatively small, we simply compute the eigenvalues of these blocks directly. This procedure computes the $m \times n \times d$ eigenvalues of $M_{(ia)(jb)}$ and conveniently associates d eigenvalues to every cell based on frequency.

Visualization. In the grid-particle-grid case, we had a single eigenvalue per grid cell, which we were able to plot conveniently as an image. In the particle-grid-particle case, we now have d eigenvalues with no particular ordering. We sort the d eigenvalues in each cell by magnitude and construct d images. The smallest eigenvalue in each cell corresponds to the first image, the second smallest eigenvalues correspond to the second image, and so on. The visualizations of the results for all methods are shown in Figure 8.

3.2.2. Analysis results - PIC and APIC

The results that are obtained for PIC and APIC are quite simple and easily understood. The image formed from the largest eigenvalue is the same image that is constructed in the grid-particle-grid case. The image formed from the second-largest eigenvalue is 1 in the center and zero elsewhere. All of the remaining images are zero.

To see why this is the case, lets consider the three steps involved in the construction of the operator $M = APB$, where B is the transfer from particle to grid, P is the pressure projection, and A is the transfer from grid to particle. We can transform this into Fourier space, in which case we can write $\hat{M} = \hat{A}\hat{P}\hat{B}$, where hats represent the individual operators in Fourier space. Observe that each of these operators is block-diagonal, with one block per grid cell. Each block $\hat{A}$ is $d \times 2$, $\hat{P}$ is 2×2, and $\hat{B}$ is $2 \times d$. Thus, M has rank at most 2. This explains why only two images are nonzero; the $d - 2$ zero images correspond to nullspace modes of the particle to grid transfer.

Next, we turn to the first two images. Observe that $\hat{M} = \hat{A}\hat{P}\hat{B}$ and $\hat{N} = \hat{B}\hat{A}\hat{P}$ have the same nonzero eigenvalues with the same multiplicity. (Indeed, if u is a vector with nonzero eigenvalue λ, then $\lambda u = \hat{M}u = \hat{A}\hat{P}\hat{B}$. $v = \hat{B}u$ is an eigenvector of $\hat{N}$, since $\hat{N}v = \hat{B}\hat{A}\hat{P}\hat{B}u = \lambda\hat{B}u = \lambda v$. Observe that $\lambda \neq 0$ implies $v \neq 0$. It is also easy to see that distinct eigenvectors of $\hat{M}$ for the same nonzero eigenvalue map to distinct eigenvectors of $\hat{N}$.)

The operator $\hat{N} = \hat{B}\hat{A}\hat{P}$ is composed of two pieces. Recall that BA is the grid-particle-grid transfer operator. The blocks of $\hat{B}\hat{A}$, which I denote using subscripts as $(\hat{B}\hat{A})_i = \hat{B}_i\hat{A}_i$ must be of the form $\lambda_i\mathbf{I}$, where λ_i was the eigenvalue computed for that cell in the grid-particle-grid case. Thus, $\hat{N}_i = \lambda_i\hat{P}_i$.

For $i = (0,0)$, $\hat{P}_i = \hat{P}_{(0,0)} = \mathbf{I}$, since the constant translation modes are divergence free. Since constant translation is preserved by PIC and APIC, $\lambda_{(0,0)} = 1$. This explains why the middle pixel of both images corresponds to eigenvalue 1.

For any other $i \neq (0,0)$, $\hat{P}_i = \mathbf{I} - \frac{ii^T}{i^Ti}$ is a rank-one projection operator with eigenvalues 0 and 1. The eigenvalues of $\hat{N}_i$ are 0 and λ_i. Since the largest eigenvalue is always λ_i, the image of maximum eigenvalues is the same as the grid-particle-grid case. The second eigenvalue of $\hat{P}_i$ being 1 for the middle and 0 elsewhere explains the image for the second-to-largest eigenvalue. Although the projection was actually performed with the finite difference stencil, the images look the same.

3.2.3. Analysis results - FLIP

The results for FLIP are very different. In this case, all images but one are identically 1. The only nontrivial image is filled with the smallest eigenvalues and is shown in the last row of Figure 8.

The explanation for the images with eigenvalue 1 everywhere is similar to the reason for the trivial images for PIC and APIC. One of the trivial images corresponds to the modes that are transferred to the grid, resulting in nonzero divergence-free velocity fields that are unaffected by the pressure projection. Since no grid velocity change occurs, FLIP behaves as the identity map on these modes. The other trivial images correspond to null modes of the particle to grid transfer. These produce a zero grid velocity, which is unchanged by pressure. In the case of FLIP, however, the grid velocity difference (zero) is mapped back, resulting in no change. The operator behaves as the identity map on these modes.

That leaves the nontrivial mode. This appears qualitatively similar to the nontrivial modes observe for PIC, except that it is inverted and has a single pixel corresponding to eigenvalue 1 in the middle. (Compare with Figure 6.) This is the mode corresponding to the pressure projection. The eigenvalue 1 in the middle

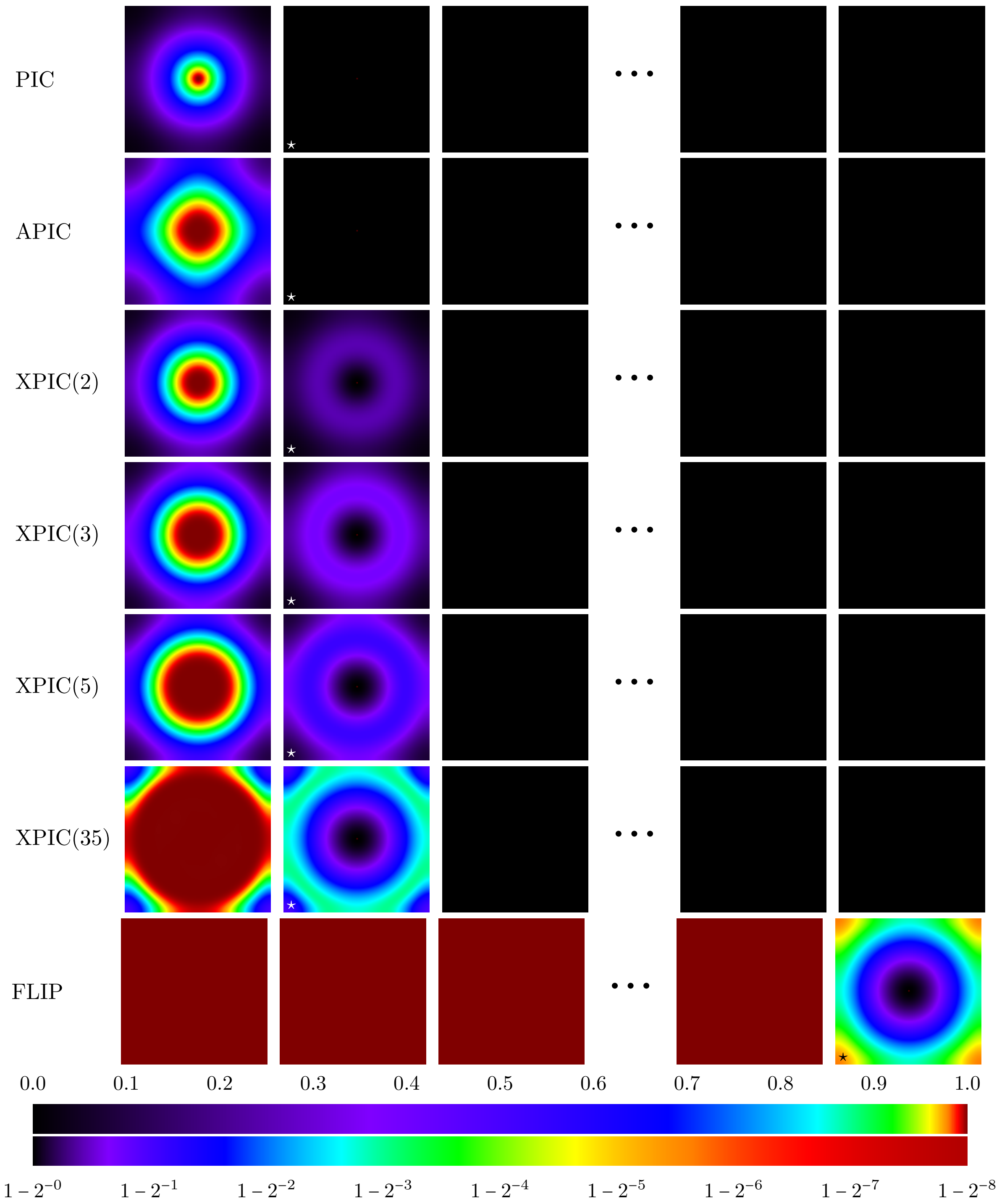


Figure 8: Eigenvalues for particle-grid-particle transfers, with **4 particles per cell** and **quadratic** splines. Some trivial eigenvalue images are omitted. Images with a “⋆” have a red dot in the center of the image (constant velocity mode). The dissipation of APIC is approximately between XPIC(2) and XPIC(3). The dissipation of XPIC(m) improves significantly with order. In exchange, XPIC(m) lets through some undesirable modes (second column), which is minimal for low orders but grows steadily with m. The general behavior of FLIP is radically different from XPIC(m) or APIC; it lets through almost everything.

corresponds to the constant velocity being divergence free. In the case of PIC and APIC, the rest of the image would be black, since all other divergent modes should be projected out. In the case of FLIP, however, this is not so. For low-frequency modes, the image appears black, and these divergent modes are projected well. Higher frequency modes, however, do not readily survive the transfer to the grid. Since only a relatively small portion of the high-frequency velocity modes survive to the grid, only that small amount of velocity can be projected out by the pressure solve. Since only a small grid change was made, only a small change is made to the particle. The result is that high-frequency divergent velocity modes on particles are not efficiently projected. This goes some way towards explaining why updating FLIP particle positions with particle velocities produces very bad results; in addition to the particle velocities being noisy, the velocity field on the particles is not even divergence free.

3.3. Analysis results - XPIC

In XPIC, all but two eigenvalues are all zero; this is related to the nullspace of XPIC transfer. This is an immediate improvement from FLIP, since the amount of noise that can survive on particles is already drastically reduced. In this way, XPIC is far more like PIC or APIC than it is like FLIP. The largest of the nontrivial eigenvalues is similar to APIC, and the eigenvalue improves as the XPIC order increases. XPIC(1) matches PIC, since the schemes are the same. The dissipation of APIC lies somewhere between XPIC(2) and XPIC(3). For higher orders, XPIC is significantly less dissipative than APIC. In practice, XPIC is typically run at XPIC(2) or XPIC(5), with the latter being significantly less dissipative than APIC.

When we start looking at the second largest eigenvalue, we see that the reduced dissipation of XPIC comes at a minor cost in the effectiveness of projection. Because XPIC performs particle-to-grid and grid-to-particle transfers repeatedly without projecting divergent modes, these divergent modes can be present in the particle velocity. These modes are not removed by pressure projection and their frequency response appear as “halo” in the other nontrivial eigenvalue (see the XPIC rows in Figure 8). Even by XPIC(3), some of these halo eigenvalue are already larger than 0.4. Practically, this means that particle divergence may persist for several time steps, but it cannot accumulate over time. It should be noted that XPIC was not constructed as a transfer scheme for incompressible fluids, so the interplay between transfers and projection was not a consideration during its development.

4. Implementation notes

4.1. Extrapolation

When a boundary is not periodic, transfers will require information from outside the fluid domain. We handle this by extrapolating information into the ghost region. For all transfer algorithms, extrapolation happens before the grid-to-particle transfer and after the particle-to-grid transfer.

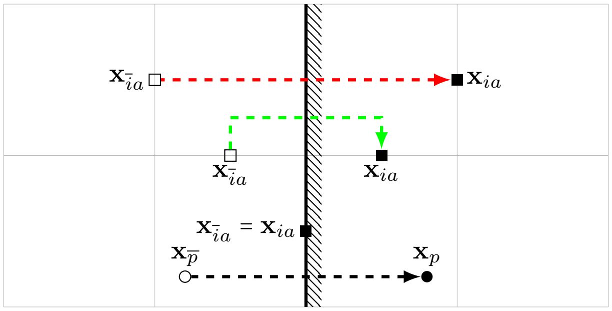


Figure 9: Grid or particle attributes (solid) and their reflected counterparts (hollow). Faces on the boundary are their own reflections. The hatched sides indicate the inside domain.

Reflected particles. Consider two physical scenarios. In the first we have particles that are moving towards a solid wall, which causes them to stop moving normal to the wall and slide tangentially along it. In the second scenario, the wall is missing but instead a mirror image of the particles is added on the other side of the wall. The particles will move towards the plane where the wall was and collide with each other, resulting in them sliding tangentially along where the wall was. The results is these two cases are the same. We can therefore mimic the effects of colliding with a wall by simply mirroring the particles on the other side. We do not actually create extra particles outside; we simply perform a *touch up* after doing particle-to-grid transfers so that they behave as if there were reflected particles. The corrections depend on the type of boundary condition and are simple modifications that are applied to the grid in a thin layer near the boundary. These take the form of adding data that was transferred into the ghost region to the reflected location inside, possibly with a change of sign.

We use a bar over a quantity to denote its corresponding reflected counterpart across the grid boundary. For example, $\mathbf{x}_{\bar{i}a}$ and $\mathbf{x}_{\bar{p}}$ are the locations of reflected grid faces and particles. For contrast, $\overline{\mathbf{x}_{ia}}$ and $\overline{\mathbf{x}_p}$ are the reflected locations of a grid face and a particle. Since we only perform this reflection across grid faces, reflecting the location of one face always results in the location of another face. We are thus justified in defining $\mathbf{x}_{\bar{i}a} = \overline{\mathbf{x}_{ia}}$ and $\mathbf{x}_{\bar{p}} = \overline{\mathbf{x}_p}$. Note that the particle represented by $\bar{p}$ is only conceptual and for derivation purposes; we do not actually construct these particles.

They are shown in Figure 9. Since the basis functions are invariant to reflections, the weight generated from reflected particle to the inside domain index, $w_{i\bar{p}a} = N(\mathbf{x}_{ia} - \mathbf{x}_{\bar{p}}) = N(\overline{\mathbf{x}_{\bar{i}a} - \mathbf{x}_p}) = N(\mathbf{x}_{\bar{i}a} - \mathbf{x}_p) = w_{\bar{i}pa}$. Similarly we also have $w_{\bar{i}\bar{p}a} = w_{ipa}$. Consider a general grid attribute q_{ia} (which could be mass or a component of velocity, momentum, or force) inside the domain consists of contributions from internal particles and reflected particles. We want the reflected value to be $q_{\bar{p}} = bq_p + c_p$, where $b = \pm 1$ and c_p depend on the type of attribute and boundary conditions (See Figure 10). Assuming for simplicity a PIC transfer,

$$q_{ia} = \sum_p w_{ipa} q_p + \sum_{\bar{p}} w_{i\bar{p}a} q_{\bar{p}} = \sum_p w_{ipa} q_p + \sum_p w_{\bar{i}pa}(bq_p + c_p)$$

$$q_{\bar{i}a} = \sum_p w_{\bar{i}pa} q_p + \sum_{\bar{p}} w_{\bar{i}\bar{p}a} q_{\bar{p}} = \sum_p w_{\bar{i}pa} q_p + \sum_p w_{ipa}(bq_p + c_p)$$

$$bq_{\bar{i}a} = \sum_p w_{\bar{i}pa} bq_p + \sum_p w_{ipa}(q_p + bc_p) = q_{ia} - \sum_p w_{\bar{i}pa} c_p + b\sum_p w_{ipa} c_p$$

Compare these with the grid values $\hat{q}_{ia}$ that would be if no reflected particles were used.

$$\hat{q}_{ia} = \sum_p w_{ipa} q_p \qquad \hat{q}_{\bar{i}a} = \sum_p w_{\bar{i}pa} q_p$$

In the case $b = 1$ and $c_p = 0$, these rules amount to $q_{ia} = q_{\bar{i}a} = \hat{q}_{ia} + \hat{q}_{\bar{i}a}$. That is, simply copy the ghost values into the interior, and then copy these values back into the ghost region. In the case $b = -1$ and $c_p = 0$, $q_{ia} = -q_{\bar{i}a} = \hat{q}_{ia} - \hat{q}_{\bar{i}a}$. This is implemented by subtracting ghost values from the inside values, then copying the inside data to the ghost region with a sign flip. Notice that these rules are very simple grid-based fixes that only need to be applied near the boundary.

The case $c_p \neq 0$ is only needed for inhomogeneous boundary conditions, which are only relevant for velocities. In this case, we would be enforcing $v = v_{bc}$ at the boundary. However, we actually transfer momentum from particles to grid, not velocity directly. Thus, we must reflect about the desired value of momentum for the particles, $c_p = 2v_{bc}m_p$. Applied to the transfer rules, we see

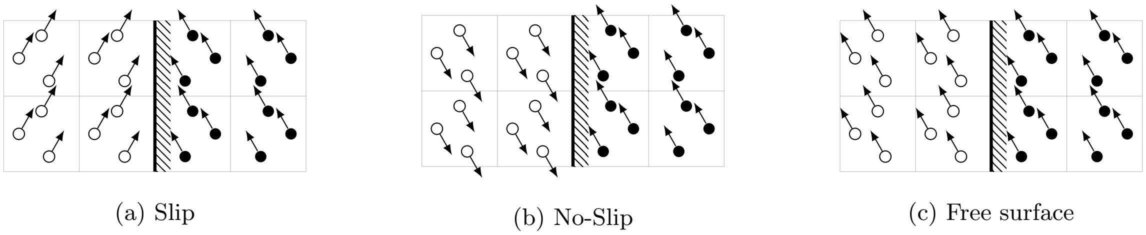
$$q_{ia} = \sum_p w_{ipa} q_p + \sum_p w_{\bar{i}pa}(bq_p + c_p) = \hat{q}_{ia} + b\hat{q}_{\bar{i}a} + \sum_p w_{\bar{i}pa} 2v_{bc} m_p = \hat{q}_{ia} + b\hat{q}_{\bar{i}a} + 2m_{\bar{i}a} v_{bc}$$

$$bq_{\bar{i}a} = q_{ia} - \sum_p w_{\bar{i}pa} 2v_{bc} m_p + b\sum_p 2w_{ipa} v_{bc} m_p = q_{ia} + 2(b\hat{m}_{ia} - \hat{m}_{\bar{i}a})v_{bc},$$

where $\hat{m}_{ia}$ also refers to velocity before the boundary condition treatment has been applied. The same correction rules can be obtained for APIC transfers by defining $\mathbf{b}_{\bar{p}a}$ appropriately.

We use $b = 1$ and $c_p = 0$ for (1) masses, (2) free surface momentum transfers, and (3) the tangential components of momentum transfers for slip boundary conditions. We use $b = -1$ and $c_p = 0$ for (1) no-slip boundary conditions and (2) normal components of momentum transfers for slip boundary conditions. See Figure 10. We use $c_p \neq 0$ for inhomogeneous velocity boundary conditions. We perform this kind of extrapolation for mass and momentum after the particle-to-grid transfer.

We also perform extrapolation after forces are applied. Ideally, velocities should continue to satisfy appropriate boundary conditions after forces have been applied. This is accomplished by extrapolating forces using the homogeneous version of the same boundary condition applied to velocities. For our analytic body force, no modifications should be made to the for inside region, since these forces are analytic and thus correct. We do, however, extrapolate the inside values to the ghost region.


(a) Slip

(b) No-Slip

(c) Free surface

Figure 10: Reflecting velocities across the boundary in different ways mimics different types of boundary conditions. The hatched side indicates the inside domain. The hollow circles are reflected particles.

There is another extrapolation performed after grid evolution, since the grid-to-particle transfers will read from the ghost region. As with analytical forces, we leave the interior values alone (the pressure projection boundary conditions ensure that the interior values already satisfying the boundary conditions) but extrapolate the interior values to the ghost regions according to the boundary condition type. This step is especially important with FLIP transfers, since FLIP will compute the velocity difference based on current and previous velocity, which has been already extrapolated.

We also note that there will be overlapped access at the corners of ghost region. The extrapolation rules for each direction are compatible with each other, this is just a matter of correct implementation. Accumulate ghost data into the interior first, then extrapolate it back to the ghost region as a second pass.

4.2. Cut-cell formulation

Some of our tests include irregular objects. For these tests we use the more accurate pressure projection formulation of [32]. The objects are represented by a level set on grid nodes. Nodes not in an object are in fluid. MAC faces hold valid velocity degrees of freedom if any of their nodes are in fluid. MAC cells hold valid pressures if any of their nodes are in fluid. Note that if a MAC face is valid then both neighboring cells are also valid. See Figure 11. This discretization is just the regular central differencing stencil away from objects. Since our objects do not touch the domain walls, we handle domain wall boundary conditions as in the finite differencing discretization.

Three concerns must be addressed for this layout to be valid. (1) Valid velocities must be available at all MAC faces being projected. Each valid face has at least one valid node. Provided the geometry is adequately resolved on the grid, this valid node will have a cell that is entirely inside fluid; reseeding will guarantee that this cell has particles. All of these particles will contribute to the velocity on the face being considered provided the transfer stencil is at least as wide as the quadratic stencil. (2) Valid faces must have valid pressures on either side (except at the domain walls) so that the pressure gradient can be applied. Our discretization has this property as noted above. (3) We must have enough valid grid velocities to transfer back to particles. This is not normally true for us, and the way we handle this determines the type of boundary condition we enforce. If the object has no-slip boundary conditions, then we set the invalid

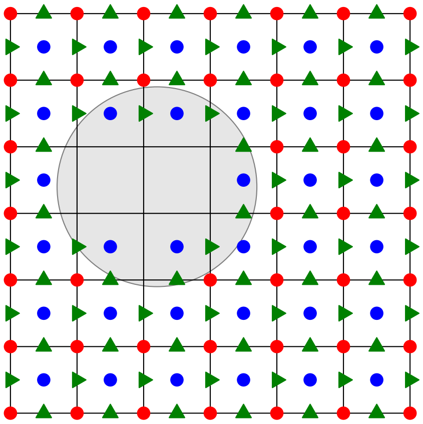


Figure 11: Cut-cell layout for our degrees of freedom. Red nodes are inside the fluid. Blue cell centers contain pressure degrees of freedom during the pressure projection. The green triangles are MAC faces where fluid velocities are being projected.

velocities inside the object with the object's velocity. If the object has no-slip boundary conditions, then we do moving least squares transfers as described below.

4.3. Moving least squares

APIC transfers may be formulated as Moving Least Squares (MLS) [28, 29] by expressing the transfers as a least squares optimization. If there is only one particle p and we were doing co-located APIC transfers, then a grid-to-particle-to-grid ($\tilde{\mathbf{v}}_i^{n+1} \to \mathbf{v}_i^{n+1}$) with $\Delta t = 0$ (so that $w_{ip}^{n+1} = w_{ip}^n$ and $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n$) can be expressed as

$$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} \qquad \mathbf{B}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n)^T \qquad \mathbf{D}_p^{n+1} = \sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n)(\mathbf{x}_i^n - \mathbf{x}_p^n)^T$$

$$m_i^{n+1} = w_{ip}^n m_p \qquad \mathbf{C}_p^{n+1} = \mathbf{B}_p^{n+1} (\mathbf{D}_p^{n+1})^{-1} \qquad m_i^{n+1} \mathbf{v}_i^{n+1} = w_{ip}^n m_p (\mathbf{v}_p^{n+1} + \mathbf{C}_p^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n))$$

The same $\mathbf{v}_p^{n+1}$ and $\mathbf{C}_p^{n+1}$ are obtained by choosing $\mathbf{v}_p^{n+1}$ and $\mathbf{C}_p^{n+1}$ to minimize the objective

$$E = \sum_i w_{ip}^n \left\| \mathbf{v}_i^{n+1} - \tilde{\mathbf{v}}_i^{n+1} \right\|^2 = \sum_i w_{ip}^n \left\| \mathbf{v}_p^{n+1} + \mathbf{C}_p^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n) - \tilde{\mathbf{v}}_i^{n+1} \right\|^2.$$

That is, the particle data is chosen so that the round trip transfer preserves the grid velocity field in a weighted least squares sense. The nice part about this formulation is that it makes sense when some data is invalid. Let $s_i = 1$ for valid nodes and $s_i = 0$ for invalid nodes. Then, $\mathbf{v}_p^{n+1}$ and $\mathbf{C}_p^{n+1}$ can be chosen to minimize

$$E = \sum_i s_i w_{ip}^n \left\| \mathbf{v}_p^{n+1} + \mathbf{C}_p^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n) - \tilde{\mathbf{v}}_i^{n+1} \right\|^2.$$

If $\mathbf{B}_p^{n+1}$ is used for state, then it can be computed with

$$\mathbf{D}_p^{n+1} = \sum_i s_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n)(\mathbf{x}_i^n - \mathbf{x}_p^n)^T \qquad \mathbf{B}_p^{n+1} = \mathbf{C}_p^{n+1} \mathbf{D}_p^{n+1}$$

For MAC transfers, a similar minimization problem can be formulated as

$$E = \sum_{ia} s_{ia} w_{ipa}^n \left(\mathbf{e}_a^T \mathbf{v}_p^{n+1} + (\mathbf{c}_{pa}^{n+1})^T (\mathbf{x}_{ia}^n - \mathbf{x}_p^n) - \tilde{v}_{ia}^{n+1} \right)^2,$$

This leads to the following algorithm. First, compute the intermediates

$$q_{pa} = \sum_i s_{ia} w_{ipa}^n \qquad \mathbf{g}_{pa} = \sum_i s_{ia} w_{ipa}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^n) \qquad \mathbf{D}_{pa}^{n+1} = \sum_i s_{ia} w_{ipa}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^n)(\mathbf{x}_{ia}^n - \mathbf{x}_p^n)^T$$

$$u_{pa} = \sum_{ia} s_{ia} w_{ipa}^n \tilde{v}_{ia}^{n+1} \qquad \mathbf{h}_{pa} = \sum_{ia} s_{ia} w_{ipa}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^n) \tilde{v}_{ia}^{n+1}$$

Then, solve a linear system and compute $\mathbf{b}_{pa}^{n+1}$ and $\mathbf{v}_p^{n+1}$.

$$\begin{pmatrix} q_{pa} & \mathbf{g}_{pa}^T \\ \mathbf{g}_{pa} & \mathbf{D}_{pa}^{n+1} \end{pmatrix} \begin{pmatrix} y_{pa} \\ \mathbf{c}_{pa}^{n+1} \end{pmatrix} = \begin{pmatrix} u_{pa} \\ \mathbf{h}_{pa} \end{pmatrix} \qquad \mathbf{b}_{pa}^{n+1} = \mathbf{D}_{pa}^{n+1} \mathbf{c}_{pa}^{n+1} \qquad \mathbf{v}_p^{n+1} = \sum_a y_{pa} \mathbf{e}_a.$$

This linear system is symmetric positive semidefinite but may be singular if insufficient valid data is available; it should be solved in the minimum norm least squares sense. The linear system is only 4×4 in 3D, so this algorithm is quite efficient. One may see that this algorithm decays into the usual transfers when all velocities are valid ($s_{ia} = 1$). In this case, $q_{pa} = 1$, $\mathbf{g} = \mathbf{0}$, $u_{pa} = y_{pa} = \mathbf{e}_a^T \mathbf{v}_p^{n+1}$, and $\mathbf{h}_{pa} = \mathbf{b}_{pa}^{n+1}$.

4.4. Reseeding

To correct the particle coverage, we reseed the particles in a similar way to [30]. For each cell the number of particles are counted and if it is below 2 we randomly sample new particles in the cell so that the number of particles meet the minimum requirement. The velocities for the new particles are interpolated from the grid. For a cell which is partially occupied by an object, we reject the sample positions that are inside the object. If the number of particles in a cell exceeds 2^{d+1} where d is the dimension, we randomly select particles for removal. We perform a particle-to-grid transfer if any new particles are added to ensure consistency between particles and grid. This extra transfer is only necessary for XPIC, but we perform it for all transfers to keep the evolution as similar as possible.

4.5. Grid forces

We use the body force in the Navier-Stokes equations as a forcing term to make chosen velocity and pressure fields into analytic solutions. This is done by substituting the velocity and pressure (assuming zero viscosity) into the equations:

$$\mathbf{f} = \frac{\partial \mathbf{v}}{\partial t} + \nabla \mathbf{v} \cdot \mathbf{v} + \frac{1}{\rho} \nabla p.$$

5. Numerical examples

5.1. Convergence

To demonstrate the correctness of the APIC transfers, as well as the correctness of the PIC and FLIP versions of the scheme used for comparison, we perform some simple convergence tests. All velocity errors are reported using the grid velocity at the end of the time step $\tilde{v}_{ia}^{n+1}$ as well as using the particle velocities at the end of the time step $\mathbf{v}_p^{n+1}$. For each test, we report the L^∞ and L^2 errors calculated according to the formulas

$$L_G^\infty = \max_{ia} \left| \tilde{v}_{ia}^{n+1} - \mathbf{v}(\tilde{\mathbf{x}}_{ia}^{n+1}, t^{n+1}) \cdot \mathbf{e}_a \right| \qquad L_G^2 = \sqrt{\frac{1}{N_G} \sum_{ia} \left(\tilde{v}_{ia}^{n+1} - \mathbf{v}(\tilde{\mathbf{x}}_{ia}^{n+1}, t^{n+1}) \cdot \mathbf{e}_a \right)^2}$$

$$L_P^\infty = \max_{p} \left\| \mathbf{v}_p^{n+1} - \mathbf{v}(\mathbf{x}_p^{n+1}, t^{n+1}) \right\|_\infty \qquad L_P^2 = \sqrt{\frac{1}{N_P} \sum_{p} \left\| \mathbf{v}_p^{n+1} - \mathbf{v}(\mathbf{x}_p^{n+1}, t^{n+1}) \right\|_2^2}$$

where $N_G = \sum_{ia} 1$, $N_P = \sum_{p} 1$ are the numbers of simulated grid faces and particles and $\mathbf{v}(\mathbf{x}, t)$ is the analytic velocity field.

5.1.1. Taylor Green

For this test, we verify the convergence of a Taylor-Green vortex. We use a $[-\pi, \pi]^2$ domain with initial velocity field $\mathbf{v}_0 = \langle -\sin(ax)\cos(ay), \cos(ax)\sin(ay) \rangle$. The fluid has physical properties $\rho = 3$ and $\mu = 0$. The analytic solution is $\mathbf{v}(\mathbf{x}, t) = \mathbf{v}_0(\mathbf{x}) e^{-2a^2 \nu t}$, where $\nu = \frac{\mu}{\rho}$. All of the tests were run with $\Delta x = \frac{2\pi}{N}$ and $\Delta t = \frac{1}{N}$ to a final time of $T = 1$, where N is the number of grid cells in each direction. The same test was run with PIC, APIC, and FLIP transfers. The same particle distribution is used for each test, which was computed as blue noise by Poisson disk sampling. In particular, the particle distribution is never the same per cell (and thus the transfer matrices are not circulant). Convergence plots are shown in the first row of Figure 12. A few observations stand out immediately from the convergence plots. PIC converges convincingly at first order with close agreement between particle and grid states. APIC converges cleanly at first order, but here the particles have some error relative to grids; this is because the extra velocity contribution on particles (due to $\mathbf{b}_{pa}^{n+1}$) was not taken into account during the error analysis; the significance of this difference diminishes under refinement. The most striking observation is with respect to FLIP. When errors are measured on the grid, first order convergence is observed. On particles, however, the convergence is weaker than first order. The reason for the reduced convergence order on FLIP is unknown, but it is suspected to be related to the accumulation of error in transfer null modes on particles (indeed, this error does not seem to affect grid convergence, though one may wonder for how long this situation can persist).

5.1.2. "Square"

For this test, we verify the convergence of an analytic velocity field with slip boundary conditions. First we make two stream functions $\phi_1(x, y) = f(x)f(y)$ and $\phi_2(x, y) = g(x)f(y)$, where $f(x) = x(1-x)(x^2-x-1)$

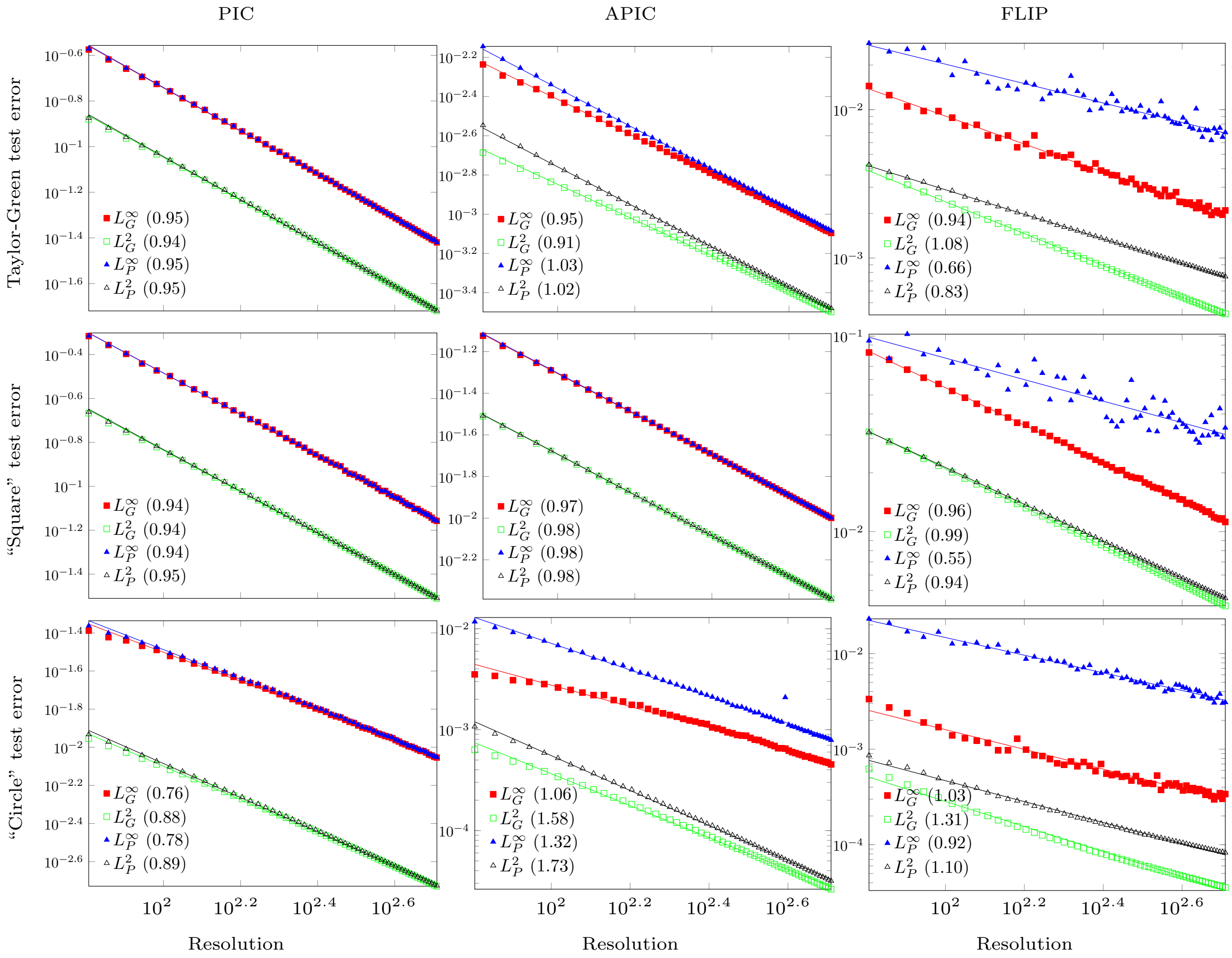


Figure 12: Convergence tests with three different transfers for each of the four error measures. The markers are the actual error computed, and the lines are least squares regression lines used to calculate the convergence order. Convergence orders are listed in the legends. Resolution is the number of cells in each direction of the grid.

and $g(x) = x(1-x)(x+1)(3x^2-7)$. Then the analytic velocity field constructed from these stream functions as $\mathbf{v} = \langle -\frac{\partial \phi_1}{\partial y}, \frac{\partial \phi_1}{\partial x} \rangle + t \langle -\frac{\partial \phi_2}{\partial y}, \frac{\partial \phi_2}{\partial x} \rangle$ on domain $[0,1]^2$. This velocity field is divergence free. The stream functions are chosen such that the normal directional derivative of tangential velocity vanishes on the boundary. Thus it is compatible with the slip boundary conditions. The analytic pressure is chosen as $p(x,y,t) = xy(1-x)(1-y)(x - xy + y^2 + t)$, so that $p = 0$ at the boundaries. The fluid has physical properties $\rho = 3$ and $\mu = 0$. All of the tests were run with $\Delta x = \frac{1}{N}$ and $\Delta t = \frac{1}{4N}$ to a final time of $T = 1$. The chosen analytic velocity field has a larger peak speed, so we use a smaller Δt for this simulation, and the convection would not affect out results. Convergence plots are shown in the second row of Figure 12. With the help of extrapolation, we get linear order convergence except L_P^∞ error of FLIP. This is due to the same reason as in the Taylor-Green convergence test.

5.1.3. "Circle"

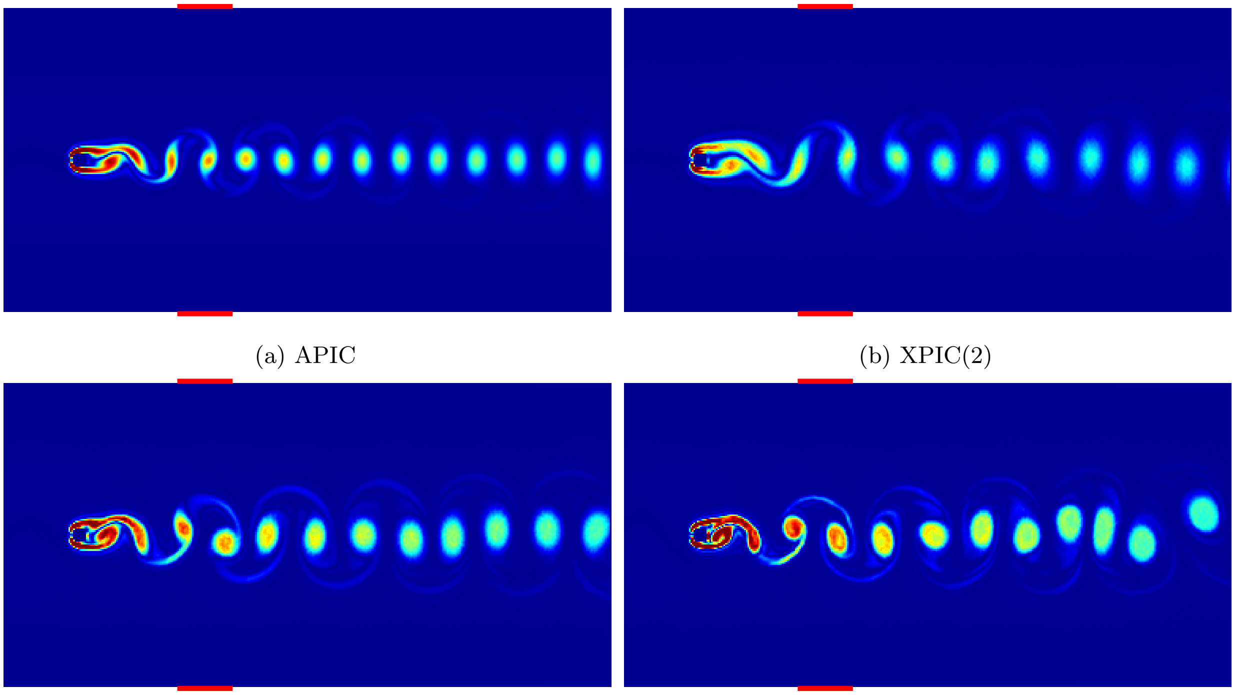
For this test, we verify the convergence of a velocity field with object boundaries that are not aligned with the grid. We use a $[-2,2]^2$ domain but exclude a circle of radius 1 at the origin. All boundaries are slip. We construct a divergence-free velocity from the stream function S defined by

$$\begin{aligned} S = &\frac{1}{177500000}(y-2)(y+2)(x-2)(x+2)(x^2+y^2-1) \\ &\times(1775x^6y^4 + 1775x^4y^6 - 14585x^6y^2 - 31229x^4y^4 - 14585x^2y^6 + 36100x^6 + 158486x^4y^2 \\ &+ 158486x^2y^4 + 36100y^6 - 248104x^4 - 538625x^2y^2 - 248104y^4 + 384500x^2 + 384500y^2 - 213392) \end{aligned}$$

This field is chosen so that the normal velocities and the normal derivative of tangential velocity are 0 at all boundaries. The analytic pressure is chosen as $p(x,y,t) = x - xy + y^2 + t$. The fluid has physical properties $\rho = 3$ and $\mu = 0$. All of the tests were run on an $N \times N$ grid with $\Delta x = \frac{4}{N}$ and $\Delta t = \frac{4}{N}$ to a final time of $T = 1$. Cut-cell discretization is used to handle the curved circle boundary, moving least squares are used for transfers, and reseeding is used to maintain particle coverage. Convergence plots are shown in the last row of Figure 12. In all tests the particle and grid error measured in L^2 norm reach the first order convergence. The convergence of L^∞_P error also approximately reach the first order but some outliers are observed in APIC. This occurred because particles were seeded so close to the boundary edge that some of their weights were near roundoff error. Since a quadratic spline is used, only one row of velocities is available, which is insufficient to reconstruct the full velocity. The MLS system in this situation is singular, which leads to an inaccurate transfer. We note that when transferred back to the grid, this particle will accurately interpolate velocity to the faces that are well-supported. The velocity interpolated to the nearly unsupported faces is inaccurate, but the weights are so small that this is irrelevant. This is why the grid velocity is clean. This problem could be avoided in a number of ways, including simply preventing particles from being so close to the domain boundaries.

5.1.4. Vortex shedding

In this test we simulate a constant velocity field $\langle 1,0\rangle$ passing a circle. We use a $[-2,14]\times[-4,4]$ domain, with slip boundary conditions on its $y = \pm 2$ sides, inflow ($u = 1$) at $x = -2$, and free surface on the $x = 14$ side. The circle is centered at $\langle 0,0\rangle$ with radius 0.25. The fluid has physical properties $\rho = 1$ and $\mu = 0$. Initially particles are sampled by blue noise in the domain. New particles are created to fill the vacancy as existing particles move in the $+x$ direction. Cut-cell discretization and reseeding are used for this test. The same test was run with APIC and XPIC(2,3,5) transfers. We run the tests to a cyclical state and then examine the frequency of vortex shedding. To extract the vortex signal, we compute vorticity inside a chosen window. This total vorticity ranges from positive (when a vortex with positive vorticity is centered in the window) to negative (when a vortex spinning the opposite way passes). This curve is oscillatory, and we compute its frequency using a Fourier transform. We show the simulation results in Figure 13. APIC sheds



(a) APIC

(b) XPIC(2)

(c) XPIC(3)

(d) XPIC(5)

Figure 13: Vortex shedding simulation with APIC and XPIC transfers. The red bars above and below the images indicate the range where vorticity is computed to extract vortex signal.

at frequency 0.49 Hz, and XPIC family sheds at frequency 0.39 Hz (XPIC(2)), 0.42 Hz (XPIC(3)), and 0.41 Hz (XPIC(5)). The corresponding Strouhal numbers are between 0.19 and 0.24. This is consistent with fluid flow over a range of higher Reynolds numbers $(300 - 10^6)$.

5.2. Dissipation and noise

5.2.1. Measuring vorticity

One of the main objectives of this work is to analyze how well vorticity is preserved under different transfers. To do this, we need useful measures of how well vorticity is preserved. We use two scalar measures

$$E_{vel} = \int_\Omega \|\mathbf{u}\|^2 \, dV \qquad\qquad E_{vort} = \int_\Omega \|\nabla \times \mathbf{u}\|^2 \, dV,$$

which measure the kinetic energy and magnitude of vorticity. We omit density and constants from the measures for convenience. We compute these on the grid from $\tilde{v}_{ia}^{n+1}$ and discretize them as

$$E_{vel} = \frac{1}{N_G} \sum_{ia} (\tilde{v}_{ia}^{n+1})^2 \qquad E_{vort} = \frac{1}{N_C} \sum_{c\alpha\beta} \left(\left(\frac{\partial u_\alpha}{\partial x_\beta} \right)_c - \left(\frac{\partial u_\beta}{\partial x_\alpha} \right)_c \right)^2 \qquad N_C = \sum_c 1,$$

where N_C is the number of cells; the index c runs over all cells that have sufficient neighboring information to compute the vorticity measure. The indices α, β run over the spatial dimensions. We normalize the discretized measures so that they do not depend on the resolution. The partial derivatives are approximated with central differences:

$$\left(\frac{\partial u_1}{\partial x_2}\right)_{(i,j)} = \frac{1}{2}\left(\frac{\tilde{v}^{n+1}_{i-\frac{1}{2},j+1} - \tilde{v}^{n+1}_{i-\frac{1}{2},j-1}}{2\Delta y} + \frac{\tilde{v}^{n+1}_{i+\frac{1}{2},j+1} - \tilde{v}^{n+1}_{i+\frac{1}{2},j-1}}{2\Delta y}\right)$$

$$\left(\frac{\partial u_2}{\partial x_1}\right)_{(i,j)} = \frac{1}{2}\left(\frac{\tilde{v}^{n+1}_{i+1,j-\frac{1}{2}} - \tilde{v}^{n+1}_{i-1,j-\frac{1}{2}}}{2\Delta x} + \frac{\tilde{v}^{n+1}_{i+1,j+\frac{1}{2}} - \tilde{v}^{n+1}_{i-1,j+\frac{1}{2}}}{2\Delta x}\right)$$

The same four-point central-differenced and central-averaged stencil is also used in 3D.

5.2.2. Taylor Green vortex

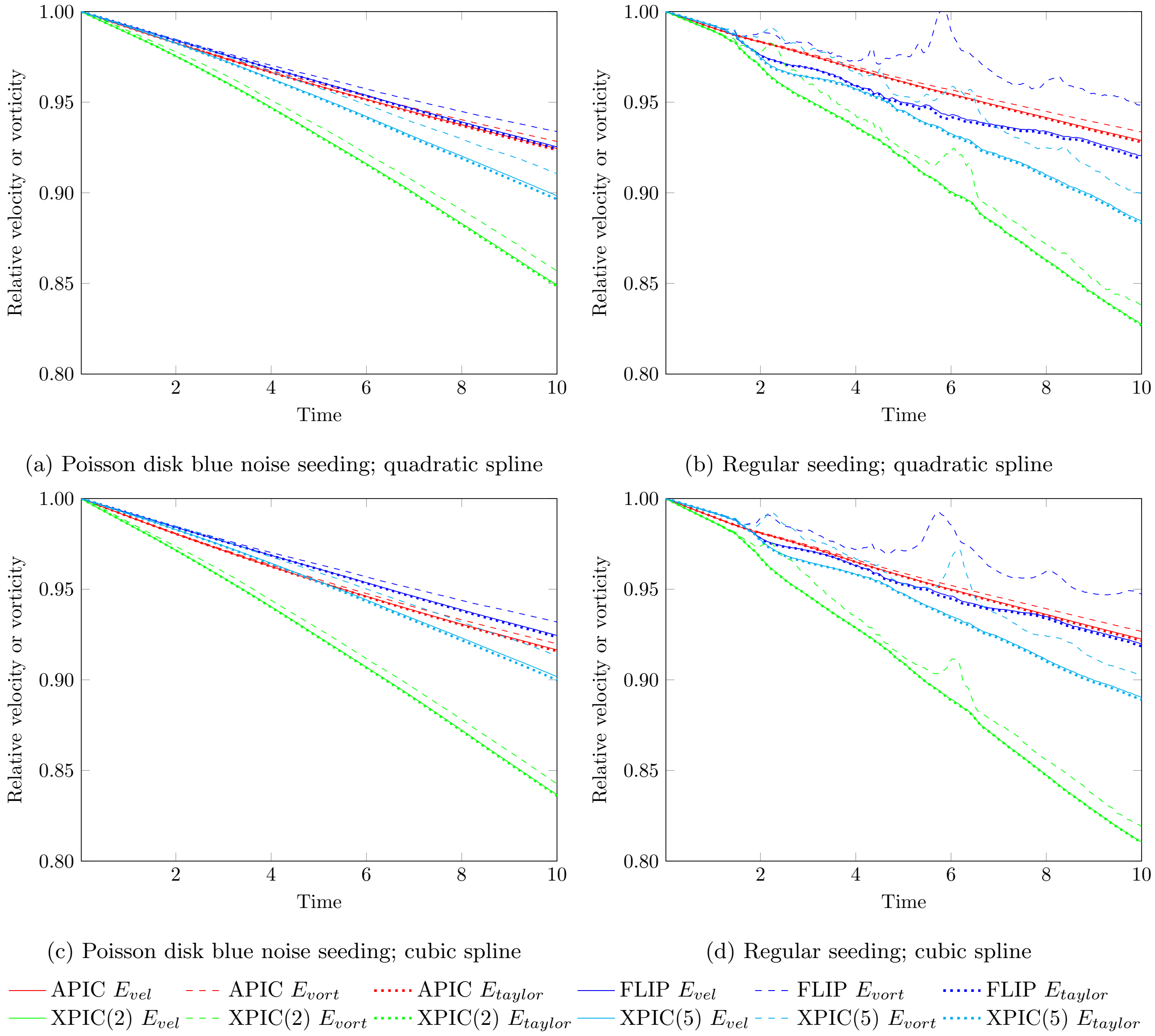
For this example, we begin with the basic setup from Section 5.1.1. For this section, we fix $\mu = 0$ and $N = 64$. We also use a later final time $T = 10$ to observe the longer-term behavior. In this test, we are interested in studying (a) dissipation of energy, (b) transfer of energy into incorrect velocity modes, (c) loss of vorticity, (d) the effects of particle seeding, and (e) the effects of spline choice (quadratic or cubic).

We test APIC, FLIP and XPIC (order 2 and 5) using Poisson disk seeding (4 particles per cell on average) and regular seeding (2×2 particles per grid cell). PIC is omitted from this test since it dissipates energy too rapidly to draw an interesting comparison. The results are shown in Figure 14. In the figure, the measures E_{vel}, E_{vort}, and E_{taylor} are normalized by their values after the first transfer from particles to grid. The measure E_{taylor} is like E_{vel}, except only contributions from the Taylor-Green Fourier modes are included.

There are a few interesting observations to be made from the results. FLIP transfers are not affected much by the choice of spline order, but it is sensitive to the particle distribution. APIC is relatively insensitive to the spline and seeding, though the higher-order spline and irregular seeding both increase dissipation very slightly. APIC and FLIP have similar levels of dissipation. XPIC is also sensitive to the particle distribution as FLIP, and the XPIC transfer with lower order shows relatively larger dissipation.

Noting the analysis of the prior sections, we perform an FFT on the velocity field and report the magnitudes of the velocity modes as a colored image as we did for the transfer modes. In Figure 15, we look at the bleeding of the vortex into other Fourier modes. From this we can see that APIC is bleeding mostly into nearby low-frequency velocity modes (near the center of the image), wheres FLIP and XPIC transfer energy into higher frequency modes. The error visible after the very first transfer for FLIP and XPIC with irregular seeding is caused by trying to represent the velocity field on particles; if the APIC particles are initialized with $\mathbf{b}^0_{pa} = \mathbf{0}$, the same errors are observed.

In Figure 15, significant portion of the velocity field can be found in incorrect high-frequency Fourier modes (modes on the order of a few percent). The quantity $E_{vel} - E_{taylor}$ for Figure 14 reflects the amount of kinetic energy that has been transferred to incorrect Fourier modes, and here the difference appears negligible. Because kinetic energy measures the squares of velocity, modest velocity errors (e.g., on the order



(a) Poisson disk blue noise seeding; quadratic spline

(b) Regular seeding; quadratic spline

(c) Poisson disk blue noise seeding; cubic spline

(d) Regular seeding; cubic spline

Figure 14: Loss in energy (solid lines) and vorticity (dashed lines) for a Taylor-Green vortex over time using FLIP, APIC and XPIC transfers. The dotted lines show the amount of energy in the Fourier modes corresponding to the Taylor-Green vortex. All curves are normalized relative to the values obtained after transferring from the particles to the grid in the first time step.

of 5%) in modes that should be zero make only a very small difference in energy (around 0.25%). Since positions are updated using velocities (not their square), these errors are still significant.

We observe that regular seeding and irregular seeding produce noticeably different results on this test. Regular seeding introduces leakage that is several times higher than for irregular seeding. (Green pixels are observed well away from the middle of the image when regular seeding is used, indicating energy leakage into high-frequency Fourier modes. For irregular seeding, only shades of blue are observed away from the low-frequency modes in the middle.) A highly regular particle distribution appears to exacerbate this bleeding.

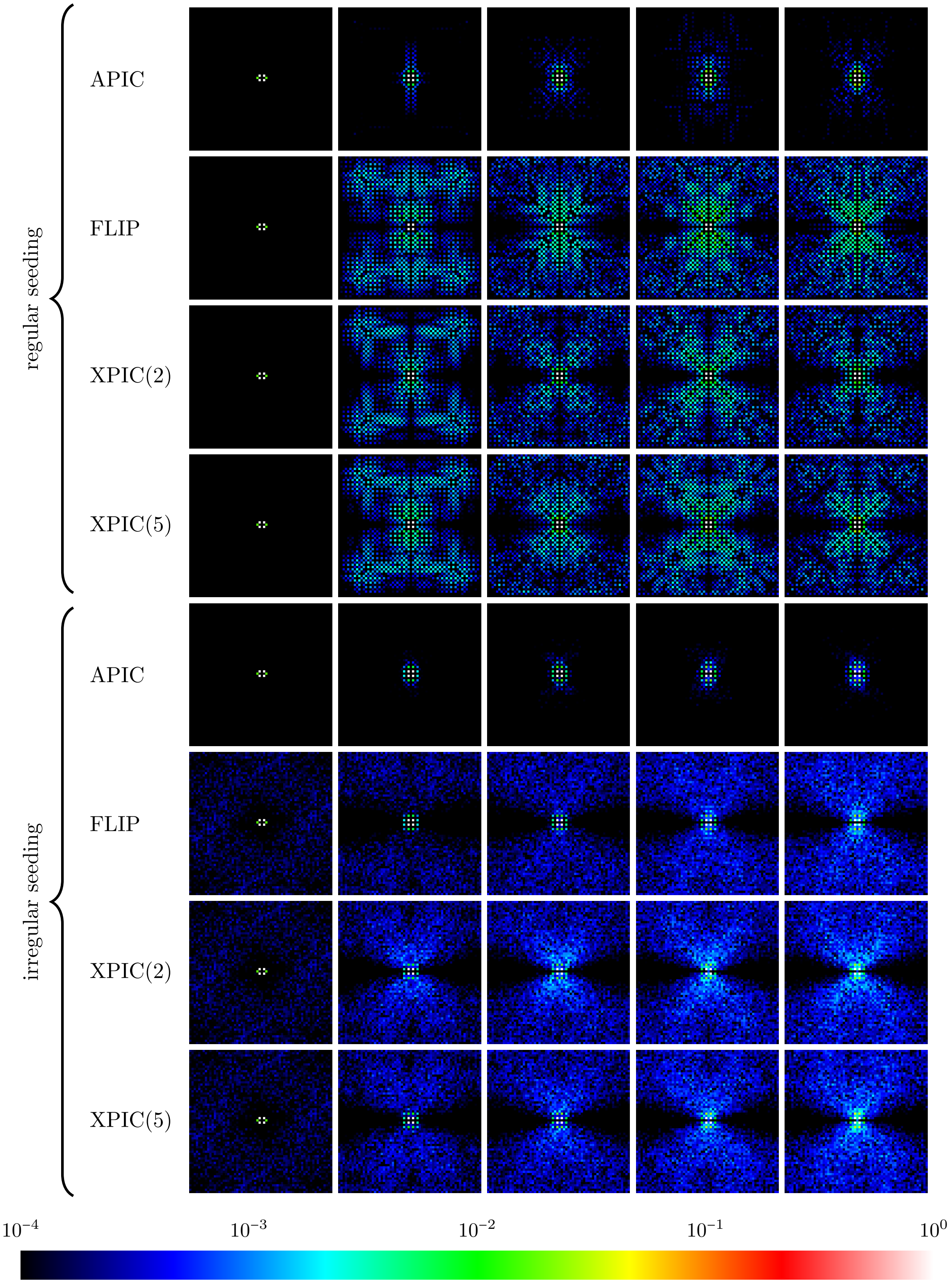


Figure 15: Energy leakage into other Fourier modes, at times $t = 0, 2, 4, 6, 8$, using regular or irregular seeding with 4 particles per cell. The first image is immediately after the initial particle to grid transfer.

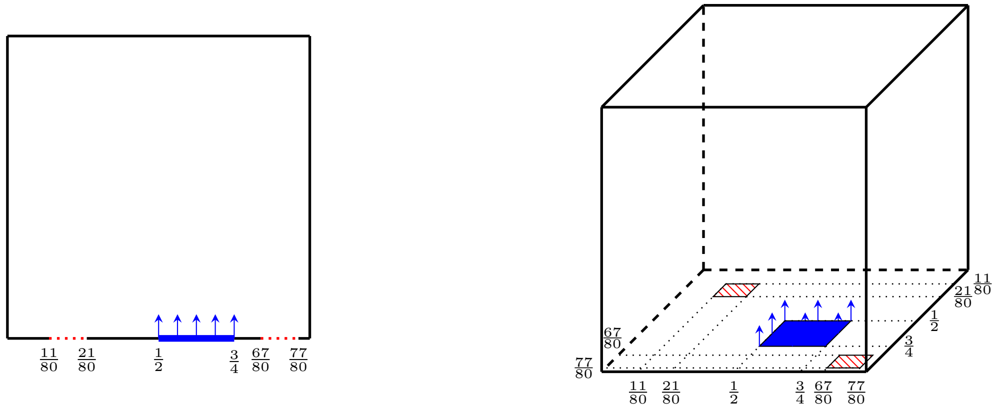


Figure 16: Inlet test configurations for 2D and 3D. The simulation domain is $[0,1]^d$ where $d = 2$ or $d = 3$. The red dotted or hatched areas are free surfaces, the blue areas are sources with a normal velocity $v = 0.2$, and all other boundaries are slip and have 0 normal velocity.

5.2.3. Inlet

In this test we use a $[0,1]^d$ domain, where $d = 2$ for 2D and $d = 3$ for 3D. All boundaries are slip except some portions of the $y = 0$ boundary, where we place sources (fixed inflow velocity $v = 0.2$) and sinks (outflow, $p = 0$). The layouts are shown in Figure 16. At time $T = 80\,s$ we turn off the source and sinks (enforcing slip boundary conditions everywhere) and observe how energy is dissipated until time $T = 200\,s$.

We compute kinetic energy for particles and grid. The kinetic for grid is discretized as

$$\mathcal{KE}^G = \sum_{ia} \frac{1}{2} m_{ia} (\tilde{v}_{ia}^{n+1})^2$$

where the indices run through all internal faces with non-zero mass. The kinetic energy for particles are discretized as

$$\mathcal{KE}^P = \sum_{p} \frac{1}{2} m_p \|\mathbf{v}_p^n\|^2 + \sum_{pa} \frac{1}{2} m_p (\mathbf{b}_{pa}^n \cdot (\mathbf{D}_{pa}^n)^{-1} \mathbf{b}_{pa}^n)$$

The extra contribution to kinetic energy is an estimate to the affine contributions; it is omitted for non-APIC transfers. The vorticity energy is computed as

$$\mathcal{V} = \sum_{p} \frac{1}{2} m_p \left\| \sum_{ia} \nabla w_{ip}^n \times v_{ia}^n \mathbf{e}_a \right\|^2 .$$

Snapshots of 2D simulations are shown in Figure 17. After the sources are turned off ($T > 80$), two vortices are formed in APIC and FLIP. In Figures 18 and 19 we plot the vorticity and kinetic energy as a function of time. Observe that the particle and grid energy closely track each other in 2D for all versions as well as in 3D for PIC and APIC. In the 3D FLIP simulation, we can observe a significant difference between the grid and particle kinetic energy. The particle energy grows while the inlet is open even though the grid energy remains stable. Although grid velocities decay for all 3D simulations, the particle energy and

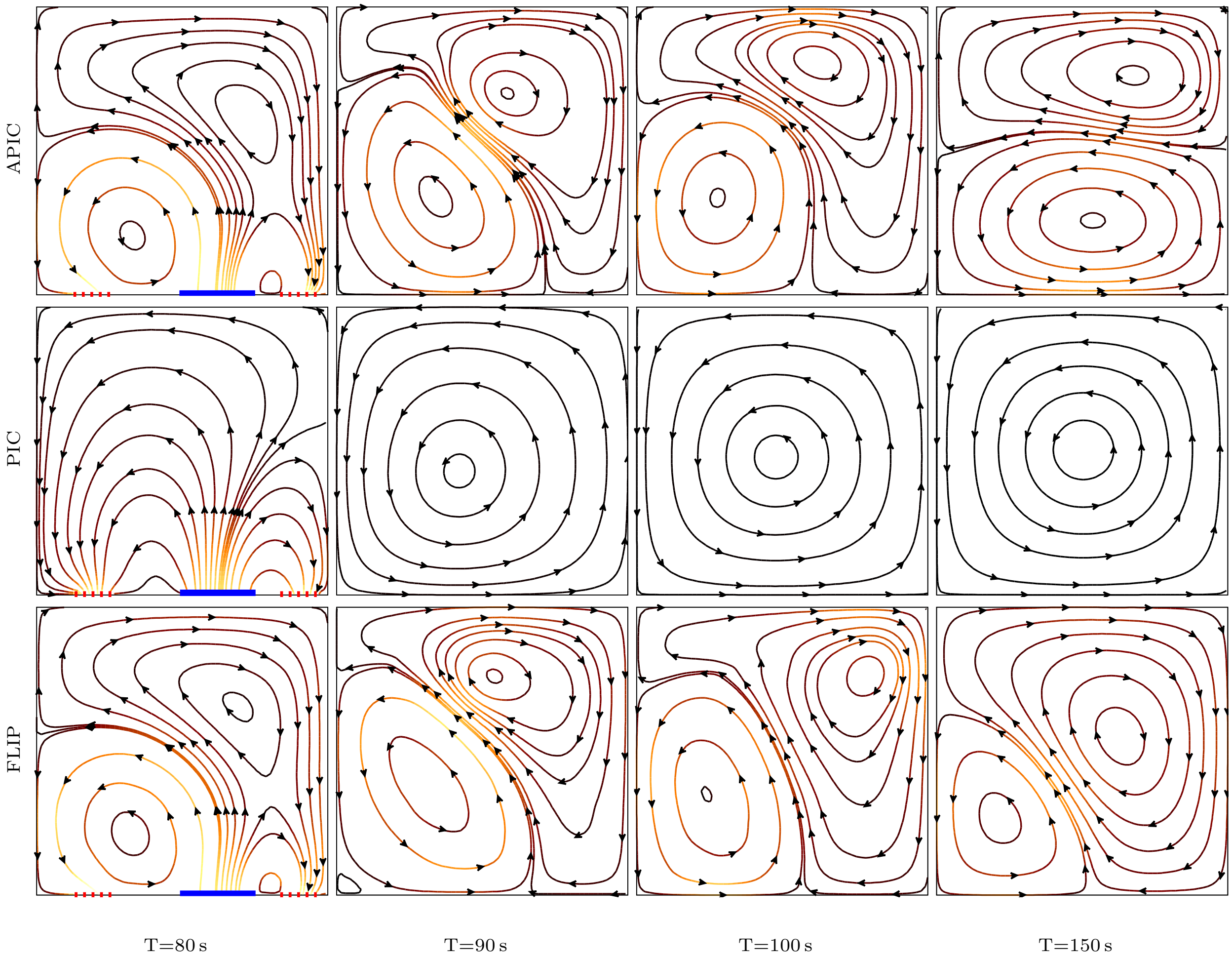


Figure 17: Snapshots of inlet tests. Streamline colors indicate fluid velocity magnitude, with black indicating slow fluid and yellow representing the fastest flow. The first column of frames ($T = 80\,s$) captures the last moment before we seal the boundary. The source and sinks are marked by solid blue and dotted red as same as Figure 16. After that vortices evolve without any input.

particle-based vorticity does not decay to zero for the 3D FLIP simulation. Even when the particles come effectively to rest, the particles carry non-negligible velocities.

6. Conclusions

We have presented a new MAC-grid-based APIC transfer that preserves linear and angular momentum and also satisfies the original APIC properties. The full scheme is not conservative, since we perform the pressure solve using constant density as a compromise to avoid *boiling* artefacts.

We used 2D Fourier transforms to understand the numerical properties of the transfer. Compared with the 1D Fourier transform currently being used to analyze transforms, Fourier transforms in 2D give us some

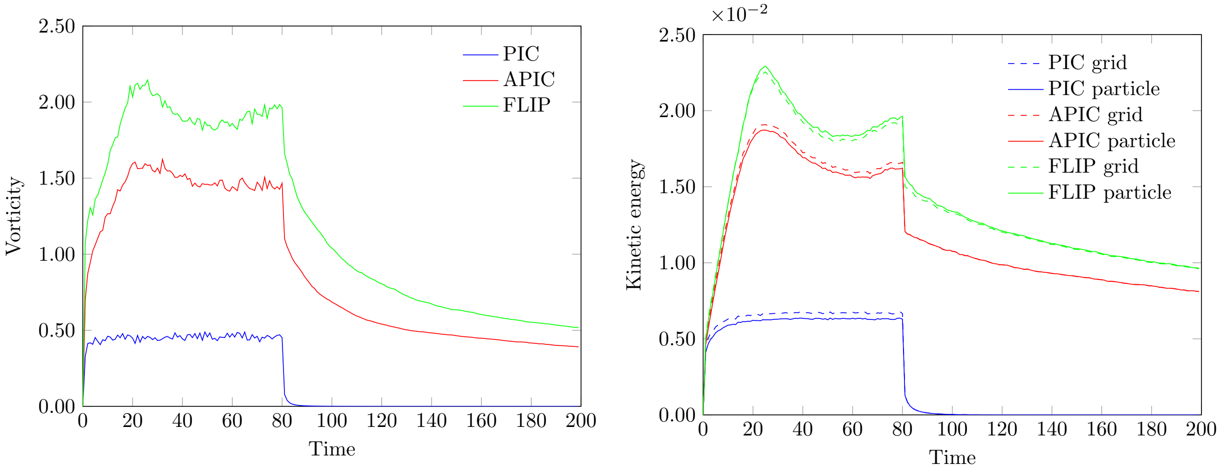


Figure 18: Vorticity and kinetic energy of 2D inlet test. Fluid is pumped through the domain until time 80 s, at which point the domain boundary is sealed and the fluid continues to circulate. For PIC, little energy is accumulated, and the circulation decays rapidly. For both FLIP and APIC, the circulation drops off quickly but then levels off. The grid and particle kinetic energy track each other closely for all three methods, which suggests that FLIP is quite stable on this example. FLIP retains more energy throughout the simulation.

advantages. The first advantage is that we are able to analyze the Taylor-Green vortex, which gives us a way of studying dissipation of vortices. The second advantage is that it allows us to include pressure projection in the analysis, which extends the analysis meaningfully to include FLIP transfers. Studying in two dimensions also lets us studying the transfers' sensitivity to different particle distributions rather than merely irregularities in particle spacing.

Compared with direct computation of the eigenvalues of the transfer matrix, the 2D Fourier transform lets us efficiently compute and intuitively understand the eigenvalues of transfers. It arranges the eigenvalues by giving us a meaningful image rather than a long list of eigenvalues. From these images we can tell how a vortex of a spatial scale dissipates for example. Finally the Fourier transform in 2D provides us images for various transfers so we can compare them conveniently and visually.

In terms of dissipation, the comparison between APIC and PIC is not a surprise; PIC is very dissipative. The comparison with XPIC is instructive, as this is the first direct comparison between the two transfers as far as we are aware. The level of dissipation in APIC lies between XPIC with order 2 and 3. XPIC becomes less dissipative with higher order. In this spectrum, FLIP has zero dissipation. The opposite side of dissipation is noise, where modes survive on particles but should not. On this side, APIC and PIC are effectively perfect. XPIC is not too bad at lower orders, but it does tend to retain divergent velocity modes on particles over short time scales (See Figure 8). Like FLIP, it also tends to transfer low-frequency velocity modes into higher-frequency velocity modes (See Figure 15).

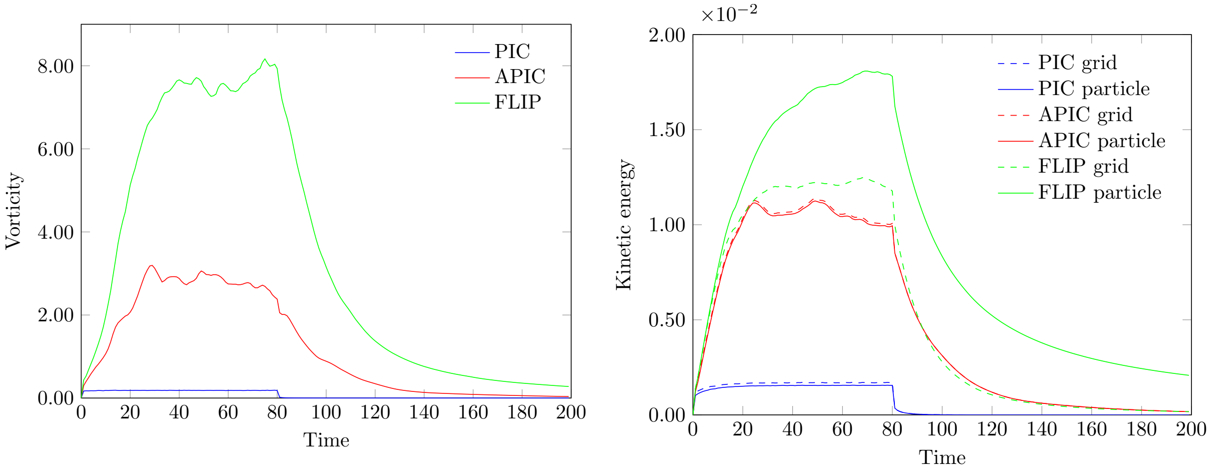


Figure 19: Vorticity and kinetic energy of 3D inlet test. Fluid is pumped through the domain until time 80 s, at which point the domain boundary is sealed and the fluid continues to circulate. The grid-based kinetic energy for all three methods decay to zero. For PIC and APIC, particle energy tracks the grid energy, and the particle-based vorticity decays to zero. The behavior of FLIP is very different. Particle energy is significantly greater than grid energy throughout the simulation. At its peak, about 1/3 of the particle energy does not transfer to the grid. During the pumped phase, energy accumulates on particles but not on the grid. As the grid energy decays away to zero, a significant amount of particle energy remains. The particle-based vorticity measure decays more than the energy, but it also stops short of zero. Since the vorticity measure is most sensitive to changes on the length scale of one grid cell, this suggests that the particles end with velocity modes whose wavelength is significantly less than the cell size.

6.1. Limitations

The analysis presented in this paper provides intuition for how transfers behave; it does not fully characterize the transfers. The use of Fourier analysis limits the analysis to using a globally regular particle arrangement, which may skew the analysis. Truly irregular particle configurations may behave somewhat differently, and they may be more or less dissipative than the tiled case. Nevertheless, the analysis presented provides useful insight into the methods involved.

The APIC transfers introduced are linear and angular momentum conserving, but the overall algorithm is not. This is because we found it necessary to use a constant-density pressure projection to avoid boiling artefacts in the simulations, where areas with thinner particle coverage appear less dense and rise under gravity. We postpone the problem of achieving full conservation of linear and angular momentum for future work. For related reasons, we postpone consideration of free surface flows for future work.

The desirable properties of the APIC transfers are tied to position update employed. As with co-located transfers, a generalized version of the transfers [2] may significantly broaden the range of positional updates over which good transfer properties may be obtained. In particular, a version of the transfers compatible with the XPIC position update (likely a MAC version of [2]) would be desirable.

The algorithm presented is not observed to work well with free surfaces. There are two reasons for this.

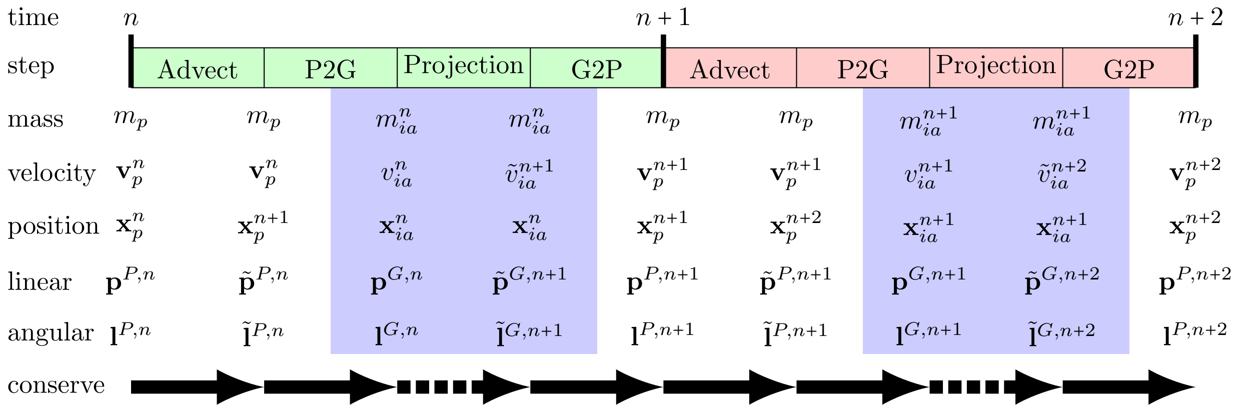

time n $n+1$ $n+2$
step Advect P2G Projection G2P Advect P2G Projection G2P
mass m_p m_p m_{ia}^n m_{ia}^n m_p m_p m_{ia}^{n+1} m_{ia}^{n+1} m_p
velocity $\mathbf{v}_p^n$ $\mathbf{v}_p^n$ v_{ia}^n $\tilde{v}_{ia}^{n+1}$ $\mathbf{v}_p^{n+1}$ $\mathbf{v}_p^{n+1}$ v_{ia}^{n+1} $\tilde{v}_{ia}^{n+2}$ $\mathbf{v}_p^{n+2}$
position $\mathbf{x}_p^n$ $\mathbf{x}_p^{n+1}$ $\mathbf{x}_{ia}^n$ $\mathbf{x}_{ia}^n$ $\mathbf{x}_p^{n+1}$ $\mathbf{x}_p^{n+2}$ $\mathbf{x}_{ia}^{n+1}$ $\mathbf{x}_{ia}^{n+1}$ $\mathbf{x}_p^{n+2}$
linear $\mathbf{p}^{P,n}$ $\tilde{\mathbf{p}}^{P,n}$ $\mathbf{p}^{G,n}$ $\tilde{\mathbf{p}}^{G,n+1}$ $\mathbf{p}^{P,n+1}$ $\tilde{\mathbf{p}}^{P,n+1}$ $\mathbf{p}^{G,n+1}$ $\tilde{\mathbf{p}}^{G,n+2}$ $\mathbf{p}^{P,n+2}$
angular $\mathbf{l}^{P,n}$ $\tilde{\mathbf{l}}^{P,n}$ $\mathbf{l}^{G,n}$ $\tilde{\mathbf{l}}^{G,n+1}$ $\mathbf{l}^{P,n+1}$ $\tilde{\mathbf{l}}^{P,n+1}$ $\mathbf{l}^{G,n+1}$ $\tilde{\mathbf{l}}^{G,n+2}$ $\mathbf{l}^{P,n+2}$
conserve

Figure 20: The proposed time integration scheme can be divided into four stages: advection, particle-to-grid transfer, pressure projection, and grid-to-particle transfer. Two full time steps are shown (first step is green, second step is red). After each stage, the state (mass, velocity, and position) are represented differently. Corresponding to each state is a corresponding measure of linear momentum and angular momentum. The transitions shown with solid arrows are conserved, as proved in Section 7.1. The transitions shown with dotted arrows are conserved under a *different* definition of momentum and angular momentum.

The first is that our pressure projection is performed with constant density. Isolated particles command a larger volume on the grid than particles in the bulk, which causes problems when escaped particles land on the fluid surface. Particle-deficient pockets are created when fluid regions merge. This is a well-known problem with MPM, and some approaches have been proposed to address it [33, 34].

7. Appendix A: Analysis

7.1. APIC properties

The properties of our MAC APIC transfers are similar to those of the co-located method [2]. The analysis of the method as presented is notationally complicated compared to the original since the position update is being delayed until the beginning of the next time step.

In this section, we demonstrate that the proposed APIC transfers satisfy the same properties as the original co-located APIC transfers: conservation of linear and angular momentum, preservation of affine velocity fields, and single particle stability.

The proposed scheme conserves momentum and angular momentum in a local sense. Total momentum is conserved, and momentum is only transferred to nearby neighbors (limited by the interpolation stencil size). Due to this interpolation, the transfers will exhibit a degree of momentum diffusion. Since the proposed transfers are not flux-based, it is unlikely that they could be used to track shocks. This is not a problem for incompressible flow (which never exhibits shocks), but this may be a limitation in some applications.

7.1.1. Linear momentum

We can define particle-based and grid-based measures of momentum in the usual way.

$$\mathbf{p}^{P,n} = \tilde{\mathbf{p}}^{P,n} = \sum_p m_p \mathbf{v}_p^n \qquad \mathbf{p}^{G,n} = \sum_{ia} m_{ia}^n v_{ia}^n \mathbf{e}_a \qquad \tilde{\mathbf{p}}^{G,n+1} = \sum_{ia} m_{ia}^n \tilde{v}_{ia}^{n+1} \mathbf{e}_a \tag{17}$$

With these definitions, momentum is conserved across a particle-to-grid transfer, since

$$\begin{aligned}
\mathbf{p}^{G,n} &= \sum_{ia} m_{ia}^n v_{ia}^n \mathbf{e}_a = \sum_{ia} \left(\sum_p w_{ipa}^n m_p \mathbf{e}_a^T \mathbf{v}_p^n + \sum_p w_{ipa}^n m_p (\mathbf{b}_{pa}^n)^T (\mathbf{D}_{pa}^n)^{-1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \right) \mathbf{e}_a \\
&= \sum_p m_p \mathbf{v}_p^n \sum_a \mathbf{e}_a \mathbf{e}_a^T \sum_i w_{ipa}^n + \sum_{ap} m_p (\mathbf{b}_{pa}^n)^T (\mathbf{D}_{pa}^n)^{-1} \mathbf{e}_a \sum_i w_{ipa}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \\
&= \sum_p m_p \mathbf{v}_p^n = \mathbf{p}^{P,n}
\end{aligned}$$

The transfer from the grid back to the particle also conserves momentum, since

$$\mathbf{p}^{P,n+1} = \sum_p m_p \mathbf{v}_p^{n+1} = \sum_p m_p \sum_{ia} w_{ipa}^n \tilde{v}_{ia}^{n+1} \mathbf{e}_a = \sum_{ia} \left(\sum_p m_p w_{ipa}^n \right) \tilde{v}_{ia}^{n+1} \mathbf{e}_a = \sum_{ia} m_{ia}^n \tilde{v}_{ia}^{n+1} \mathbf{e}_a = \tilde{\mathbf{p}}^{G,n+1}$$

7.1.2. Angular momentum

We can define the following particle-based and grid-based measures of angular momentum.

$$\mathbf{l}^{P,n} = \sum_p \mathbf{x}_p^n \times m_p \mathbf{v}_p^n + \sum_{ap} m_p \mathbf{b}_{pa}^n \times \mathbf{e}_a \qquad \mathbf{l}^{G,n} = \sum_{ia} \mathbf{x}_{ia}^n \times m_{ia}^n v_{ia}^n \mathbf{e}_a \tag{18}$$

$$\tilde{\mathbf{l}}^{P,n} = \sum_p \mathbf{x}_p^{n+1} \times m_p \mathbf{v}_p^n + \sum_{ap} m_p \mathbf{b}_{pa}^n \times \mathbf{e}_a \qquad \tilde{\mathbf{l}}^{G,n+1} = \sum_{ia} \tilde{\mathbf{x}}_{ia}^{n+1} \times m_{ia}^n \tilde{v}_{ia}^{n+1} \mathbf{e}_a \tag{19}$$

With these definitions, angular momentum is conserved across a particle-to-grid transfer, since

$$\begin{aligned}
\mathbf{l}^{G,n} &= \sum_{ia} \mathbf{x}_{ia}^n \times m_{ia}^n v_{ia}^n \mathbf{e}_a \\
&= \sum_{ia} \mathbf{x}_{ia}^n \times \mathbf{e}_a \left(\sum_p w_{ipa}^n m_p \mathbf{e}_a^T \mathbf{v}_p^n + \sum_p w_{ipa}^n m_p (\mathbf{b}_{pa}^n)^T (\mathbf{D}_{pa}^n)^{-1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \right) \\
&= \sum_{ia} \mathbf{x}_{ia}^n \times \mathbf{e}_a \sum_p w_{ipa}^n m_p \mathbf{e}_a^T \mathbf{v}_p^n + \sum_{ia} \mathbf{x}_{ia}^n \times \mathbf{e}_a \sum_p w_{ipa}^n m_p (\mathbf{b}_{pa}^n)^T (\mathbf{D}_{pa}^n)^{-1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \\
&= \sum_{pa} \left(\sum_i w_{ipa}^n \mathbf{x}_{ia}^n \right) \times \mathbf{e}_a m_p \mathbf{e}_a^T \mathbf{v}_p^n + \sum_{pa} m_p \mathbf{e}_a^{*T} \left(\sum_i \mathbf{x}_{ia}^n w_{ipa}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})^T \right) (\mathbf{D}_{pa}^n)^{-1} \mathbf{b}_{pa}^n \\
&= \sum_{pa} \mathbf{x}_p^{n+1} \times \mathbf{e}_a m_p \mathbf{e}_a^T \mathbf{v}_p^n + \sum_{pa} m_p \mathbf{e}_a^{*T} \mathbf{D}_{pa}^n (\mathbf{D}_{pa}^n)^{-1} \mathbf{b}_{pa}^n \\
&= \sum_p m_p (\mathbf{x}_p^{n+1})^* \left(\sum_a \mathbf{e}_a \mathbf{e}_a^T \right) \mathbf{v}_p^n + \sum_{pa} m_p \mathbf{e}_a^{*T} \mathbf{b}_{pa}^n \\
&= \sum_p \mathbf{x}_p^{n+1} \times m_p \mathbf{v}_p^n + \sum_{ap} m_p \mathbf{b}_{pa}^n \times \mathbf{e}_a = \tilde{\mathbf{l}}^{P,n}
\end{aligned}$$

The transfer from the grid back to the particle also conserves angular momentum, since

$$\begin{aligned}
\mathbf{l}^{P,n+1} &= \sum_p \mathbf{x}_p^{n+1} \times m_p \mathbf{v}_p^{n+1} + \sum_{ap} m_p \mathbf{b}_{pa}^{n+1} \times \mathbf{e}_a \\
&= \sum_p \mathbf{x}_p^{n+1} \times m_p \sum_{ia} w_{ipa}^n \tilde{v}_{ia}^{n+1} \mathbf{e}_a + \sum_{ap} m_p \sum_i w_{ipa}^n \tilde{v}_{ia}^{n+1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \times \mathbf{e}_a \\
&= \sum_{ipa} m_p w_{ipa}^n \tilde{v}_{ia}^{n+1} \big(\mathbf{x}_p^{n+1} + (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})\big) \times \mathbf{e}_a \\
&= \sum_{ia} \left(\sum_p m_p w_{ipa}^n \right) \tilde{v}_{ia}^{n+1} \mathbf{x}_{ia}^n \times \mathbf{e}_a \\
&= \sum_{ia} \mathbf{x}_{ia}^n \times m_{ia}^n \tilde{v}_{ia}^{n+1} \mathbf{e}_a \\
&= \sum_{ia} (\tilde{\mathbf{x}}_{ia}^{n+1} - \Delta t \tilde{v}_{ia}^{n+1} \mathbf{e}_a) \times m_{ia}^n \tilde{v}_{ia}^{n+1} \mathbf{e}_a \\
&= \tilde{\mathbf{l}}^{G,n+1} - \Delta t \sum_{ia} \tilde{v}_{ia}^{n+1} \mathbf{e}_a \times m_{ia}^n \tilde{v}_{ia}^{n+1} \mathbf{e}_a = \tilde{\mathbf{l}}^{G,n+1}
\end{aligned}$$

7.1.3. Affine

The APIC affine property is that an affine velocity field is preserved across transfers between particles and grid when particles are not moved ($\Delta t = 0$). Since $\Delta t = 0$, we have $\mathbf{x}_{ia} = \tilde{\mathbf{x}}_{ia}$ (since nothing is moving). We can also ignore superscripts, since time does not matter. Let $f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$ define an arbitrary affine function. We begin by assuming that the grid velocity is described by this function ($\tilde{v}_{ia} = \mathbf{e}_a^T (\mathbf{A}\mathbf{x}_{ia} + \mathbf{b})$) and transferring to particles ($\tilde{v}_{ia} \to \{\mathbf{v}_p, \mathbf{b}_{pa}\}$).

$$\begin{aligned}
\mathbf{v}_p &= \sum_{ia} w_{ipa} \tilde{v}_{ia} \mathbf{e}_a = \sum_{ia} w_{ipa} \mathbf{e}_a \mathbf{e}_a^T (\mathbf{A}\mathbf{x}_{ia} + \mathbf{b}) = \left(\sum_a \mathbf{e}_a \mathbf{e}_a^T \right) (\mathbf{A}\mathbf{x}_p + \mathbf{b}) = \mathbf{A}\mathbf{x}_p + \mathbf{b} \\
\mathbf{b}_{pa} &= \sum_i w_{ipa} (\mathbf{x}_{ia} - \mathbf{x}_p) \tilde{v}_{ia} = \left(\sum_i w_{ipa} (\mathbf{x}_{ia} - \mathbf{x}_p)(\mathbf{x}_{ia})^T \right) \mathbf{A}^T \mathbf{e}_a + \left(\sum_i w_{ipa} (\mathbf{x}_{ia} - \mathbf{x}_p) \right) \mathbf{e}_a^T \mathbf{b} = \mathbf{D}_p \mathbf{A}^T \mathbf{e}_a
\end{aligned}$$

From this we can see the corresponding representation of this affine velocity field on particles. Using these particle values and transferring back to the grid ($\{\mathbf{v}_p, \mathbf{b}_{pa}\} \to v_{ia}$) yields.

$$\begin{aligned}
m_{ia} v_{ia} &= \sum_p w_{ipa} m_p \mathbf{e}_a^T \mathbf{v}_p + \sum_p w_{ipa} m_p (\mathbf{b}_{pa})^T (\mathbf{D}_{pa})^{-1} (\mathbf{x}_{ia} - \mathbf{x}_p) \\
&= \sum_p w_{ipa} m_p \mathbf{e}_a^T (\mathbf{A}\mathbf{x}_p + \mathbf{b}) + \sum_p w_{ipa} m_p (\mathbf{D}_p \mathbf{A}^T \mathbf{e}_a)^T (\mathbf{D}_{pa})^{-1} (\mathbf{x}_{ia} - \mathbf{x}_p) \\
&= \sum_p w_{ipa} m_p \mathbf{e}_a^T (\mathbf{A}\mathbf{x}_p + \mathbf{b}) + \sum_p w_{ipa} m_p \mathbf{e}_a^T \mathbf{A} (\mathbf{x}_{ia} - \mathbf{x}_p) \\
&= \sum_p w_{ipa} m_p \mathbf{e}_a^T (\mathbf{A}\mathbf{x}_p + \mathbf{b} + \mathbf{A}\mathbf{x}_{ia} - \mathbf{A}\mathbf{x}_p) \\
&= \left(\sum_p w_{ipa} m_p \right) \mathbf{e}_a^T (\mathbf{A}\mathbf{x}_{ia} + \mathbf{b}) = m_{ia} \mathbf{e}_a^T (\mathbf{A}\mathbf{x}_{ia} + \mathbf{b}) = m_{ia} \tilde{v}_{ia}
\end{aligned}$$

Since $v_{ia} = \tilde{v}_{ia}$, the velocity field obtained by transferring from grid to particles and back to grid matches the original velocity field.

7.1.4. Stability

The final APIC property is the stability criterion, which requires that a single particle translating in the absence of forces ($\tilde{v}_{ia}^{n+1} = v_{ia}^{n}$) should translate with no change to velocity or affine momentum state ($\mathbf{v}_p^{n+1} = \mathbf{v}_p^n$, $\mathbf{b}_{pa}^{n+1} = \mathbf{b}_{pa}^n$, and $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{v}_p^n$). The translation requirement is trivially satisfied. Starting from particles, we first compute the grid-based quantities. Note that summation on particles is omitted since there is only one particle.

$$\begin{aligned}
\mathbf{x}_p^{n+1} &= \mathbf{x}_p^n + \Delta t \mathbf{v}_p^n \\
m_{ia}^n &= w_{ipa}^n m_p \\
m_{ia}^n v_{ia}^n &= w_{ipa}^n m_p \mathbf{e}_a^T \mathbf{v}_p^n + w_{ipa}^n m_p (\mathbf{b}_{pa}^n)^T (\mathbf{D}_{pa}^n)^{-1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \\
v_{ia}^n &= \mathbf{e}_a^T \mathbf{v}_p^n + (\mathbf{b}_{pa}^n)^T (\mathbf{D}_{pa}^n)^{-1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})
\end{aligned}$$

We can see that velocities are unchanged since

$$\begin{aligned}
\sum_{ia} w_{ipa}^n v_{ia}^n \mathbf{e}_a &= \sum_{ia} w_{ipa}^n \mathbf{e}_a \left(\mathbf{e}_a^T \mathbf{v}_p^n + (\mathbf{b}_{pa}^n)^T (\mathbf{D}_{pa}^n)^{-1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \right) \\
&= \left(\sum_a \mathbf{e}_a \mathbf{e}_a^T \right) \mathbf{v}_p^n + \sum_a \mathbf{e}_a (\mathbf{b}_{pa}^n)^T (\mathbf{D}_{pa}^n)^{-1} \sum_i w_{ipa}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) = \mathbf{v}_p^n \\
\mathbf{v}_p^{n+1} &= \sum_{ia} w_{ipa}^n \tilde{v}_{ia}^{n+1} \mathbf{e}_a = \sum_{ia} w_{ipa}^n v_{ia}^n \mathbf{e}_a = \mathbf{v}_p^n
\end{aligned}$$

Finally, affine momentum is unchanged since

$$\begin{aligned}
\mathbf{b}_{pa}^{n+1} &= \sum_i w_{ipa}^n \tilde{v}_{ia}^{n+1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) = \sum_i w_{ipa}^n v_{ia}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \\
&= \sum_i w_{ipa}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \left(\mathbf{e}_a^T \mathbf{v}_p^n + (\mathbf{b}_{pa}^n)^T (\mathbf{D}_{pa}^n)^{-1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \right) \\
&= \left(\sum_i w_{ipa}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \right) \mathbf{e}_a^T \mathbf{v}_p^n + \sum_i w_{ipa}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) (\mathbf{b}_{pa}^n)^T (\mathbf{D}_{pa}^n)^{-1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \\
&= \left(\sum_i w_{ipa}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})^T \right) (\mathbf{D}_{pa}^n)^{-1} \mathbf{b}_{pa}^n = \mathbf{D}_{pa}^n (\mathbf{D}_{pa}^n)^{-1} \mathbf{b}_{pa}^n = \mathbf{b}_{pa}^n
\end{aligned}$$

This establishes the one-particle stability criterion.

Acknowledgements

This work was funded partially through an initial funding package from the University of California Riverside. Ounan Ding was partially funded through a Graduate Division Fellowship from the University of California Riverside.

References

[1] C. Jiang, C. Schroeder, A. Selle, J. Teran, A. Stomakhin, The affine particle-in-cell method, ACM Trans Graph 34 (4) (2015) 51:1–51:10.

[2] C. Jiang, C. Schroeder, J. Teran, An angular momentum conserving affine-particle-in-cell method, J Comp Phys 338 (2017) 137–164.

[3] Y. Grigoryev, V. Vshivkov, M. Fedoruk, Numerical Particle-In-Cell Methods: Theory and Applications, Walter de Gruyter, 2002.

[4] F. Harlow, The particle-in-cell method for numerical solution of problems in fluid dynamics, Meth Comp Phys 3 (1964) 319–343.

[5] S. Bardenhagen, E. Kober, The generalized interpolation material point method, Comp Mod in Eng and Sci 5 (6) (2004) 477–496.

[6] M. Steffen, R. M. Kirby, M. Berzins, Analysis and reduction of quadrature errors in the material point method (MPM), Int J Numer Meth Eng 76 (6) (2008) 922–948.

[7] J. Brackbill, H. Ruppel, FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions, J Comp Phys 65 (1986) 314–343.

[8] J. Brackbill, D. Kothe, H. Ruppel, FLIP: A low-dissipation, PIC method for fluid flow, Comp Phys Comm 48 (1988) 25–38.

[9] J. A. Nairn, Numerical simulation of orthogonal cutting using the material point method, Engineering Fracture Mechanics 149 (2015) 262–275.

[10] A. Sadeghirad, R. M. Brannon, J. Burghardt, A convected particle domain interpolation technique to extend applicability of the material point method for problems involving massive deformations, International Journal for numerical methods in Engineering 86 (12) (2011) 1435–1456.

[11] V. P. Nguyen, C. T. Nguyen, T. Rabczuk, S. Natarajan, On a family of convected particle domain interpolations in the material point method, Finite Elements in Analysis and Design 126 (2017) 50–64.

[12] D. Burgess, D. Sulsky, J. Brackbill, Mass matrix formulation of the FLIP particle-in-cell method, J Comp Phys 103 (1992) 1–15.

[13] J. Brackbill, On modelling angular momentum and vorticity in compressible fluid flow, Comp Phys Comm 47 (1) (1987) 1–16.

[14] E. Love, D. L. Sulsky, An energy-consistent material-point method for dynamic finite deformation plasticity, Int J Numer Meth Eng 65 (10) (2006) 1608–1638.

[15] E. Love, D. Sulsky, An unconditionally stable, energy-momentum consistent implementation of the the material point method, Comp Meth App Mech Eng 195 (2006) 3903–3925.

[16] D. Sulsky, Z. Chen, H. Schreyer, A particle method for history-dependent materials, Comp Meth in App Mech Eng 118 (1) (1994) 179–196.

[17] D. Sulsky, S. Zhou, H. Schreyer, Application of a particle-in-cell method to solid mechanics, Comp Phys Comm 87 (1) (1995) 236–252.

[18] A. Langdon, Effects of spatial grid simulation in plasmas, J Comp Phys 6 (2) (1970) 247–267.

[19] H. Okuda, Nonphysical noises and instabilities in plasma simulation due to a spatial grid, J Comp Phys 10 (3) (1972) 475–486.

[20] J. Brackbill, The ringing instability in particle-in-cell calculations of low-speed flow, J Comp Phys 75 (2) (1988) 469–492.

[21] C. Gritton, Ringing instabilities in particle methods, Ph.D. thesis, The University of Utah (2014).

[22] Y. Zhu, R. Bridson, Animating sand as a fluid, ACM Trans Graph 24 (3) (2005) 965–972.

[23] R. Bridson, Fluid simulation for computer graphics, Taylor & Francis, 2008.

[24] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, A. Selle, A material point method for snow simulation, in: ACM Transactions on Graphics (SIGGRAPH 2013), 2013, pp. 102:1–10.

[25] P. Wallstedt, J. Guilkey, Improved velocity projection for the material point method, Comp Mod in Eng and Sci 19 (3) (2007) 223.

[26] C. C. Hammerquist, J. A. Nairn, A new method for material point method particle updates that reduces noise and enhances stability, Computer Methods in Applied Mechanics and Engineering 318 (2017) 724–738.

[27] E. Edwards, R. Bridson, A high-order accurate particle-in-cell method, Int J Numer Meth Eng 90 (2012) 1073–1088.

[28] C. Fu, Q. Guo, T. Gast, C. Jiang, J. Teran, A polynomial particle-in-cell method, ACM Transactions on Graphics (TOG) 36 (6) (2017) 222.

[29] Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, C. Jiang, A moving least squares material point method with displacement discontinuity and two-way rigid body coupling, ACM Transactions on Graphics (TOG) 37 (4) (2018) 150.

[30] L. Boyd, R. Bridson, Multiflip for energetic two-phase fluid simulation, ACM Transactions on Graphics (TOG) 31 (2) (2012) 16.

[31] T. Yabe, F. Xiao, T. Utsumi, The constrained interpolation profile method for multiphase analysis, J Comp Phys 169 (2) (2001) 556–593.

[32] Y. T. Ng, C. Min, F. Gibou, An efficient fluid–solid coupling algorithm for single-phase flows, Journal of Computational Physics 228 (23) (2009) 8807–8829.

[33] A. P. Tampubolon, T. Gast, G. Klár, C. Fu, J. Teran, C. Jiang, K. Museth, Multi-species simulation of porous sand and water mixtures, ACM Transactions on Graphics (SIGGRAPH 2017) 36 (4) (2017) 105.

[34] M. Gao, A. Pradhana, X. Han, Q. Guo, G. Kot, E. Sifakis, C. Jiang, Animating fluid sediment mixture in particle-laden flows, ACM Transactions on Graphics (TOG) 37 (4) (2018) 149.