Journal of Computational Physics (2024)

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Second order accurate particle-in-cell discretization of the Navier-Stokes equations

Han Jiang[a], Craig Schroeder[b,*]

[a]The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
[b]Computer Science and Engineering, University of California, Riverside, 351 Winston Chung Hall, Riverside, CA 92521-0429

ARTICLE INFO

Article history:

ABSTRACT

We propose the use of PolyPIC transfers [10] to construct a second order accurate discretization of the Navier-Stokes equations within a particle-in-cell framework on MAC grids. We investigate the accuracy of both APIC [16, 17, 8] and quadratic PolyPIC [10] transfers and demonstrate that they are suitable for constructing schemes converging with orders of approximately 1.5 and 2.5 respectively. We combine PolyPIC transfers with BDF-2 time integration and a splitting scheme for pressure and viscosity and demonstrate that the resulting scheme is second order accurate. Prior *high order* particle-in-cell schemes interpolate accelerations (not velocities) from the grid to particles and rely on moving least squares to transfer particle velocities to the computational grid. The proposed method instead transfers velocities to particles, which avoids the accumulation of noise on particle velocities but requires the polynomial reconstruction to be performed using polynomials that are one degree higher. Since this polynomial reconstruction occurs over the regular grid (rather than irregularly distributed particles), the resulting weighted least squares problem has a fixed sparse structure, can be solved efficiently in closed form, and is independent of particle coverage.

1. Introduction

Early Material Point Method (MPM) formulations suffered from serious sources of errors and instabilities, especially the finite grid instability [18, 21, 1] and ringing [3, 12]. These issues have largely been addressed with smoother and more accurate basis functions such as splines [25, 27, 11], GIMP [1], or CPDI variants [23, 20, 24]. Weighted least squares [30] and moving least squares (MLS) [9, 28, 14] provide a similar effect by using a least-squares fit to particle data instead of weighting kernels. In MLS, a polynomial is fit to the local particle data and then evaluated as needed. By using polynomials of suitable order, a fit of any accuracy can in theory be obtained, though in practice this will be strongly limited by particle seeding density. MLS is particularly attractive due to its ability to retain transfer accuracy with irregular particle sampling [9, 28]. See [6] for a thorough overview of the development of MPM.

*Corresponding authors: email: craigs@cs.ucr.edu

Improving accuracy of MPM has generally proved to be very difficult. The first major challenge in achieving second order accuracy with MPM is the irregular distribution of the particles. Standard particle-to-grid transfers are not accurate enough when the particle distribution is irregular. The first real success at producing a higher order hybrid particle-grid scheme was [9], which succeeded at producing simulations up to fifth order accurate for a variety of PDEs in the absence of boundaries by using MLS for transfers, WENO for interpolation, and Runge-Kutta for temporal discretization. Although this scheme was not MPM (they did not consider solids), it demonstrated that MLS could be used to address the accuracy of particle-grid transfers, even with irregular particle distributions.

Achieving second order accuracy for MPM is complicated by the use of particles to compute stresses. The deformation gradient tracked on particles depends on the velocity gradients, which must be computed with sufficient accuracy. In addition, the MPM particles play a similar role to the quadrature points used in traditional finite elements [25, 26]. This leads to accurate results when particles are regularly distributed but quickly drops to first order (or even worse) when the particles move from this ideal arrangement. This problem was addressed by [28], which used MLS for both the particle-to-grid transfer and for transferring stresses from particles to the centers of grid cells. In this way, their force computation behaved as a regular grid with quadrature points in the centers of cells; this also allowed them to impose standard boundary conditions at grid boundaries. They were able to demonstrate second order accurate convergence for MPM in 1D and 2D, but not with general boundary conditions. A similar approach based on Taylor series expansions was used by [31], which demonstrated higher order on 1D problems. Second order with axis-aligned boundaries was also demonstrated by [7]. General application of Neumann or Dirichlet boundary conditions for MPM remains challenging [2], and this has never to our knowledge been achieved with higher order.

As with the finite element method, the natural boundary condition for MPM is the traction-free boundary condition. As a result, it is widespread practice for MPM methods to ignore boundary condition treatments. To apply inhomogeneous traction boundary (such as surface tension), it is necessary to integrate these forces over the boundary. The simplest way to do this is to simply spread out the interface forces over a region of particles near the boundary [15]. Another approach is to represent the boundary as part of the particle interpolation kernels as is done in CPDI/CPDI2 [23, 20, 24]. Though CPDI can be second order on perturbative problems with regularly distributed particles [29], these approaches are generally limited to first order accuracy. An alternative strategy is to construct an explicit surface representation [2], on which boundary conditions are readily applied. Another strategy is to compute boundary conditions from a level set [32, 30]. Moutsanidis et al. [19] used boundary-fitted curved grids based on NURBS, which can capture conic sections exactly, though second order convergence was only demonstrated for a vibrational test.

Many MPM formulations found in engineering (and all of the higher order schemes) use FLIP-style updates, where accelerations or forces are transfered from grid to particles rather than velocity. This formulation has attractive conservative properties, but they introduce spurious null velocity modes on particles and lead to simulation errors [4, 1, 16, 8]. These errors can be managed by including filtering into the transfer algorithm (XPIC [13]) or through the use of stabilization [9]. However, these methods do not eliminate particle noise but merely seek to maintain it at an acceptable level. Alternatively, one may directly interpolate velocities from the grid back to particles, but this results in an unacceptably high level of dissipation, since velocities are repeatedly averaged across each transfer. Schemes such as Affine Particle In Cell (APIC) [16, 17, 8] or Polynomial Particle In Cell (PolyPIC) [10] avoid this by enriching the particle representation to include not only velocities but also estimates of their derivatives as well. In this way, particles effectively store a local polynomial reconstruction of the velocity field in the vacinity of the particle. APIC has been shown to be effective alternative to FLIP or XPIC transfers for a projection-based fluid solver [8]. Since information does not accumulate on particles with APIC transfers, noise is not able to build up there. At the same time, since particles are able to represent the underlying grid velocitity much more accurately, less energy is lost during the interpolations, which in turn dramatically reduces dissipation [8]. We note that APIC, PolyPIC, and MLS are all weighted least squares polynomial reconstructions. We discuss the differences between these strategies (and compare to [9] more generally) in Section 3.2.

In this work, *we investigate APIC and quadratic PolyPIC as transfer schemes for the second order accurate solution of the Navier-Stokes equations*. This avoids the use of FLIP or XPIC for transfers and the particle noise that accompanies them. We demonstrate that the use of quadratic PolyPIC transfers combined with a standard second order backward difference formula (BDF-2) time integration scheme on the grid yields a scheme that is second order accurate in velocities in both the L_2 and L_∞ norms. We also demonstrate that using APIC instead of quadratic PolyPIC reduces the convergence order to approximately 1.5. As with [9], the focus of this work is on transfer accuracy, so we focus on periodic boundary conditions. We do not consider irregular boundary conditions, which would drastically

Name	type	location	meaning
$\triangle t$	scalar	-	time step size
Δx	scalar	-	grid resolution
μ	scalar	-	viscosity coefficient
ρ	scalar	-	fluid density
$\mathbf{e}_a$	vector	-	axis vector
m_p	scalar	particle	mass
$\mathbf{x}_p^n$	vector	particle	position
$\mathbf{v}_p^n$	vector	particle	velocity
$\mathbf{C}_p^n$	matrix	particle	velocity gradient
$\mathbf{H}_p^n$	tensor	particle	velocity Hessian
$\mathbf{v}_p^{n+\frac{1}{2}}$	vector	particle	advection velocity
$\mathbf{v}_p^{bd}$	vector	particle	intermediate velocity
$\mathbf{C}_p^{bd}$	matrix	particle	intermediate velocity gradient
$\mathbf{H}_p^{bd}$	tensor	particle	intermediate velocity Hessian
m_{ia}^n	scalar	MAC face	mass
$\mathbf{x}_{ia}^n$	vector	MAC face	MAC face location
$\hat{u}_{ia}^n$	scalar	MAC face	final grid velocity component
$u_{ia}^{\star\star}$	scalar	MAC face	intermediate velocity component
$u_{ia}^{\star}$	scalar	MAC face	intermediate velocity component
u_{ia}^{bd}	scalar	MAC face	intermediate velocity component
a_{ia}^n	scalar	MAC face	acceleration component
f_{ia}^n	scalar	MAC face	force component
p_i^n	scalar	cell center	pressure
w_{iap}^n	scalar	mixed	transfer weights
ξ	scalar	constant	second moment of transfer weights
σ_{pab}	scalar	particle	third moment of transfer weights
τ_{pab}	scalar	particle	fourth moment of transfer weights
r_{pa}	scalar	particle	zeroth moment of grid velocity
M_{pab}	scalar	particle	first moment of grid velocity
T_{pabc}	scalar	particle	second moment of grid velocity

Table 1: Summary of notation used in this paper.

complicate the underlying Eulerian scheme. We also limit our study to fluids, which avoids the additional complications involved in computing second order accurate stresses. Nevertheless, we observe that boundary conditions do not pose problems for the transfers and demonstrate second order accuracy with axis-aligned boundaries. In addition, we present efficient and explicit formulas for computing the required PolyPIC transfers, which makes PolyPIC transfers significantly less expensive than the equivalent MLS transfers. We demonstrate that quadratic splines are unsuitable for PolyPIC and that cubic splines are required instead. Finally, we compare quadratic PolyPIC with APIC and the XPIC family in terms of dissipation following the analysis of [8] in Section 4.1.

2. Numerical method

2.1. Notation

The notation that we use is summarized in Table 1. As a general rule, bold lowercase symbols ($\mathbf{x}_p^n$) are vectors, bold uppercase symbols ($\mathbf{C}_p^n$) are matrices or rank-3 tensors, and non-bold symbols ($\tilde{v}_{ia}^{n+1}$, M_{pab}) are scalars. Subscripts are used to index grid nodes (i), particles (p), and spatial dimensions (a, b, c). The superscripts $\mathbf{v}_p^{n-1}$ $\mathbf{v}_p^n$, $\mathbf{v}_p^{n+\frac{1}{2}}$, and $\mathbf{v}_p^{n+1}$ indicate the time at which quantities naturally live. Other adornments are used to distinguish quantities that would otherwise get the same name. To avoid confusion, we will never use the summation convention in this document; all summation is specified explicitly.

2.2. State variables

Hybrid particle-grid methods store their primary state on particles; this is the information that persists from time step to time step. For the proposed scheme, each particle stores mass (m_p), position ($\mathbf{x}_p^n$), velocity ($\mathbf{v}_p^n$), velocity gradient ($\mathbf{C}_p^n$), and velocity Hessian ($\mathbf{H}_p^n$). Since the discretization we propose is a multistep method, we also require velocity information for the previous time step ($\mathbf{v}_p^{n-1}$, $\mathbf{C}_p^{n-1}$, $\mathbf{H}_p^{n-1}$). Storing velocity and its first two spatial derivatives allows us to represent a quadratic polynomial velocity field on each particle, which approximates the local velocity field. Since particle mass m_p never changes, we do not assign it a superscript. We do not consider any grid information to be simulation state; all information that must be used on the grid will be reconstructed from particles during the time step.

2.3. Particle movement

We begin the time step by moving our particles. The particles effectively behave as Lagrangian particles, and moving the particles replaces the need to perform Eulerian advection. To achieve second order accuracy in time, it is not sufficient to move positions using grid velocities (or by interpolating final grid velocities from the previous time step). Instead, we must use a midpoint approximation of the velocity.

$$\mathbf{v}_p^{n+\frac{1}{2}} = \frac{3}{2}\mathbf{v}_p^n - \frac{1}{2}\mathbf{v}_p^{n-1} \tag{1}$$

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \triangle t \mathbf{v}_p^{n+\frac{1}{2}} \tag{2}$$

When particles move through a velocity gradient, the local velocity field will deform along with it. Unlike a typical hybrid scheme, we explicitly store a polynomial representation of the velocity, not merely a velocity sample. This suggests that it might be necessary to update this particle state, but we found that this is not necessary.

2.4. Particle to grid transfers

We discretize our time derivative using a standard second order backward difference ODE discretazation (which we refer to as BDF-2), which approximates the time derivative of velocity as

$$\frac{\partial \mathbf{v}_p^{n+1}}{\partial t} \approx \frac{3\mathbf{v}_p^{n+1} - 4\mathbf{v}_p^n + \mathbf{v}_p^{n-1}}{2\triangle t} = \frac{\mathbf{v}_p^{n+1} - \mathbf{v}_p^{bd}}{\alpha \triangle t}, \tag{3}$$

where we have let $\alpha = \frac{2}{3}$ and introduced the intermediates

$$\mathbf{v}_p^{bd} = \frac{4}{3}\mathbf{v}_p^n - \frac{1}{3}\mathbf{v}_p^{n-1} \qquad \mathbf{C}_p^{bd} = \frac{4}{3}\mathbf{C}_p^n - \frac{1}{3}\mathbf{C}_p^{n-1} \qquad \mathbf{H}_p^{bd} = \frac{4}{3}\mathbf{H}_p^n - \frac{1}{3}\mathbf{H}_p^{n-1}. \tag{4}$$

During the first time step, we bootstrap the multistep method with $\mathbf{v}_p^{bd} = \mathbf{v}_p^n$, $\mathbf{C}_p^{bd} = \mathbf{C}_p^n$, $\mathbf{H}_p^{bd} = \mathbf{H}_p^n$, and $\alpha = 1$.

Next, we must transfer our particle velocities to our background MAC grid. Each particle stores a quadratic velocity field, which can be evaluated at nearby MAC faces. Multiplying by particle mass yields a per-MAC face momentum. We also transfer mass the to grid.

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

$$m_{ia}^n u_{ia}^n = \sum_p w_{iap}^n m_p \mathbf{e}_a^T \left(\mathbf{v}_p^{bd} + \mathbf{C}_p^{bd}(\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) + \frac{1}{2}\mathbf{H}_p^{bd} : ((\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})(\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})^T) \right) \tag{6}$$

Then, grid velocity is $u_{ia}^n = (m_{ia}^n u_{ia}^n)/m_{ia}^n$. We use cubic splines for our transfers, though we also consider quadratic splines for comparison purposes (See Section 4.2).

2.5. Grid evolution

Once on the grid, we follow a standard second order accurate Eulerian discretization of the remaining terms of the Navier-Stokes equations. First we apply explicit forces

$$u^{\star}_{ia} = u^{n}_{ia} + \frac{\alpha \triangle t}{m^{n}_{ia}} f^{n+1}_{ia}. \tag{7}$$

In this paper, we will use the forcing term f^{n+1}_{ia} for the method of manufactured solutions [22]. Next we apply viscosity

$$u^{\star\star}_{ia} = u^{\star}_{ia} + \frac{\triangle t \alpha \mu}{\rho} \nabla^2 u^{\star\star}_{ia}. \tag{8}$$

We complete the grid evolution by projecting the final grid velocity to make it divergence free.

$$\nabla \left(\frac{1}{\rho} \nabla p^{n+1}_{i} \right) = \frac{1}{\triangle t \alpha} \nabla \cdot u^{\star\star}_{ia} \qquad\qquad \hat{u}^{n+1}_{ia} = u^{\star\star}_{ia} - \frac{\triangle t \alpha}{\rho} \nabla p^{n+1}_{i}. \tag{9}$$

This simple scheme on the background grid is sufficient to achieve second order accurate velocities. We note that the choice of grid scheme is largely independent of the transfers, which are our main focus.

We note that the scheme as presented could be improved by including a predictor for the pressure, where the previous time step's pressure would be added as an explicit force before computing viscosity. The pressure solve would then compute a pressure correction, which is added to the predicted pressure. This has the advantage of making the viscosity a bit more accurate and making the Poisson equation quicker to solve. The predicted pressure is not needed to achieve second order accuracy on the velocities. We chose to omit the predictor for the pressure, since this would have introduced a state variable on the grid. This is straightforward to handle in our case since we do not have moving boundaries, but we felt that this was out of the spirit of the scheme. Since our focus is on the transfers and not the grid portion of the scheme, we decided to omit the predictor. If performance is critical, a predictor could be included, or the pressure solver could be warm started using the pressure from the previous time step.

2.6. Grid to particle transfers

To complete the time step, we must transfer our grid velocities back to particles. We do this using quadratic PolyPIC [10] (also referred to as PolyPIC6 in that paper). For simplicity, completeness, and to develop efficient explicit formulas for computing it, we present a derivation of quadratic PolyPIC here.

Since our particles store a quadratic polynomial velocity approximation, our transfers from grid to particle amount to solving a weighted least squares problem (as one might do in a moving least squares (MLS) discretization). To represent a quadratic velocity field in 3D, one needs 3 velocity samples, 9 gradient samples, and 18 Hessian samples. Specifically, $\mathbf{v}_p, \mathbf{C}_p, \mathbf{H}_p$ should be chosen to minimize

$$\min_{\mathbf{v}_p, \mathbf{C}_p, \mathbf{H}_p} \sum_{ia} w^{n}_{iap} \left\| \mathbf{e}_a^T \mathbf{v}_p + \mathbf{e}_a^T \mathbf{C}_p (\mathbf{x}^{n}_{ia} - \mathbf{x}^{n+1}_{p}) + \frac{1}{2} \mathbf{e}_a^T \mathbf{H}_p : ((\mathbf{x}^{n}_{ia} - \mathbf{x}^{n+1}_{p})(\mathbf{x}^{n}_{ia} - \mathbf{x}^{n+1}_{p})^T) - u_{ia} \right\|^2 \tag{10}$$

Noting that these optimization problems are independent for all choices of a and p and using the notation $\mathbf{H}_{pa} = \mathbf{e}_a^T \mathbf{H}_p$,

$$\min_{\mathbf{v}_{pa}, \mathbf{C}_{pa}, \mathbf{H}_{pa}} \sum_{i} w^{n}_{iap} \left\| \mathbf{v}_{pa} + \mathbf{C}_{pa} (\mathbf{x}^{n}_{ia} - \mathbf{x}^{n+1}_{p}) + \frac{1}{2} (\mathbf{x}^{n}_{ia} - \mathbf{x}^{n+1}_{p})^T \mathbf{H}_{pa} (\mathbf{x}^{n}_{ia} - \mathbf{x}^{n+1}_{p}) - u_{ia} \right\|^2. \tag{11}$$

A straightforward solution to this weighted least squares problem would requires inverting 10×10 linear system. Three of these (one per velocity component) are required per particle. While this scales linearly with the number of particles (and generally better than the pressure projection), this can be quite expensive. Because we are doing a weighted least squares fit to grid data (not particle data as with MLS), we can take advantage of the regularity of this data to efficiently solve the weighted least squares problem. In fact, the solution can be computed directly in closed form. Doing this will require some moments.

Moments. Since we are fitting quadratic polynomials, we will need spline moments up to four:

$$1 = \sum_i w^n_{iap} \quad 0 = \sum_i w^n_{iap}(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)_b \quad \xi = \sum_i w^n_{iap}(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)^2_b \quad \sigma_{pab} = \sum_i w^n_{iap}(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)^3_b \quad \tau_{pab} = \sum_i w^n_{iap}(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)^4_b. \tag{12}$$

Here, the notation $(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)^2_b$ means evaluate the quantity in parenthesis (a vector in this case), take the b-th component of it, and square it. In the case of quadratic and cubic splines that we consider, the second moment is independent of the particle positions, so no indices are required. The third and fourth moments are simple polynomials that depend on the position of the particle within a cell. We provide simple polynomial formulas for these moments in Section 2.7 which can be used to efficiently compute them; the moments are *not* computed by summing over grid values.

We will also require moments of our final grid velocities

$$r_{pa} = \sum_i w^n_{iap}\hat{u}^{n+1}_{ia} \qquad M_{pab} = \sum_i w^n_{iap}\hat{u}^{n+1}_{ia}(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)_b \qquad T_{pabc} = \sum_i w^n_{iap}\hat{u}^{n+1}_{ia}(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)_b(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)_c. \tag{13}$$

Note that these moments are over the grid data and are computed by summing over the grid.

Weighted least squares problem. With moments available, we can write the transfer as a matrix-vector multiply. At the risk of notation reuse on i, for the rest of this section we denote the grid index as $i = (i, j, k)$. Then, we may let $\langle x_i, y_j, z_k\rangle = \mathbf{x}^n_{ia} - \mathbf{x}^{n+1}_p$, noting that a and p are fixed during this least squares solve; the a and p indices are ignored where convenient. The missing indices will be put back in the result. Noting that our weights have tensor product structure, we can write $w^n_{iap} = w_i w_j w_k$. The least squares problem can now be expressed as minimizing E where

$$E = \sum_i w^n_{iap}\|\mathbf{A}_i^T\mathbf{U} - u_{ia}\|^2. \tag{14}$$

and

$$\mathbf{A}_i = \begin{pmatrix} 1 \\ x_i \\ y_j \\ z_k \\ x_i^2 \\ y_j^2 \\ z_k^2 \\ y_j z_k \\ z_k x_i \\ x_i y_j \end{pmatrix} \qquad \mathbf{U} = \begin{pmatrix} \mathbf{v}_{pa} \\ \mathbf{C}_{pax} \\ \mathbf{C}_{pay} \\ \mathbf{C}_{paz} \\ \frac{1}{2}\mathbf{H}_{paxx} \\ \frac{1}{2}\mathbf{H}_{payy} \\ \frac{1}{2}\mathbf{H}_{pazz} \\ \mathbf{H}_{payz} \\ \mathbf{H}_{pazx} \\ \mathbf{H}_{paxy} \end{pmatrix}. \tag{15}$$

Rewriting, we have

$$E = \sum_i w^n_{iap}\|\mathbf{A}_i^T\mathbf{U} - u_{ia}\|^2 \tag{16}$$

$$= \sum_i w^n_{iap}(\mathbf{A}_i^T\mathbf{U} - u_{ia})^T(\mathbf{A}_i^T\mathbf{U} - u_{ia}) \tag{17}$$

$$= \mathbf{U}^T\left(\sum_i w^n_{iap}\mathbf{A}_i\mathbf{A}_i^T\right)\mathbf{U} - 2\left(\sum_i w^n_{iap}u_{ia}\mathbf{A}_i\right)^T\mathbf{U} + \left(\sum_i w^n_{iap}u^2_{ia}\right) \tag{18}$$

$$0 = \frac{\partial E}{\partial \mathbf{U}} = 2\left(\sum_i w^n_{iap}\mathbf{A}_i\mathbf{A}_i^T\right)\mathbf{U} - 2\left(\sum_i w^n_{iap}u_{ia}\mathbf{A}_i\right) \tag{19}$$

$$\underbrace{\left(\sum_i w^n_{iap}\mathbf{A}_i\mathbf{A}_i^T\right)}_{\mathbf{N}}\mathbf{U} = \underbrace{\sum_i w^n_{iap}u_{ia}\mathbf{A}_i}_{\mathbf{B}} \tag{20}$$

The vector **B** is just the moments over the data,

$$\mathbf{B} = \begin{pmatrix} r_{pa} \\ M_{pax} \\ M_{pay} \\ M_{paz} \\ T_{paxx} \\ T_{payy} \\ T_{pazz} \\ T_{payz} \\ T_{pazx} \\ T_{paxy} \end{pmatrix}. \tag{21}$$

In the simplified notation of this section, we can write our moments as

$$1 = \sum_i w_i \qquad 0 = \sum_i w_i x_i \qquad \xi = \sum_i w_i x_i^2 \qquad \sigma_x = \sum_i w_i x_i^3 \qquad \tau_x = \sum_i w_i x_i^4, \tag{22}$$

with similar expressions for expressions for y and z. With these,

$$\mathbf{N} = \sum_i w_{iap}^n \mathbf{A}_i \mathbf{A}_i^T \tag{23}$$

$$= \sum_{ijk} w_i w_j w_k \begin{pmatrix} 1 & x_i & y_j & z_k & x_i^2 & y_j^2 & z_k^2 & y_j z_k & z_k x_i & x_i y_j \\ x_i & x_i^2 & x_i y_j & z_k x_i & x_i^3 & x_i y_j^2 & x_i z_k^2 & x_i y_j z_k & z_k x_i^2 & x_i^2 y_j \\ y_j & x_i y_j & y_j^2 & y_j z_k & x_i^2 y_j & y_j^3 & y_j z_k^2 & y_j^2 z_k & x_i y_j z_k & x_i y_j^2 \\ z_k & z_k x_i & y_j z_k & z_k^2 & z_k x_i^2 & y_j^2 z_k & z_k^3 & y_j z_k^2 & x_i z_k^2 & x_i y_j z_k \\ x_i^2 & x_i^3 & x_i^2 y_j & z_k x_i^2 & x_i^4 & x_i^2 y_j^2 & x_i^2 z_k^2 & x_i^2 y_j z_k & x_i^3 z_k & x_i^3 y_j \\ y_j^2 & x_i y_j^2 & y_j^3 & y_j^2 z_k & x_i^2 y_j^2 & y_j^4 & y_j^2 z_k^2 & y_j^3 z_k & y_j^2 z_k x_i & y_j^3 x_i \\ z_k^2 & x_i z_k^2 & y_j z_k^2 & z_k^3 & x_i^2 z_k^2 & y_j^2 z_k^2 & z_k^4 & z_k^3 y_j & z_k^3 x_i & z_k^2 x_i y_j \\ y_j z_k & x_i y_j z_k & y_j^2 z_k & y_j z_k^2 & x_i^2 y_j z_k & y_j^3 z_k & z_k^3 y_j & y_j^2 z_k^2 & z_k^2 x_i y_j & y_j^2 z_k x_i \\ z_k x_i & z_k x_i^2 & x_i y_j z_k & x_i z_k^2 & x_i^3 z_k & y_j^2 z_k x_i & z_k^3 x_i & z_k^2 x_i y_j & x_i^2 z_k^2 & x_i^2 y_j z_k \\ x_i y_j & x_i^2 y_j & x_i y_j^2 & x_i y_j z_k & x_i^3 y_j & y_j^3 x_i & z_k^2 x_i y_j & y_j^2 z_k x_i & x_i^2 y_j z_k & x_i^2 y_j^2 \end{pmatrix} \tag{24}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & \xi & \xi & \xi & 0 & 0 & 0 \\ 0 & \xi & 0 & 0 & \sigma_x & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \xi & 0 & 0 & \sigma_y & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \xi & 0 & 0 & \sigma_z & 0 & 0 & 0 \\ \xi & \sigma_x & 0 & 0 & \tau_x & \xi^2 & \xi^2 & 0 & 0 & 0 \\ \xi & 0 & \sigma_y & 0 & \xi^2 & \tau_y & \xi^2 & 0 & 0 & 0 \\ \xi & 0 & 0 & \sigma_z & \xi^2 & \xi^2 & \tau_z & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \xi^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \xi^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \xi^2 \end{pmatrix} \tag{25}$$

Letting

$$\lambda_x = \sigma_x^2 + \xi(\xi^2 - \tau_x) \qquad \lambda_y = \sigma_y^2 + \xi(\xi^2 - \tau_y) \qquad \lambda_z = \sigma_z^2 + \xi(\xi^2 - \tau_z), \tag{26}$$

the solution can be written in closed form as

$$\mathbf{U} = \mathbf{N}^{-1}\mathbf{B} = \begin{pmatrix} 1-\frac{\xi^3}{\lambda_z}-\frac{\xi^3}{\lambda_y}-\frac{\xi^3}{\lambda_x} & -\frac{\sigma_x\xi}{\lambda_x} & -\frac{\sigma_y\xi}{\lambda_y} & -\frac{\sigma_z\xi}{\lambda_z} & \frac{\xi^2}{\lambda_x} & \frac{\xi^2}{\lambda_y} & \frac{\xi^2}{\lambda_z} & 0 & 0 & 0 \\ -\frac{\sigma_x\xi}{\lambda_x} & \frac{\xi^2-\tau_x}{\lambda_x} & 0 & 0 & \frac{\sigma_x}{\lambda_x} & 0 & 0 & 0 & 0 & 0 \\ -\frac{\sigma_y\xi}{\lambda_y} & 0 & \frac{\xi^2-\tau_y}{\lambda_y} & 0 & 0 & \frac{\sigma_y}{\lambda_y} & 0 & 0 & 0 & 0 \\ -\frac{\sigma_z\xi}{\lambda_z} & 0 & 0 & \frac{\xi^2-\tau_z}{\lambda_z} & 0 & 0 & \frac{\sigma_z}{\lambda_z} & 0 & 0 & 0 \\ \frac{\xi^2}{\lambda_x} & \frac{\sigma_x}{\lambda_x} & 0 & 0 & -\frac{\xi}{\lambda_x} & 0 & 0 & 0 & 0 & 0 \\ \frac{\xi^2}{\lambda_y} & 0 & \frac{\sigma_y}{\lambda_y} & 0 & 0 & -\frac{\xi}{\lambda_y} & 0 & 0 & 0 & 0 \\ \frac{\xi^2}{\lambda_z} & 0 & 0 & \frac{\sigma_z}{\lambda_z} & 0 & 0 & -\frac{\xi}{\lambda_z} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\xi^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\xi^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\xi^2} \end{pmatrix} \begin{pmatrix} r_{pa} \\ M_{pax} \\ M_{pay} \\ M_{paz} \\ T_{paxx} \\ T_{payy} \\ T_{pazz} \\ T_{payz} \\ T_{pazx} \\ T_{paxy} \end{pmatrix}. \tag{27}$$

From this we can read off the solution directly,

$$(\mathbf{H}_p^{n+1})_{abb} = 2\frac{\sigma_{pab}M_{pab} - \xi(T_{pabb} - \xi r_{pa})}{\sigma_{pab}^2 + \xi(\xi^2 - \tau_{pab})} \qquad (\mathbf{H}_p^{n+1})_{abc} = \frac{T_{pabc}}{\xi^2} \qquad b \neq c \tag{28}$$

$$(\mathbf{C}_p^{n+1})_{ab} = \frac{(\xi^2 - \tau_{pab})M_{pab} + \sigma_{pab}(T_{pabb} - \xi r_{pa})}{\sigma_{pab}^2 + \xi(\xi^2 - \tau_{pab})} \qquad (\mathbf{v}_p^{n+1})_a = r_{pa} - \xi\sum_b (\mathbf{H}_p^{n+1})_{abb}. \tag{29}$$

Observe that in the case of cubic splines (which are the splines we recommend using; see Section 4.2), the third moment of the transfer weights vanishes ($\sigma_{pab} = 0$), which leads to much simpler formulas for $(\mathbf{H}_p^{n+1})_{abb}$ and $(\mathbf{C}_p^{n+1})_{ab}$:

$$(\mathbf{H}_p^{n+1})_{abb} = 2\frac{\xi r_{pa} - T_{pabb}}{\xi^2 - \tau_{pab}} \qquad (\mathbf{C}_p^{n+1})_{ab} = \frac{M_{pab}}{\xi} \qquad \text{(Cubic splines only)}. \tag{30}$$

Because of the existence of these simple closed-form formulas, the grid-to-particle transfers are inexpensive.

2.7. Computing spline moments

The choice of transfer splines that is used is important in achieving good numerical properties in the overall algorithm. Unlike HOPIC [9], where the underlying spline is largely arbitrary and mostly just serves to transition out influence from particles far away, the splines for APIC and PolyPIC are very important. The mathematical properties of the splines are important, beyond merely the overall shape. Throughout this paper, we assume quadratic or cubic splines due to their smoothness (avoiding cell-crossing instabilities [1]) and favorable second moment [16].

The quadratic PolyPIC transfers rely on the spline moments up to order four, which we can summarize and name as follows:

$$1 = \sum_i w_{iap}^n \qquad 0 = \sum_i w_{iap}^n(\mathbf{x}_{ia}^n - \mathbf{x}_p^n)_b \qquad \xi = \sum_i w_{iap}^n(\mathbf{x}_{ia}^n - \mathbf{x}_p^n)_b^2 \tag{31}$$

$$\sigma_{pab} = \sum_i w_{iap}^n(\mathbf{x}_{ia}^n - \mathbf{x}_p^n)_b^3 \qquad \tau_{pab} = \sum_i w_{iap}^n(\mathbf{x}_{ia}^n - \mathbf{x}_p^n)_b^4 \tag{32}$$

The zeroth moment must be one; this is the partition of unity property. It is essential for achieving basic conservation properties (such as conservation of momentum). The first moment should be zero; this is the interpolation property. It is also essential. The second moment is special in the case of quadratic and cubic splines, since it is independent of the particle's position on the grid. Fix i as a reference grid node for the particle p and let $z = x_{ia}^n - x_{pa}^n$. The reference point is chosen such that $-\frac{\Delta x}{2} \leq z < \frac{\Delta x}{2}$ for quadratic and such that $0 \leq z < \Delta x$ for cubic. For quadratic splines,

$$\xi = \frac{\Delta x^2}{4} \qquad \sigma_{pa} = \frac{1}{4}\Delta x^2 z - z^3 \qquad \tau_{pa} = 3z^4 - \frac{3}{2}\Delta x^2 z^2 + \frac{1}{4}\Delta x^4 \tag{33}$$

For cubic splines,

$$\xi = \frac{\Delta x^2}{3} \qquad \sigma_{pa} = 0 \qquad \tau_{pa} = \frac{1}{3}\Delta x^4 - z^2(z-\Delta x)^2 \tag{34}$$

Note that for cubic splines, the vanishing cubic moment greatly simplifies the PolyPIC formulas.

The moments above are scalars, but we can also define the corresponding tensors, which will be useful later.

$$\xi \mathbf{I} = \sum_i w^n_{iap}(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)^T \tag{35}$$

$$(S_{pa})_{bcd} = \sum_i w^n_{iap}(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)_b(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)_c(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)_d \tag{36}$$

$$(R_{pa})_{bcde} = \sum_i w^n_{iap}(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)_b(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)_c(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)_d(\mathbf{x}^n_{ia} - \mathbf{x}^n_p)_e \tag{37}$$

Noting that the zero moment is one and the first moment is zero, most entries of the third and fourth order tensors vanish, with the only nonzero entries being

$$(S_{pa})_{bbb} = \sigma_{pab} \qquad (R_{pa})_{bbbb} = \tau_{pab} \qquad (R_{pa})_{bbcc} = (R_{pa})_{bcbc} = (R_{pa})_{bccb} = \xi^2 \qquad b \neq c. \tag{38}$$

In the case of cubic splines, the third order tensor vanishes entirely.

2.8. Summary

For clarity, we summarize the full numerical method here.

1. Move particles using a midpoint velocity approximation

$$\mathbf{v}_p^{n+\frac{1}{2}} = \frac{3}{2}\mathbf{v}_p^n - \frac{1}{2}\mathbf{v}_p^{n-1} \qquad \mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \triangle t \mathbf{v}_p^{n+\frac{1}{2}} \tag{39}$$

2. During the first time step, let $\alpha = 1$; thereafter, $\alpha = \frac{2}{3}$.
3. Compute the temporal intermediates on particles using the BDF-2 temporal derivative approximation.

$$\mathbf{v}_p^{bd} = (2-\alpha)\mathbf{v}_p^n - (\alpha-1)\mathbf{v}_p^{n-1} \qquad \mathbf{C}_p^{bd} = (2-\alpha)\mathbf{C}_p^n - (\alpha-1)\mathbf{C}_p^{n-1} \qquad \mathbf{H}_p^{bd} = (2-\alpha)\mathbf{H}_p^n - (\alpha-1)\mathbf{H}_p^{n-1}. \tag{40}$$

4. Transfer mass and momentum from particles to grid, after which velocity is obtained by division.

$$m^n_{ia}u^n_{ia} = \sum_p w^n_{iap} m_p \mathbf{e}_a^T \left(\mathbf{v}_p^{bd} + \mathbf{C}_p^{bd}(\mathbf{x}^n_{ia} - \mathbf{x}_p^{n+1}) + \frac{1}{2}\mathbf{H}_p^{bd} : ((\mathbf{x}^n_{ia} - \mathbf{x}_p^{n+1})(\mathbf{x}^n_{ia} - \mathbf{x}_p^{n+1})^T) \right) \tag{41}$$

$$m^n_{ia} = \sum_p w^n_{iap} m_p \qquad u^n_{ia} = \frac{m^n_{ia}u^n_{ia}}{m^n_{ia}} \tag{42}$$

5. Apply explicit forces

$$u^\star_{ia} = u^n_{ia} + \frac{\alpha \triangle t}{m^n_{ia}} f^{n+1}_{ia}. \tag{43}$$

6. Add viscosity by solving a Helmholtz equation

$$u^{\star\star}_{ia} = u^\star_{ia} + \frac{\triangle t \alpha \mu}{\rho}\nabla^2 u^{\star\star}_{ia}. \tag{44}$$

7. Compute pressures and project the velocities to be divergence free by solving a Poisson equation.

$$\nabla\left(\frac{1}{\rho}\nabla p_i^{n+1}\right) = \frac{1}{\triangle t \alpha}\nabla \cdot u^{\star\star}_{ia} \qquad \hat{u}^{n+1}_{ia} = u^{\star\star}_{ia} - \frac{\triangle t \alpha}{\rho}\nabla p_i^{n+1}. \tag{45}$$

8. Compute the grid velocity moments

$$r_{pa} = \sum_i w_{iap}^n \hat{u}_{ia}^{n+1} \qquad M_{pab} = \sum_i w_{iap}^n \hat{u}_{ia}^{n+1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^n)_b \qquad T_{pabc} = \sum_i w_{iap}^n \hat{u}_{ia}^{n+1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^n)_b (\mathbf{x}_{ia}^n - \mathbf{x}_p^n)_c. \tag{46}$$

9. Compute the spline moments ξ, σ_{pab}, and τ_{pab} using the explicit polynomial formulas in Section 2.7.
10. Complete the PolyPIC grid-to-particle transfers using the closed form solutions to compute the new particle velocities $\mathbf{v}_p^{n+1}$, velocity gradients $\mathbf{C}_p^{n+1}$, and velocity Hessians $\mathbf{H}_p^{n+1}$.

$$(\mathbf{H}_p^{n+1})_{abb} = 2\frac{\sigma_{pab} M_{pab} - \xi(T_{pabb} - \xi r_{pa})}{\sigma_{pab}^2 + \xi(\xi^2 - \tau_{pab})} \qquad (\mathbf{H}_p^{n+1})_{abc} = \frac{T_{pabc}}{\xi^2} \qquad b \neq c \tag{47}$$

$$(\mathbf{C}_p^{n+1})_{ab} = \frac{(\xi^2 - \tau_{pab}) M_{pab} + \sigma_{pab}(T_{pabb} - \xi r_{pa})}{\sigma_{pab}^2 + \xi(\xi^2 - \tau_{pab})} \qquad (\mathbf{v}_p^{n+1})_a = r_{pa} - \xi \sum_b (\mathbf{H}_p^{n+1})_{abb}. \tag{48}$$

3. Notes and analysis

3.1. Stability of quadratic and cubic splines

The stability of the transfers can be predicted by examining the denominators in the PolyPIC formulas: $\sigma_{pa}^2 + \xi(\xi^2 - \tau_{pa})$. In the case of cubic,

$$\text{denom} = \sigma_{pa}^2 + \xi(\xi^2 - \tau_{pa}) = \frac{\Delta x^2}{27}\left(-2\Delta x^4 + 9z^2(z - \Delta x)^2\right) \tag{49}$$

$$-\frac{2}{27}\Delta x^6 \leq \text{denom} \leq -\frac{23}{432}\Delta x^6 \qquad 0 \leq z < \Delta x \tag{50}$$

This is safely bounded away from zero, so there are no numerical difficulties with PolyPIC transfers for cubic splines. For quadratic splines, however,

$$\text{denom} = \sigma_{pa}^2 + \xi(\xi^2 - \tau_{pa}) = -\frac{1}{64}(3\Delta x^2 - 4z^2)(\Delta x - 2z)^2(\Delta x + 2z)^2 \qquad -\frac{\Delta x}{2} \leq z < \frac{\Delta x}{2} \tag{51}$$

In this case, we see that the denominator is zero at $z = \pm\frac{\Delta x}{2}$, which occurs when particles reach the edge of the support of the weighting functions. As a result, quadratic splines should not be used with PolyPIC. Our numerical studies bear this out; see Sections 4.2, 4.3 and 4.4.

The reason for the numerical difficulties can be understood by counting degrees of freedom. In 1D, there are three coefficients in the quadratic polynomial being reconstructed for PolyPIC, and the quadratic basis splines will involve three grid nodes in their support. At the edge of this support, one of those nodes transitions to zero weight. In that limit, only two nodes have nonzero weight, and there are not enough points to fit the quadratic polynomial for PolyPIC. The weighted least squares problem becomes underdetermined.

3.2. Comparison to MLS

Both PolyPIC and MLS are used to perform grid-to-particle and particle-to-grid transfers, and both are based on weighted least squares. However, the resulting schemes are very different and have very different properties, which we discuss here. For purposes of comparison, we focus on [9] here, noting that details may differ somewhat between implementations.

Noise vs dissipation. Edwards and Bridson [9] transfer accelerations from the grid back to particles, while our PolyPIC scheme transfers velocities back to particles instead. Transferring accelerations means information must be accumulated on particles, which allows noise to build up there. Edwards and Bridson [9] use a filter to encourage these noisy modes to decay away while keeping the errors below the truncation error. The velocity transfer strategy does not accumulate information on particles, so no noise develops there. However, the price of this is repeated averaging. Unlike the acceleration transfer strategy, velocities are effectively averaged twice per time step, which can cause significant dissipation.

Temporal errors. The difference between the two strategies has dramatic effects on the truncation errors of the overall scheme. A quadratic polynomial weighted least squares transfer results in $O(\Delta x^3)$ truncation error (see Section 4.2). Consider that both the MLS and PolyPIC schemes are run for one time step of size $\triangle t$. In the PolyPIC scheme, this causes $O(\Delta x^3)$ truncation error. However, in the MLS case, only the velocity changes are being transferred, and those changes will be proportional to $\triangle t$. In particular, if a time step were taken with $\triangle t = 0$, no accelerations would be applied on the grid, so no changes to particle state would occur. Because of this, the resulting truncation error is only $O(\triangle t \Delta x^3)$. Assuming refinement is performed using $\triangle t \sim \Delta x$, MLS is able to achieve the same level of accuracy using a polynomial reconstruction that is one degree less. This is consistent with MLS using cubic polynomials for fourth order accuracy and our scheme using second order polynomials for second order accuracy. In practice, the accuracy loss is not a full order as this simple argument suggests; the actual loss is observed to be closer to half of an order (See Sections 4.3 and 4.4). In particular, APIC is observed to produce a full scheme whose convergence order is approximately 1.5, less than the second order one might expect from the spatial accuracy alone (APIC transfers introduce $O(\Delta x^2)$ truncation error), but better than the first order that the argument above suggests. Based on Section 4.3, we would expect quadratic PolyPIC to suffice up to a refinement order of about 2.5, but since the remainder of the scheme is only second order, that is what we observe.

Properties of the weighted least squares problem. Edwards and Bridson [9] solves a weighted least squares problem during the transfer from particles to grid, whereas PolyPIC solves this problem during the transfer from grid to particles. There are several noteworthy consequences of this difference. (1) The least squares problem for MLS depends on all of the positions of the particles within the support radius, and the properties of the matrix that must be inverted depend on all of those positions. For PolyPIC, the data for the transfers is stored on a regular grid, only the position of the output particle affects the numerical properties of the matrix. (2) The PolyPIC matrix is sparse with a relatively simple fixed structure and many symmetries. This matrix can be inverted symbolically, and the result is also sparse with relatively simple structure and entries. As a result, the least squares problem can be solved in closed form without the need to assemble and solve a linear system at runtime. The explicit formulas for this solution are provided in Section 2.6. For MLS case, the matrix that must be inverted will be dense and must be inverted numerically. (3) PolyPIC grid-to-particle transfers must store a polynomial least squares solution in each particle so that they can be used during the particle-to-grid transfer. This increases the memory requirements by 27 floats per particle in 3D.

Particle coverage. PolyPIC and MLS transfers differ significantly in terms of their sensitivity to particle coverage. PolyPIC transfers velocity information to the grid using the same kernels that it uses to transfer this information back to particles. As such, grid velocity information is always available at precisely the same grid locations where it is required during the least squares transfers. Provided cubic splines are used for interpolation, the least squares problem will always be well-conditioned. Since each particle carries a full polynomial approximation of the velocity field in its local neighborhood, accurate grid velocities can be obtained from even a single particle. Deficient particle coverage at the boundaries of domains does not cause problems (See Section 4.7). For MLS, adequate particle coverage is very important. If a grid location lacks sufficient neighboring particles within the kernel support radius (or those particles are arranged unfavorably), the least squares problem will be underdetermined, which can lead to errors and impede convergence. Edwards and Bridson [9] only considered periodic problems partly to avoid these complications.

Importance of transfer kernel choice. Edwards and Bridson [9] chose an arbitrary kernel for their transfers, since they found the choice of transfer kernel to be relatively insignificant. PolyPIC, by contrast, depends significantly on the properties of the kernels. In particular, the least squares matrix is sparse because the first moment of the quadratic and cubic B-splines vanish (See Section 2.6). In addition, the transfer formulas for cubic B-splines are dramatically simplified because the third moment also vanishes (See Section 2.7).

3.3. Momentum

Under normal circumstances a particle with mass m_p moving with velocity $\mathbf{v}_p^n$ would be expected to contribute $m_p\mathbf{v}_p^n$ momentum and $\mathbf{x}_p^n \times m_p\mathbf{v}_p^n$ angular momentum. Similarly, one might expect the particle to possess $\frac{1}{2}m_p\|\mathbf{v}_p^n\|^2$ kinetic energy. With APIC, particles represent the usual amount of momentum but slightly more angular momentum ($\mathbf{x}_p^n \times m_p\mathbf{v}_p^n + m_p\xi(\mathbf{C}_p^n)^T : \boldsymbol{\epsilon}$) and kinetic energy ($\frac{1}{2}m_p\|\mathbf{v}_p^n\|^2 + \frac{1}{2}m_p\xi\|\mathbf{C}_p^n\|_F^2$), where $\boldsymbol{\epsilon}$ is the rank-3 permutation tensor. In that case, the quantities on the particle were defined based on what would exist on the grid if the particle were transferred to the grid in isolation. We use the same strategy to define these quantities on particles for PolyPIC.

The mass and momentum that would be on the grid after transfers if there were only one particle p would be

$$m_{ia}^n = w_{iap}^n m_p \tag{52}$$

$$m_{ia}^n u_{ia}^n = w_{iap}^n m_p \mathbf{e}_a^T \left(\mathbf{v}_p^{bd} + \mathbf{C}_p^{bd}(\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) + \frac{1}{2}\mathbf{H}_p^{bd} : ((\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})(\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})^T) \right) \tag{53}$$

$$u_{ia}^n = \mathbf{e}_a^T \left(\mathbf{v}_p^{bd} + \mathbf{C}_p^{bd}(\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) + \frac{1}{2}\mathbf{H}_p^{bd} : ((\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})(\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})^T) \right) \tag{54}$$

The particle's momentum $\mathbf{p}_p^{bd}$ is defined as the total grid momentum on the grid after this transfer, or

$$\mathbf{p}_p^{bd} = \sum_{ia} m_{ia}^n u_{ia}^n \mathbf{e}_a \tag{55}$$

$$= \sum_{ia} w_{iap}^n m_p \mathbf{e}_a \mathbf{e}_a^T \left(\mathbf{v}_p^{bd} + \mathbf{C}_p^{bd}(\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) + \frac{1}{2}\mathbf{H}_p^{bd} : ((\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})(\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})^T) \right) \tag{56}$$

$$= \left(\sum_a \left(\sum_i w_{iap}^n \right) \mathbf{e}_a \mathbf{e}_a^T \right) m_p \mathbf{v}_p^{bd} + \sum_a m_p \mathbf{e}_a \mathbf{e}_a^T \mathbf{C}_p^{bd} \sum_i w_{iap}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) \tag{57}$$

$$+ \frac{1}{2} m_p \left(\sum_a \mathbf{e}_a \mathbf{e}_a^T \right) \left(\mathbf{H}_p^{bd} : \sum_i w_{iap}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})(\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})^T \right) \tag{58}$$

$$= m_p \mathbf{v}_p^{bd} + \frac{1}{2} m_p \xi (\mathbf{H}_p^{bd} : \mathbf{I}) \tag{59}$$

From this we deduce that PolyPIC particles effectively carry an amount of momentum equal to

$$\mathbf{p}_p^n = m_p \mathbf{v}_p^n + \frac{1}{2} m_p \xi (\mathbf{H}_p^n : \mathbf{I}). \tag{60}$$

The grid will then receive momentum equal to $\mathbf{p}_p^{bd} = \frac{4}{3}\mathbf{p}_p^n - \frac{1}{3}\mathbf{p}_p^{n-1}$. Since total momentum is conserved, the grid will have the same total momentum as the particles. This definition of momentum on particles allows us to track momentum conservation across transfers and also to analyze the momentum conservation properties of the transfers.

3.4. Angular momentum

Similarly for momentum, a particle's angular velocity will be

$$\mathbf{l}_p^{bd} = \sum_{ia} \mathbf{x}_{ia}^n \times m_{ia}^n u_{ia}^n \mathbf{e}_a \tag{61}$$

$$= \sum_{ia} w_{iap}^n m_p (\mathbf{x}_{ia}^n \times \mathbf{e}_a) \mathbf{e}_a^T \left(\mathbf{v}_p^{bd} + \mathbf{C}_p^{bd}(\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1}) + \frac{1}{2}\mathbf{H}_p^{bd} : ((\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})(\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})^T) \right) \tag{62}$$

Expanding the big parenthesis leads to three terms, which we simplify separately. The constant piece is

$$\sum_{ia} w_{iap}^n m_p (\mathbf{x}_{ia}^n \times \mathbf{e}_a) \mathbf{e}_a^T \mathbf{v}_p^{bd} = \sum_a \left(\left(\sum_i w_{iap}^n \mathbf{x}_{ia}^n \right) \times \mathbf{e}_a \right) \mathbf{e}_a^T m_p \mathbf{v}_p^{bd} = \mathbf{x}_p^n \times \left(\left(\sum_a \mathbf{e}_a \mathbf{e}_a^T \right) m_p \mathbf{v}_p^{bd} \right) = \mathbf{x}_p^n \times m_p \mathbf{v}_p^{bd}. \tag{63}$$

The linear term is

$$\sum_{ia} w_{iap}^n m_p (\mathbf{x}_{ia}^n \times \mathbf{e}_a)(\mathbf{e}_a^T \mathbf{C}_p^{bd}(\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})) \tag{64}$$

$$= \sum_a m_p \mathbf{e}_a^{*T} \left(\sum_i w_{iap}^n \mathbf{x}_{ia}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})^T \right) (\mathbf{C}_p^{bd})^T \mathbf{e}_a \tag{65}$$

$$= \sum_a m_p \mathbf{e}_a^{*T} \left(\sum_i w_{iap}^n (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})(\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})^T \right) (\mathbf{C}_p^{bd})^T \mathbf{e}_a + \sum_a m_p \mathbf{e}_a^{*T} \left(\sum_i w_{iap}^n \mathbf{x}_p^{n+1} (\mathbf{x}_{ia}^n - \mathbf{x}_p^{n+1})^T \right) (\mathbf{C}_p^{bd})^T \mathbf{e}_a \tag{66}$$

$$= \xi m_p \sum_a \mathbf{e}_a^{*T} (\mathbf{C}_p^{bd})^T \mathbf{e}_a \tag{67}$$

$$= \xi m_p (\mathbf{C}_p^{bd})^T : \boldsymbol{\epsilon} \tag{68}$$

The third piece is

$$\sum_{ia} w^n_{iap} m_p (\mathbf{x}^n_{ia} \times \mathbf{e}_a) \mathbf{e}_a^T \left(\frac{1}{2} \mathbf{H}^{bd}_p : ((\mathbf{x}^n_{ia} - \mathbf{x}^{n+1}_p)(\mathbf{x}^n_{ia} - \mathbf{x}^{n+1}_p)^T) \right). \tag{69}$$

To make the notation and manipulations more manageable, we drop the superscripts and p subscript (which are unambiguous).

$$\mathbf{b} = \sum_{ia} w_{ia} m (\mathbf{x}_{ia} \times \mathbf{e}_a) \mathbf{e}_a^T \left(\frac{1}{2} \mathbf{H} : ((\mathbf{x}_{ia} - \mathbf{x})(\mathbf{x}_{ia} - \mathbf{x})^T) \right). \tag{70}$$

Next, we switch to index notation, applying the usual summation convention only to indices that are Greek letters.

$$\begin{aligned}
b_r &= \frac{1}{2} m \sum_{ia} w_{ia} e_{r\kappa a} x_{ia\kappa} H_{a\beta\gamma} (x_{ia\beta} - x_\beta)(x_{ia\gamma} - x_\gamma) && (71)\\
&= \frac{1}{2} m e_{r\kappa a} H_{a\beta\gamma} \sum_{ia} w_{ia} x_{ia\kappa} (x_{ia\beta} - x_\beta)(x_{ia\gamma} - x_\gamma) && (72)\\
&= \frac{1}{2} m e_{r\kappa a} H_{a\beta\gamma} \sum_{ia} w_{ia} (x_{ia\kappa} - x_\kappa)(x_{ia\beta} - x_\beta)(x_{ia\gamma} - x_\gamma) + \frac{1}{2} m e_{r\kappa a} H_{a\beta\gamma} \sum_{ia} w_{ia} x_\kappa (x_{ia\beta} - x_\beta)(x_{ia\gamma} - x_\gamma) && (73)\\
&= \frac{1}{2} m \sum_{a} e_{r\kappa a} H_{a\beta\gamma} S_{a\kappa\beta\gamma} + \frac{1}{2} m \sum_{a} e_{r\kappa a} H_{a\beta\gamma} x_\kappa \xi \delta_{\beta\gamma} && (74)\\
&= \sum_{ab} \frac{1}{2} m e_{rba} H_{abb} \sigma_{ab} + \frac{1}{2} m \sum_{a} e_{r\kappa a} x_\kappa \xi H_{a\beta\gamma} \delta_{\beta\gamma} && (75)\\
\mathbf{b} &= \frac{1}{2} m \boldsymbol{\epsilon} : \mathbf{K} + \frac{1}{2} m \xi \mathbf{x} \times (\mathbf{H} : \mathbf{I}) && (76)\\
\mathbf{b}_p &= \frac{1}{2} m_p \boldsymbol{\epsilon} : \mathbf{K}_p + \frac{1}{2} m_p \xi \mathbf{x}^{n+1}_p \times (\mathbf{H}^{bd}_p : \mathbf{I}) && (77)
\end{aligned}$$

where we have defined $(K_p)_{ba} = H_{pabb}\sigma_{pab}$ (with no summation implied). The angular momentum on a particle is then

$$\mathbf{l}^n_p = \mathbf{x}^n_p \times m_p \mathbf{v}^n_p + \xi m_p (\mathbf{C}^n_p)^T : \boldsymbol{\epsilon} + \frac{1}{2} m_p \boldsymbol{\epsilon} : \mathbf{K}_p + \frac{1}{2} m_p \xi \mathbf{x}^{n+1}_p \times (\mathbf{H}^n_p : \mathbf{I}). \tag{78}$$

As with linear momentum, this particle-based definition of angular momentum allows us to track angular momentum conservation across the transfers.

3.5. Kinetic energy

As with momentum and angular momentum, kinetic energy for a particle is defined as the kinetic energy that would be deposited onto the grid if the particle were transferred in isolation.

$$\begin{aligned}
E_p &= \frac{1}{2} \sum_{ia} m^n_{ia} (u^n_{ia})^2 && (79)\\
&= \frac{1}{2} \sum_{ia} w^n_{iap} m_p \mathbf{e}_a^T \left(\mathbf{v}^{bd}_p + \mathbf{C}^{bd}_p (\mathbf{x}^n_{ia} - \mathbf{x}^{n+1}_p) + \frac{1}{2} \mathbf{H}^{bd}_p : ((\mathbf{x}^n_{ia} - \mathbf{x}^{n+1}_p)(\mathbf{x}^n_{ia} - \mathbf{x}^{n+1}_p)^T) \right) && (80)\\
&\quad \mathbf{e}_a^T \left(\mathbf{v}^{bd}_p + \mathbf{C}^{bd}_p (\mathbf{x}^n_{ia} - \mathbf{x}^{n+1}_p) + \frac{1}{2} \mathbf{H}^{bd}_p : ((\mathbf{x}^n_{ia} - \mathbf{x}^{n+1}_p)(\mathbf{x}^n_{ia} - \mathbf{x}^{n+1}_p)^T) \right) && (81)
\end{aligned}$$

After expanding and substituting in nonvanishing components of the moments,

$$\begin{aligned}
E_p &= \frac{1}{2} m_p \|\mathbf{v}^{bd}_p\|^2 + \frac{1}{2} m_p \xi \|\mathbf{C}^{bd}_p\|^2 + \frac{1}{2} m_p \xi \mathbf{v}^{bd}_p \cdot \mathbf{H}^{bd}_p : \mathbf{I} + \frac{1}{2} m_p \sum_{ab} \sigma_{pab} (\mathbf{C}^{bd}_p)_{ab} (\mathbf{H}^{bd}_p)_{abb} && (82)\\
&\quad + \frac{1}{8} m_p \sum_{ab} (\tau_{pab} - 3\xi^2) (\mathbf{H}^{bd}_p)^2_{abb} + \frac{1}{4} m_p \xi^2 \sum_{abc} (\mathbf{H}^{bd}_p)^2_{abc} + \frac{1}{8} m_p \xi^2 \|\mathbf{H}^{bd}_p : \mathbf{I}\|^2. && (83)
\end{aligned}$$

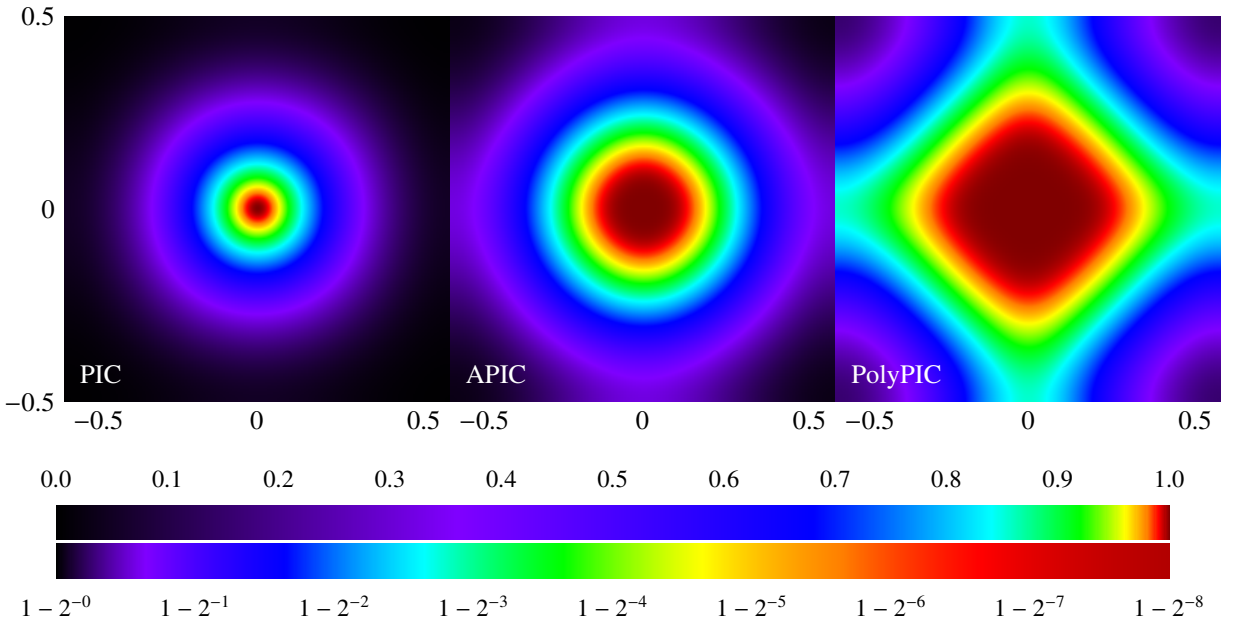

Fig. 1: This figure shows the rate at which PIC, APIC, and PolyPIC transfers dissipate different Fourier modes. The middle of the image corresponds to low-frequency modes and the outside of the image corresponds to high frequency modes as in [8]. The dark red modes are not appreciably dissipated; black modes are almost entirely filtered by transfers. As noted in [8], APIC is significantly less dissipative than PIC. Here we see that PolyPIC is quite significantly less dissipative than APIC. Indeed, the range of frequencies that are not significantly dissipated is significantly wider.

Replacing *bd* quantities with time *n* quantities gives us an instantaneous measure of kinetic energy on particles. We use this measure of kinetic energy in Section 4.5, where we separately evaluate the kinetic energy on particles and the grid.

4. Numerical examples

In this section we demonstrate the numerical behavior of the proposed scheme and explore the behavior compared to alternatives. Throughout this section, we estimate errors for grid velocities u_{ia} and particle velocities $\mathbf{v}_p$ under the L_∞ and L_2 error norms.

In all examples, particles are seeded in the relevant portion of the domain using Poisson disk sampling [5] with $n = 2^d$ particles per cell (where $d = 2, 3$ is the dimension). All particles have the same mass $m_p = \frac{\Delta x^d}{2^d}\rho$. For analytic tests with nonzero initial velocity fields $\mathbf{u}(\mathbf{x})$, particle velocity information is initialized with $\mathbf{v}_p = \mathbf{u}(\mathbf{x}_p)$. For APIC or PolyPIC, $\mathbf{C}_p$ is sampled from the analytic velocity gradient, $\mathbf{C}_p = \nabla\mathbf{u}(\mathbf{x}_p)$. For PolyPIC, $\mathbf{H}_p$ is sampled from the analytic velocity Hessian, $\mathbf{H}_p = \nabla\nabla\mathbf{u}(\mathbf{x}_p)$. All physical quantities are assumed to be in SI units. Unless otherwise stated, all simulations use a domain of $[-\pi, \pi]^d$ with periodic boundary conditions. The initial grid resolution is 32×32 in 2D and $16 \times 16 \times 16$ in 3D. The initial time step size is $\triangle t = \frac{1}{24}$ in 2D and $\triangle t = \frac{1}{12}$ in 3D. Refinement is performed by doubling the resolution and taking twice as many time steps.

4.1. PolyPIC dissipation

One of the original motivations for PolyPIC was as a means for producing MPM simulations with reduced dissipation as a result of transfers between the particles and grid [10]. While our motivation for using PolyPIC is its potential for more accurate transfers and not dissipation per se., it is interesting to consider how the improved accuracy the transfers affects the amount of dissipation that it produces. To do this, we follow the dissipation analysis of [8].

Begin with an $N \times N$ grid with four particles regularly seeded in each cell and periodic boundaries. Next, consider the transfer from grid velocities to particles and back to the grid, without moving the particles. This mimics the situation that occurs when very small time steps are taken, since velocities do not change much on the grid due to

forces and particles do not move much. Under these conditions, the grid-particle-grid round-trip transfer matrix M_{ij} is a linear mapping from grid velocities u_j to grid velocities u_i. Due to the periodic boundary conditions and identical particle placement in each cell, all cells are identical, so that M_{ij} is circulant in the multidimensional sense. Let $i = (r, s)$ and $j = (u, v)$ be grid indices in 2D, so that $M_{ij} = M_{(r,s),(u,v)} = M_{(r+k,s+m),(u+k,v+m)}$ for any k and m (treating indices as periodic). Thus, we can describe the transfer matrix by its first column c_{ij}, noting that $M_{(r,s),(u,v)} = c_{r-u,s-v}$. The eigenvalues $\lambda_i = \lambda_{rs}$ of M_{ij} are given by the Fourier transform

$$\lambda_{rs} = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} c_{uv} e^{\frac{2\pi i r u}{N}} e^{\frac{2\pi i s v}{N}} = \sum_{u=-w}^{N-1-w}\sum_{v=-w}^{N-1-w} c_{uv} e^{\frac{2\pi i r u}{N}} e^{\frac{2\pi i s v}{N}} \qquad w = \left\lfloor \frac{N}{2} \right\rfloor \tag{84}$$

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

$$\lambda(x, y) = \sum_{u=-w}^{w}\sum_{v=-w}^{w} c_{uv} e^{2\pi i x u} e^{2\pi i y v}. \tag{85}$$

In the limit of increasing resolution, $N \to \infty$, the function $\lambda(x, y)$ converges to a continuous real-valued function. The transfers are local, so c_{uv} contains only a fixed finite number of entries, so $\lambda(x, y)$ is given by a finite number of sines and cosines. Due to the symmetrical particle layout, $\lambda(x, y)$ is real-valued, and $\lambda(x, y) = \lambda(-x, y) = \lambda(x, -y) = \lambda(-x, -y)$. A plot of $\lambda(x, y)$ is shown in Figure 1, with the constant Fourier mode $\lambda(0, 0)$ in the middle of the image. If the round trip transfers were perfect, the matrix describing the grid to particle to grid transfer would be the identity, and its eigenvalues would be identically 1 (dark red in the figure). In practice, the transfers result in some dissipation, which appears as eigenvalues between 0 and 1. Eigenmodes near zero are effectively filtered out entirely. Figure 1 shows the eigenvalue images for PolyPIC alongside PIC and APIC transfers for comparison. The center is the constant mode, which is untouched by all three transfer algorithms. The dark red areas have the least dissipation, with the center of the image corresponding to low-frequency modes. The larger the dark red and red areas are, the less dissipative the transfers are. Note that the colors are on a logarithmic scale (centered around 1), since the modes very near 1 are the most important. We can see that just as APIC is far less dissipative than PIC, so too is PolyPIC far less dissipative than APIC. Compared to the XPIC results (See Figure 8 in [8]), APIC was around XPIC 2.3 (partway between 2 and 3, but closer to 2), while PolyPIC is similar to XPIC 5.

4.2. PolyPIC vs APIC single-transfer accuracy

In this test, we investigate the spatial errors for APIC and PolyPIC transfers. The velocities are initialized on the grid as $\mathbf{v}(\mathbf{x}) = \langle \cos(y)\sin(x), -\sin(y)\cos(x) \rangle$ on a $[-\pi, \pi]^2$ domain with periodic boundaries. All particles have the same mass. We disable all parts of the time integration other than particle to grid and grid to particle transfers. We do not move particles or apply viscosity or pressure. Thus, we are looking purely at the effects of the transfers. The initial grid resolution is 32×32 and we double the resolution for each refinement level. For each level of spatial refinement, we perform one round of grid-to-particle-to-grid transfers and report the velocity errors on particles and the grid. Since only one set of transfers is performed, there is no refinement in time. In particular, this test is measuring only spatial discretization errors. The same test was run with APIC and PolyPIC transfers, as well as quadratic and cubic splines. Table 2 shows the convergence results. APIC transfers with both splines converges cleanly at second order on particle velocities and approximately second order on grid velocities. PolyPIC with cubic splines shows clean third-order convergence on particle velocities and approximate third-order convergence on grid velocities. These results show that an APIC transfer is $O(\Delta x^2)$ and a PolyPIC transfer is $O(\Delta x^3)$.

For quadratic splines, however, particle velocity errors diverge as grid resolution increases. This is due to insufficient velocity information on the grid when particles approach the edge of the support of the transfer weights, effectively reducing the number of particles involved in the transfer. In 1D, PolyPIC would store three dofs on particles (one for each of $\mathbf{v}, \mathbf{C}, \mathbf{H}$, corresponding to the three coefficients of the cubic polynomial stored on the particle). This corresponds nicely with the three grid dofs that a quadratic spline would access from the grid. However, when the particle reaches the edge of the support of the weights, one of the weights vanishes, and effectively only two grid degrees of freedom are available for the transfer. The resulting problem is underdetermined, thus leading to the nullspace and the corresponding numerical accuracy problems. As refinement is increased, there are more opportunities to encounter particles interacting with faces near the edge of their support. For this reason, PolyPIC should not

			u_{ia}-L_2		u_{ia}-L_∞		$\mathbf{v}_p$-L_2		$\mathbf{v}_p$-L_∞	
		res	error	order	error	order	error	order	error	order
PolyPIC	cubic	32	2.87×10^{-5}		1.16×10^{-4}		1.14×10^{-4}		2.56×10^{-4}	
		64	3.69×10^{-6}	2.96	1.49×10^{-5}	2.96	1.40×10^{-5}	3.03	2.98×10^{-5}	3.10
		128	4.61×10^{-7}	3.00	2.38×10^{-6}	2.65	1.73×10^{-6}	3.02	3.59×10^{-6}	3.06
		256	5.75×10^{-8}	3.00	3.01×10^{-7}	2.99	2.16×10^{-7}	3.00	4.39×10^{-7}	3.03
		512	7.19×10^{-9}	3.00	4.07×10^{-8}	2.88	2.69×10^{-8}	3.00	5.44×10^{-8}	3.01
	quadratic	32	1.71×10^{-5}		6.39×10^{-5}		2.18×10^{-4}		5.11×10^{-4}	
		64	2.16×10^{-6}	2.99	9.57×10^{-6}	2.74	2.75×10^{-5}	2.99	6.10×10^{-5}	3.07
		128	2.69×10^{-7}	3.00	1.43×10^{-6}	2.74	3.43×10^{-6}	3.00	1.31×10^{-5}	2.22
		256	3.39×10^{-8}	2.99	1.90×10^{-7}	2.91	5.42×10^{-6}	-0.66	3.43×10^{-3}	-8.03
		512	4.24×10^{-9}	3.00	2.46×10^{-8}	2.95	3.27×10^{-6}	0.73	4.30×10^{-3}	-0.33
APIC	cubic	32	4.26×10^{-4}		1.88×10^{-3}		6.38×10^{-3}		1.28×10^{-2}	
		64	9.75×10^{-5}	2.13	4.62×10^{-4}	2.03	1.60×10^{-3}	1.99	3.21×10^{-3}	1.99
		128	2.48×10^{-5}	1.98	1.40×10^{-4}	1.72	4.01×10^{-4}	2.00	8.03×10^{-4}	2.00
		256	6.14×10^{-6}	2.01	3.99×10^{-5}	1.81	1.00×10^{-4}	2.00	2.01×10^{-4}	2.00
		512	1.54×10^{-6}	2.00	9.37×10^{-6}	2.09	2.51×10^{-5}	2.00	5.02×10^{-5}	2.00
	quadratic	32	3.43×10^{-4}		1.26×10^{-3}		4.79×10^{-3}		9.59×10^{-3}	
		64	8.28×10^{-5}	2.05	3.68×10^{-4}	1.77	1.20×10^{-3}	1.99	2.41×10^{-3}	2.00
		128	2.12×10^{-5}	1.97	1.18×10^{-4}	1.64	3.01×10^{-4}	2.00	6.02×10^{-4}	2.00
		256	5.21×10^{-6}	2.02	3.08×10^{-5}	1.94	7.53×10^{-5}	2.00	1.51×10^{-4}	2.00
		512	1.31×10^{-6}	1.99	8.23×10^{-6}	1.90	1.88×10^{-5}	2.00	3.76×10^{-5}	2.00

Table 2: This figure evaluates the accuracy of APIC and PolyPIC transfers across **one** round trip from grid to particles and back to grid. Both quadratic and cubic B-splines basis functions are evaluated. APIC is consistently second order accurate regardless of basis function. PolyPIC is third order when used with cubic B-splines, but the convergence order breaks down for quadratic B-splines.

be used with quadratic splines. We recommend using cubic splines instead, which also have the advantage of simpler transfers. We analyze the source of the instability in more detail in Section 3.1, where we also show that cubic splines do not suffer from this problem.

4.3. PolyPIC vs APIC transfer accuracy under refinement

In this test, we use the same setup as the previous test, except that we perform a proper refinement study. We simulate to a final time of $T = 1$ with all parts of the algorithm disabled except transfers between particles and grid. This allows us to observe the consequences of repeated transfers between particles and grid under refinement. The results are shown in table 3. Although we might expect that APIC is $O(\Delta x^2/\triangle t)$ and PolyPIC is $O(\Delta x^3/\triangle t)$, the actual convergence order is observed to be slightly higher than this, around 1.7 for APIC and 2.7 for PolyPIC. This is consistent with the errors being the result of filtering of high-frequency information from the grid that is not accurately represented on particles. Once this information is already filtered, subsequent transfers make less error. Thus, the accuracy observed is somewhat higher than our pessimistic estimate might suggest. For PolyPIC with quadratic splines, we again observe error divergence at higher refinement levels.

4.4. PolyPIC vs APIC simulation and spline order

In this test, we compare PolyPIC with APIC, as well as cubic splines with quadratic splines on a full simulation. We choose an *arbitrary* analytic velocity and pressure fields

$$\mathbf{u} = \begin{pmatrix} 2\cos\left(t + \frac{\pi}{6}\right)\sin(2y)\cos(x) + \frac{1}{5}e^t\cos(y) \\ -\cos\left(t + \frac{\pi}{6}\right)\sin(x)\cos(2y) + \frac{1}{5}(1 - t + 5t^2)\sin(x) \end{pmatrix} \qquad p(\mathbf{x}, t) = \sin\left(t - \frac{\pi}{5}\right)e^{\cos(2x)\cos(y) - t}, \tag{86}$$

			u_{ia}-L_2		u_{ia}-L_∞		$\mathbf{v}_p$-L_2		$\mathbf{v}_p$-L_∞	
		res	error	order	error	order	error	order	error	order
PolyPIC	cubic	32	9.25×10^{-3}		1.89×10^{-2}		9.32×10^{-3}		1.89×10^{-2}	
		64	2.37×10^{-3}	1.96	4.82×10^{-3}	1.97	2.39×10^{-3}	1.96	4.81×10^{-3}	1.97
		128	5.99×10^{-4}	1.99	1.21×10^{-3}	1.99	6.01×10^{-4}	1.99	1.21×10^{-3}	1.99
		256	1.50×10^{-4}	2.00	3.02×10^{-4}	2.00	1.51×10^{-4}	2.00	3.02×10^{-4}	2.00
		512	3.76×10^{-5}	2.00	7.57×10^{-5}	2.00	3.76×10^{-5}	2.00	7.57×10^{-5}	2.00
	quadratic	32	7.00×10^{-3}		1.54×10^{-2}		7.06×10^{-3}		1.53×10^{-2}	
		64	1.79×10^{-3}	1.97	3.74×10^{-3}	2.04	1.79×10^{-3}	1.98	3.76×10^{-3}	2.02
		128	4.50×10^{-4}	1.99	9.49×10^{-4}	1.98	4.51×10^{-4}	1.99	9.48×10^{-4}	1.99
		256	1.13×10^{-4}	2.00	2.34×10^{-4}	2.02	1.13×10^{-4}	2.00	3.48×10^{-3}	-1.88
		512	2.82×10^{-5}	2.00	5.90×10^{-5}	1.99	2.84×10^{-5}	1.99	4.29×10^{-3}	-0.30
APIC	cubic	32	1.63×10^{-3}		6.30×10^{-3}		5.31×10^{-3}		1.24×10^{-2}	
		64	9.99×10^{-4}	0.71	3.37×10^{-3}	0.90	7.20×10^{-4}	2.88	2.66×10^{-3}	2.22
		128	3.38×10^{-4}	1.57	1.15×10^{-3}	1.55	1.21×10^{-4}	2.57	5.77×10^{-4}	2.20
		256	9.62×10^{-5}	1.81	3.35×10^{-4}	1.78	2.84×10^{-5}	2.10	1.67×10^{-4}	1.79
		512	2.58×10^{-5}	1.90	1.07×10^{-4}	1.65	8.16×10^{-6}	1.80	5.09×10^{-5}	1.71
	quadratic	32	2.34×10^{-3}		9.50×10^{-3}		2.95×10^{-3}		8.09×10^{-3}	
		64	9.24×10^{-4}	1.34	3.53×10^{-3}	1.43	4.46×10^{-4}	2.73	2.18×10^{-3}	1.89
		128	2.82×10^{-4}	1.71	1.07×10^{-3}	1.72	9.83×10^{-5}	2.18	5.10×10^{-4}	2.10
		256	7.77×10^{-5}	1.86	3.20×10^{-4}	1.74	2.72×10^{-5}	1.85	1.63×10^{-4}	1.65
		512	2.07×10^{-5}	1.91	9.83×10^{-5}	1.70	8.14×10^{-6}	1.74	5.54×10^{-5}	1.56

Table 3: This figure evaluates the performance of APIC and PolyPIC transfers with quadratic and cubic B-spline basis functions in the absence of pressure, viscosity, or other forces. PolyPIC with cubic B-splines is second order accurate, as would be expected from the combination of second order BDF-2 temporal discretization and third order accurate transfers. As in the single-transfer case, PolyPIC with quadratic B-splines breaks down. Of interesting note here is the convergence of APIC, which is noticeably less than second order and appears to be somewhere between order 1.5 and 2.0.

where $\mathbf{u}$ is divergence free. Since these fields are not a solution to the Navier-Stokes equations, we use the method of manufactured solutions [22] and add a body force $\mathbf{f}$ that is analytically computed to make these arbitrary fields satisfy the Navier-Stokes equations

$$\mathbf{f} = \rho \frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p - \mu \nabla^2 \mathbf{u}. \tag{87}$$

We use a constant density $\rho = 1$ and constant viscosity $\nu = 0.2$ over the entire domain. The results are shown in table 4. Here we see that the results of the refinement study with only transfers extends to full simulations. APIC is around order 1.5 accurate in practice. To get full second order accuracy, PolyPIC is required. We also observe the same convergence problem of PolyPIC with quadratic splines. We do not consider quadratic splines further.

4.5. Particle noise and dissipation

In this test, we reproduce the inlet example from Section 5.2.3 of [8]. The 2D version of this test uses a $[0, 1]^2$ domain with inlets and outflows along the bottom wall ($y = 0$). The inlet is at $\frac{1}{2} \leq x \leq \frac{3}{4}$, and the outlets are at $\frac{11}{80} \leq x \leq \frac{21}{80}$ and $\frac{67}{80} \leq x \leq \frac{77}{80}$. The 3D version of this test uses a $[0, 1]^3$ domain with inlets and outflows along the bottom wall ($z = 0$). The inlet is at $\frac{1}{2} \leq x, y \leq \frac{3}{4}$, and the outlets are at $\frac{11}{80} \leq x, y \leq \frac{21}{80}$ and $\frac{67}{80} \leq x, y \leq \frac{77}{80}$. See [8] for illustrations showing the domain layout. During the first 80 of simulation, fluid is pumped in through the inlet at a constant 0.2. After that time, the inlet is set to zero velocity. The outflows are implemented as Dirichlet pressure ($p = 0$) boundary conditions through the entire simulation. The simulation is run for 200. Particle kinetic energy for

			u_{ia}-L_2		u_{ia}-L_∞		$\mathbf{v}_p$-L_2		$\mathbf{v}_p$-L_∞	
		res	error	order	error	order	error	order	error	order
PolyPIC	cubic	32	1.02×10^{-2}		2.51×10^{-2}		1.02×10^{-2}		2.51×10^{-2}	
		64	2.66×10^{-3}	1.94	6.59×10^{-3}	1.93	2.67×10^{-3}	1.94	6.59×10^{-3}	1.93
		128	6.72×10^{-4}	1.99	1.66×10^{-3}	1.99	6.72×10^{-4}	1.99	1.66×10^{-3}	1.99
		256	1.68×10^{-4}	2.00	4.15×10^{-4}	2.00	1.69×10^{-4}	2.00	4.15×10^{-4}	2.00
		512	4.22×10^{-5}	2.00	1.04×10^{-4}	2.00	4.22×10^{-5}	2.00	1.04×10^{-4}	2.00
	quadratic	32	8.49×10^{-3}		2.08×10^{-2}		8.51×10^{-3}		2.08×10^{-2}	
		64	2.18×10^{-3}	1.96	5.36×10^{-3}	1.96	2.18×10^{-3}	1.96	5.36×10^{-3}	1.96
		128	5.49×10^{-4}	1.99	1.34×10^{-3}	1.99	5.48×10^{-4}	1.99	1.35×10^{-3}	1.99
		256	1.44×10^{-4}	1.93	3.58×10^{-4}	1.91	1.44×10^{-4}	1.93	3.59×10^{-4}	1.91
		512	3.55×10^{-5}	2.02	1.57×10^{-4}	1.19	3.63×10^{-5}	1.99	9.69×10^{-3}	-4.76
APIC	cubic	32	4.28×10^{-3}		1.02×10^{-2}		5.24×10^{-3}		1.41×10^{-2}	
		64	1.17×10^{-3}	1.87	2.74×10^{-3}	1.90	9.00×10^{-4}	2.54	2.34×10^{-3}	2.59
		128	5.67×10^{-4}	1.05	1.43×10^{-3}	0.94	4.82×10^{-4}	0.90	1.22×10^{-3}	0.94
		256	2.03×10^{-4}	1.48	5.16×10^{-4}	1.47	1.80×10^{-4}	1.42	4.52×10^{-4}	1.44
		512	7.14×10^{-5}	1.51	1.84×10^{-4}	1.49	6.50×10^{-5}	1.47	1.63×10^{-4}	1.47
	quadratic	32	2.44×10^{-3}		7.35×10^{-3}		1.76×10^{-3}		4.69×10^{-3}	
		64	1.46×10^{-3}	0.74	3.54×10^{-3}	1.06	1.18×10^{-3}	0.58	3.00×10^{-3}	0.64
		128	5.37×10^{-4}	1.44	1.34×10^{-3}	1.40	4.65×10^{-4}	1.34	1.17×10^{-3}	1.36
		256	1.79×10^{-4}	1.59	4.51×10^{-4}	1.57	1.60×10^{-4}	1.54	3.96×10^{-4}	1.56
		512	6.15×10^{-5}	1.54	1.58×10^{-4}	1.52	5.64×10^{-5}	1.50	1.42×10^{-4}	1.48

Table 4: In this table, the full Navier-Stokes equations are simulated using all four transfer combinations evaluated in this paper. Including the full set of forces does not change the results for PolyPIC; cubic splines are second order and quadratic splines encounter convergence problems. The APIC convergence orders are now a bit lower and are convincingly 1.5.

PolyPIC is computed as described in Section 3.5. Otherwise, quantities are computed as in [8]. This test was run in five configurations. First, the test was run without using the second order in time discretization (the scheme used to start the multistep method) using APIC, PolyPIC, and FLIP transfers. Then, the test was repeated for APIC, PolyPIC transfers but using the BDF-2 discretization. Using FLIP transfers with the BDF-2 discretization is not a meaningful scheme, so this configuration was not run. All of the simulations were run at 64^d resolution. As a neutral point of reference, we have also run the same test using an Eulerian solver at resolution 128^2 in 2D and both 64^3 and 96^3 in 3D. The Eulerian solver uses BDF-2 time integration with predicted pressures and fifth order WENO advection.

2D. Figure 2 shows the vorticity and kinetic energy in the 2D case. Note that the Eulerian scheme has a significantly different qualitative profile compared to the particle-based schemes. Until about $t = 20$, the kinetic energy closely tracks the proposed scheme, after which the two schemes begin to deviate. The Eulerian scheme continues to accumulate energy for longer before stabilizing. Overall, the Eulerian scheme accumulates significantly more energy than the other schemes, and it retains more of that energy when the source is closed. This suggests that the purely Eulerian scheme is less dissipative. It is worth noting that the Eulerian scheme was run at a higher resolution for reference, though the results at lower resolutions are qualitatively similar. Although the equations being solved in this example are inviscid, the numerical schemes employed are not. There is a small amount of dissipation inherent in the numerical methods, which sets the effective Reynolds number and thus the detailed flow and vortex structure for the solutions.

The next observation is that FLIP always has the most energy of the hybrid schemes, which is not especially surprising, since repeated grid-particle-grid transfers would be expected to dissipate energy. The next observation is that PolyPIC is significantly less dissipative than APIC. This is also expected, since PolyPIC transfers better approximate

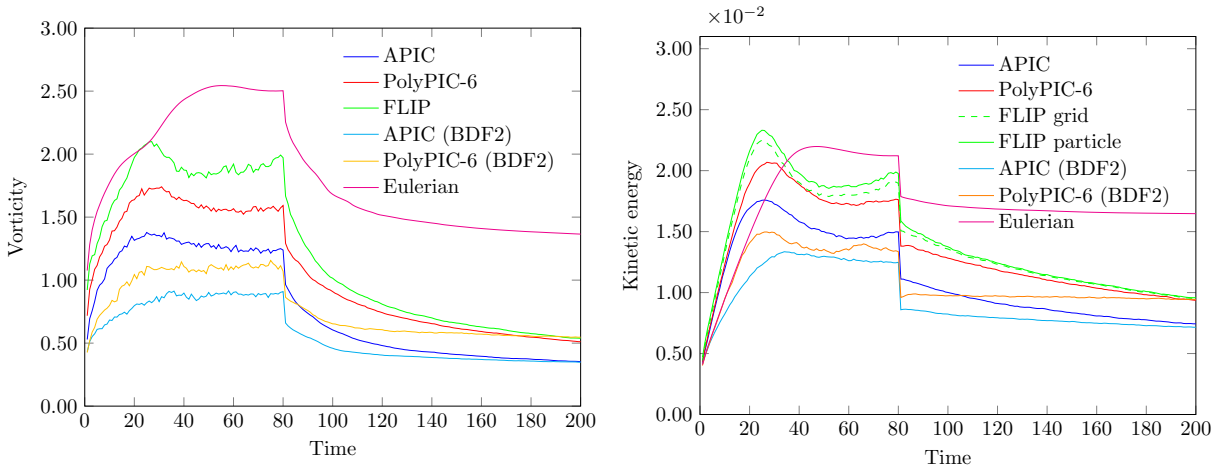

Fig. 2: This figure shows the vorticity and kinetic energy for the **2D** inlet test. During the first 80 s, fluid is pumped in through the inlet at the bottom of the domain and is allowed to flow out passively through outflow holes. After this time, the pump is stopped and the fluid continues to circulate.

the velocity field being transferred and therefore lose less energy. Perhaps surprising, however, is that the second order in time BDF-2 discretization is actually less energetic than the first order discretization. Since kinetic energy is readily computed on both the grid and the particles, one may compare the energy on the particles with the energy observed on the grid. In the case of FLIP transfers, there is a small discrepancy between the two, which decays away once the pump is stopped. In all other cases, the grid and particle energies are so close that the lines would be indistinguishable in the figure, so they have been omitted.

3D. Figure 3 shows the vorticity and kinetic energy in the 3D case. In this case, the differences between the different transfers are much less significant, though BDF-2 retains notably less vorticity while maintaining similar energy. Note that the grid measure of kinetic energy for FLIP (dashed green line) is similar to the other schemes. The particle measure of kinetic energy, however, is significantly higher than the grid measure. This indicates that the particles are accumulating a great deal of noise. This is the reason velocity filtering is frequently used with FLIP transfers [9]. It is interesting to note that in the 3D version of this test, BDF-2 has only a very modest impact on the results. By contrast, APIC retains far less vorticity than the other schemes. In particular, PolyPIC retains a comparable amount of energy and vorticity as FLIP on this test.

In the 3D case, we have plotted the results for the Eulerian scheme at higher resolution (96^3) and the same resolution as the hybrid schemes (64^3). The The kinetic energy of the Eulerian schemes are similar across resolutions, and they are similar to the hybrid schemes, though the Eulerian scheme is observed to be less dissipative. The vorticity, however, looks quite different, with the vorticity changing markedly with resolution. This is a result of changes in the detailed vortex structures, which vary by resolution and change over time during the simulation.

Quantifying noise. We can estimate the level of particle noise present in each scheme by dividing the grid kinetic energy by the particle kinetic energy. Ideally, this ratio will be 1, since both are supposed to be representations of the same velocity field. In all of the APIC and PolyPIC simulations, this ratio stays between 0.999 and 1.007, which suggests particles are not storing velocity modes that are at a higher frequency than the grid and that the particle-based kinetic energy formulas for these methods are very accurate.

In the case of FLIP in 2D, the ratio is lowest at the beginning at 0.94 and rises to 0.96 by time 80 (when the source is shut off), and then rises further to 0.99 by the end of the simulation. As the velocity field becomes smoother, less energy is lost during the particle-to-grid transfer, and the ratio climbs closer to 1. On the 2D version of the simulation, FLIP does not accumulate significant noise. This provides a convenient point of reference for comparison.

In the case of FLIP in 3D, the ratio at the beginning is also 0.94; as in 2D, this can be assumed to be free of any significant noise. By the time the source is shut off, the ratio has fallen to 0.61, indicating that 39% of the kinetic energy on particles is lost during the transfer. Using the initial ratio as an estimate of the expected averaging losses in the absence of noise, we conclude that about a third of the particle energy is high-frequency noise. By the end of

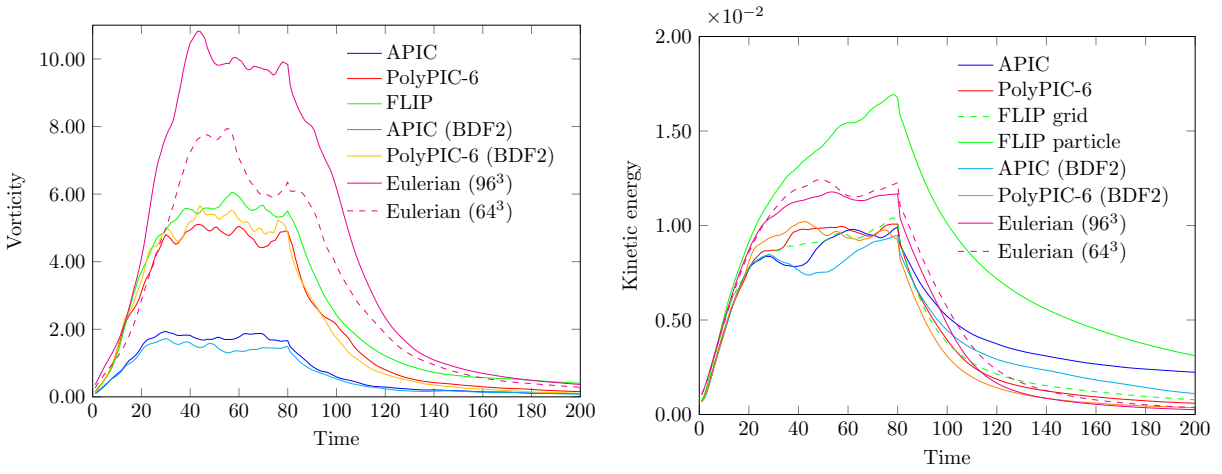

Fig. 3: This figure shows the vorticity and kinetic energy for the **3D** inlet test. During the first 80 s, fluid is pumped in through the inlet at the bottom of the domain and is allowed to flow out passively through outflow holes. After this time, the pump is stopped and the fluid continues to circulate. Note that the particle and grid measures of kinetic energy differ significantly for FLIP transfers on this test, with up to 1/3 of the particle energy being noise.

the simulation, the ratio falls to 0.25, which is its lowest value during the entire simulation; about three-quarters of the particles' kinetic energy is noise. Although both real kinetic energy and particle noise decay once the source is closed, particle noise is invisible to the grid and dissipates less efficiently, causing the noise fraction to rise.

4.6. Taylor-Green

Here we perform a refinement study on a Tayler-Green vortex. The analytic velocity and pressure fields are

$$\mathbf{u} = e^{-\nu t}\begin{pmatrix}\cos(y)\sin(x)\\ -\sin(y)\cos(x)\end{pmatrix} \qquad p(\mathbf{x},t) = \frac{1}{4}e^{-2\nu t}(\cos(2x)+\cos(2y)) \tag{88}$$

on a $[-\pi,\pi]^2$ domain with periodic boundaries. The initial velocity field is $\mathbf{u}_0(\mathbf{x}) = \mathbf{u}(\mathbf{x},0)$. This is a solution to the Navier-Stokes equations, so we don't apply body forces. The fluid has physical properties $\rho = 1$ and $\nu = 0.001$. Note that we use a small viscosity coefficient so that decay is relevant to the convergence order but does not create artificially low velocity errors by decaying the velocity too much. We use the same grid resolution and time step sizes as the previous tests. The results are shown in table 5, which demonstrates second-order accuracy on all velocity error measures.

4.7. Non-periodic

We focus on periodic tests in this paper to avoid the complexities of second order accurate treatments for boundary conditions for our pressure and viscosity solvers. Unlike MLS-type schemes, the lack of full particle coverage does

PolyPIC+cubic

	$\mathbf{u}_{ia}$-L_2		$\mathbf{u}_{ia}$-L_∞		$\mathbf{v}_p$-L_2		$\mathbf{v}_p$-L_∞	
res	error	order	error	order	error	order	error	order
32	5.90×10^{-3}		1.26×10^{-2}		5.89×10^{-3}		1.26×10^{-2}	
64	1.50×10^{-3}	1.97	3.15×10^{-3}	2.00	1.51×10^{-3}	1.96	3.14×10^{-3}	2.00
128	3.80×10^{-4}	1.98	8.07×10^{-4}	1.96	3.81×10^{-4}	1.99	8.08×10^{-4}	1.96
256	9.52×10^{-5}	2.00	1.98×10^{-4}	2.03	9.53×10^{-5}	2.00	1.98×10^{-4}	2.03
512	2.38×10^{-5}	2.00	4.98×10^{-5}	1.99	2.38×10^{-5}	2.00	4.98×10^{-5}	1.99

Table 5: This table shows the convergence results for the Taylor-Green vortex. PolyPIC transfers with cubic B-spline basis functions were used. Second order accuracy is observed.

not pose special problems for PolyPIC (or APIC); see the discussion in Section 3.2. In this test, we show that second order accuracy is achieved as long as the finite difference discretizations of pressure and viscosity are second order, even though the edges and corners of the domain have incomplete particle coverage. We do need to be careful to ensure that grid velocities in a band outside the fluid region are filled before the grid-to-particle transfers; for this we use the same reflection-based treatment as [8].

We perform the "square" test from [8]. In this test, two stream functions are defined as

$$\phi_1(\mathbf{x}) = f(x)f(y) \qquad \phi_2(\mathbf{x}) = f(x)g(y) \qquad f(x) = x(1-x)(x^2-x-1) \qquad g(x) = x(1-x)(x+1)(3x^2-7) \tag{89}$$

The analytic velocity field is constructed form these stream functions as $\mathbf{u} = \langle -\frac{\partial\phi_1}{\partial y}, \frac{\partial\phi_1}{\partial x}\rangle + t\langle -\frac{\partial\phi_2}{\partial y}, \frac{\partial\phi_2}{\partial x}\rangle$ on a $[0,1]^2$ domain with slip boundary conditions. This velocity field is divergence free. On the boundary, the velocities are zero along the normal direction to the boundary, compatible with the slip boundary condition. The pressure field is $p(\mathbf{x},t) = xy(1-x)(1-y)(x-xy+y^2+t)$, which is zero at the boundaries. Unlike the inviscid setup in [8], we use $\rho = 1$ and $\nu = 0.1$. Grid resolution and time step sizes are the same as in previous tests. The results are shown in table 6. We can see that second-order convergence is achieved with slip boundary conditions.

For more complex boundaries, a more accurate spatial discretization for pressure and viscosity would be required. Particle transfers are generally unaware of and unaffected by the type and shape of boundary conditions, other than the effects it may have on particle arrangement and coverage. Since our focus is on transfer accuracy, we do not pursue other types of boundary conditions or curved boundaries in this paper.

4.8. 3D Taylor-Green

Our proposed second-order scheme extends automatically to 3D. In this test case, we demonstrate second order accuracy on a 3D Taylor-Green vortex. We adopt a 3D Taylor-Green formulation

$$\mathbf{u} = e^{-\nu t}\begin{pmatrix}\sin(z)+\cos(y)\\ \sin(x)+\cos(z)\\ \sin(y)+\cos(x)\end{pmatrix} \qquad p = -e^{-2\nu t}\left(\sin(z)\cos(y)+\sin(x)\cos(z)+\sin(y)\cos(x)\right). \tag{90}$$

This is a solution to the Navier-Stokes equations, so no extra forces are applied. We use $\rho = 1$ and $\nu = 0.1$. As shown in table 7, our method demonstrates second-order accuracy.

4.9. 3D manufactured solution

To test convergence in 3D more robustly, we set up *arbitrary* analytic velocity field and pressure fields as we did in section 4.4, where

$$\mathbf{u} = \begin{pmatrix} 2\cos\left(t+\frac{\pi}{6}\right)\cos(2x)\sin(3y)\sin(z) + \frac{1}{5}e^t\cos(y)\\ -\cos\left(t+\frac{\pi}{6}\right)\sin(2x)\cos(3y)\sin(z) + \frac{1}{5}(1-t+10t^2)\sin(z)\\ -\cos\left(t+\frac{\pi}{6}\right)\sin(2x)\sin(3y)\cos(z) + \frac{1}{5}(1-t+10t^2)\sin(x)\end{pmatrix} \qquad p = \sin\left(t-\frac{\pi}{5}\right)e^{\cos(2x)\cos(y)\sin(3z)-t} \tag{91}$$

As before, we use density $\rho = 1$ and viscosity $\nu = 0.1$. The results are shown in table 8, which also demonstrates second-order accuracy in velocities.

PolyPIC+cubic

	$\mathbf{u}_{ia}$-L_2		$\mathbf{u}_{ia}$-L_∞		$\mathbf{v}_p$-L_2		$\mathbf{v}_p$-L_∞	
res	error	order	error	order	error	order	error	order
32	2.99×10^{-2}		6.34×10^{-2}		3.15×10^{-2}		6.35×10^{-2}	
64	8.03×10^{-3}	1.90	1.71×10^{-2}	1.89	8.28×10^{-3}	1.93	1.71×10^{-2}	1.89
128	2.05×10^{-3}	1.97	4.36×10^{-3}	1.97	2.09×10^{-3}	1.99	4.36×10^{-3}	1.97
256	5.17×10^{-4}	1.99	1.10×10^{-3}	1.99	5.22×10^{-4}	2.00	1.10×10^{-3}	1.99
512	1.30×10^{-4}	2.00	2.74×10^{-4}	2.00	1.30×10^{-4}	2.00	2.74×10^{-4}	2.00

Table 6: Convergence results for the full Navier-Stokes equations on non-periodic test case with slip boundary conditions. PolyPIC with cubic B-splines were used, and second order convergence is observed. Although the number of neighbor particles will be reduced near the domain walls, and especially corners, the convergence is not affected.

5. Conclusion

We have presented a second order accurate discretization of the Navier-Stokes equations within the framework of a particle-in-cell method. The scheme avoids generating particle noise by using direct transfers of velocities from grid to particles instead of accelerations, which avoids the need for stabilization. We have provided explicit closed-form solutions for quadratic PolyPIC transfers.

Performing velocity transfers from grid to particle reduces convergence order by about half an order in practice. So while APIC and quadratic PolyPIC have transfer errors of $O(\Delta x^2)$ and $O(\Delta x^3)$ across a single transfer, when used over many time steps, the order drops to about $O(\Delta x^{1.5})$ and $O(\Delta x^{2.5})$. This means that achieving second order accuracy requires the use of quadratic PolyPIC transfers instead of much simpler APIC transfers.

We also repeated the Fourier analysis from [8] on quadratic PolyPIC to qualitatively characterize its dissipation relative to alternative transfers and found it significantly less dissipative than APIC (originally noted by [10]) and comparable to XPIC 5.

5.1. Limitations

Following Edwards and Bridson [9], this work focuses on particle-to-grid and grid-to-particle transfers and their effect on the order of accuracy of a Navier-Stokes fluid solver. As such, we have neglected many complications, including support for irregular boundary conditions, moving boundaries, and free surfaces. These do not create significant problems for PolyPIC particles transfers (other than filling ghost cells), but they very significantly complicate the grid discretizations for pressure and viscosity.

The need for an extra polynomial order would normally be very expensive in the context of weighted least squares. We must also perform more reconstructions, since there are typically many particles per grid cell. However, because of the regularity of the transfers, we were able to perform the reconstruction in closed form, which significantly reduces the computational expense. In our implementation, the grid-to-particle transfer (where the polynomial reconstruction occurs) is actually less expensive than the particle-to-grid transfer.

Unlike MLS, which throws the polynomial reconstruction away immediately after use, PolyPIC must store these polynomial reconstructions for each particle for use during the particle-to-grid transfers. This significantly increases the memory footprint and corresponding cost of our transfers.

Edwards and Bridson [9] considered schemes as accurate as fourth order. Weighted least squares extends naturally to any polynomial order, allowing them to achieve higher convergence orders without obvious difficulty. They are able to choose a convenient and inexpensive kernel, and increasing the polynomial order is a simple matter of increasing the kernel's support radius. Of course, the cost of the transfers grows rapidly with increased polynomial order. PolyPIC also naturally extends to higher order polynomials, but the system that must be solved grows rapidly in size and becomes less sparse. Implementing cubic PolyPIC in closed form, while entirely feasible, is significantly more complicated. It would also require switching to quartic B-spline basis functions in order to ensure enough neighboring grid velocities to perform the polynomial reconstruction. Quartic PolyPIC transfers (which would be needed for fourth order convergence) would be even more complex to implement and require quintic B-spline basis functions.

6. Acknowledgements

This work was supported in part by National Science Foundation award NSF-2006570 as well as University of California award M23PL6076.

PolyPIC+cubic

	$\mathbf{u}_{ia}$-L_2		$\mathbf{u}_{ia}$-L_∞		$\mathbf{v}_p$-L_2		$\mathbf{v}_p$-L_∞	
res	error	order	error	order	error	order	error	order
16	2.18×10^{-2}		4.32×10^{-2}		2.15×10^{-2}		4.31×10^{-2}	
32	5.66×10^{-3}	1.94	1.12×10^{-2}	1.94	5.65×10^{-3}	1.93	1.12×10^{-2}	1.94
64	1.43×10^{-3}	1.99	2.81×10^{-3}	2.00	1.43×10^{-3}	1.99	2.82×10^{-3}	2.00
128	3.57×10^{-4}	2.00	6.93×10^{-4}	2.02	3.57×10^{-4}	2.00	6.93×10^{-4}	2.02

Table 7: This table shows the convergence results for the 3D Taylor-Green vortex. PolyPIC transfers with cubic B-spline basis functions were used, and second order accuracy is observed.

PolyPIC+cubic

	$\mathbf{u}_{ia}$-L_2		$\mathbf{u}_{ia}$-L_∞		$\mathbf{v}_p$-L_2		$\mathbf{v}_p$-L_∞	
res	error	order	error	order	error	order	error	order
16	1.03×10^{-1}		4.48×10^{-1}		1.16×10^{-1}		5.67×10^{-1}	
32	2.91×10^{-2}	1.83	1.44×10^{-1}	1.63	3.08×10^{-2}	1.92	1.49×10^{-1}	1.92
64	7.97×10^{-3}	1.87	4.00×10^{-2}	1.85	8.18×10^{-3}	1.91	4.04×10^{-2}	1.89
128	5.79×10^{-4}	3.78	2.16×10^{-3}	4.21	5.79×10^{-4}	3.82	2.21×10^{-3}	4.19

Table 8: This test shows the convergence orders for a fully general 3D Navier Stokes test constructed using the method of manufactured solutions.

References

[1] SG. Bardenhagen and EM. Kober. The generalized interpolation material point method. *Comp Mod in Eng and Sci*, 5(6):477–496, 2004.

[2] Y Bing, M Cortis, TJ Charlton, WM Coombs, and CE Augarde. B-spline based boundary conditions in the material point method. *Computers & Structures*, 212:257–274, 2019.

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

[4] J.U. Brackbill and G. Lapenta. Particle-in-cell magnetohydrodynamics. In *16th Int Conf on the Numer Sim of Plasmas*, 1998.

[5] R. Bridson. Fast poisson disk sampling in arbitrary dimensions. In *SIGGRAPH sketches*, page 22, 2007.

[6] Alban de Vaucorbeil, Vinh Phu Nguyen, Sina Sinaie, and Jian Ying Wu. Material point method after 25 years: theory, implementation and applications. *Submitted to Advances in Applied Mechanics*, page 1, 2019.

[7] Alban de Vaucorbeil, Vinh Phu Nguyen, and Christopher R Hutchinson. A total-lagrangian material point method for solid mechanics problems involving large deformations. *Computer Methods in Applied Mechanics and Engineering*, 360:112783, 2020.

[8] O. Ding, T. Shinar, and C. Schroeder. Affine particle in cell method for mac grids and fluid simulation. *Journal of Computational Physics*, 408: 109311, 2020. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2020.109311. URL `https://www.sciencedirect.com/science/article/pii/S0021999120300851`.

[9] E. Edwards and R. Bridson. A high-order accurate particle-in-cell method. *Int J Numer Meth Eng*, 90:1073–1088, 2012.

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

[11] Yong Gan, Zheng Sun, Zhen Chen, Xiong Zhang, and Yu Liu. Enhancement of the material point method using b-spline basis functions. *International Journal for numerical methods in engineering*, 113(3):411–431, 2018.

[12] C. Gritton. *Ringing Instabilities in Particle Methods*. PhD thesis, The University of Utah, 2014.

[13] Chad C Hammerquist and John A Nairn. A new method for material point method particle updates that reduces noise and enhances stability. *Computer Methods in Applied Mechanics and Engineering*, 318:724–738, 2017.

[14] Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4):150, 2018.

[15] Issam Jassim, Dieter Stolle, and Pieter Vermeer. Two-phase dynamic analysis by material point method. *International journal for numerical and analytical methods in geomechanics*, 37(15):2502–2522, 2013.

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

[17] C. Jiang, C. Schroeder, and J. Teran. An angular momentum conserving affine-particle-in-cell method. *J Comp Phys*, 338:137–164, 2017.

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

[19] Georgios Moutsanidis, Christopher C Long, and Yuri Bazilevs. Iga-mpm: The isogeometric material point method. *Computer Methods in Applied Mechanics and Engineering*, 372:113346, 2020.

[20] Vinh Phu Nguyen, Chi Thanh Nguyen, Timon Rabczuk, and Sundararajan Natarajan. On a family of convected particle domain interpolations in the material point method. *Finite Elements in Analysis and Design*, 126:50–64, 2017.

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

[22] Patrick J Roache. Code verification by the method of manufactured solutions. *Journal of fluids engineering*, 124(1):4–10, 2002.

[23] A Sadeghirad, Rebecca M Brannon, and 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): 1435–1456, 2011.

[24] A Sadeghirad, Rebecca M Brannon, and JE Guilkey. Second-order convected particle domain interpolation (cpdi2) with enrichment for weak discontinuities at material interfaces. *International Journal for numerical methods in Engineering*, 95(11):928–952, 2013.

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

[26] Michael Steffen, Robert M Kirby, and Martin Berzins. Decoupling and balancing of space and time errors in the material point method (mpm). *International journal for numerical methods in engineering*, 82(10):1207–1243, 2010.

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

[28] Deborah Sulsky and Ming Gong. Improving the material-point method. In *Innovative numerical approaches for multi-field and multi-scale problems*, pages 217–240. Springer, 2016.

[29] Quoc Anh Tran, Wojciech Sołowski, Martin Berzins, and James Guilkey. A convected particle least square interpolation material point method. *International Journal for Numerical Methods in Engineering*, 121(6):1068–1100, 2020.

[30] PC Wallstedt and JE Guilkey. A weighted least squares particle-in-cell method for solid mechanics. *Int J Numer Meth Eng*, 85(13):1687–1704, 2011.

[31] Elizaveta Wobbes, Matthias Möller, Vahid Galavi, and Cornelis Vuik. Conservative taylor least squares reconstruction with application to material point methods. *International Journal for Numerical Methods in Engineering*, 117(3):271–290, 2019.

[32] Fan Zhang, Xiong Zhang, Kam Yim Sze, Yanping Lian, and Yan Liu. Incompressible material point method for free surface flow. *Journal of Computational Physics*, 330:92–110, 2017.