A Polynomial Particle-In-Cell Method

CHUYUAN FU, University of California, Los Angeles
QI GUO, University of California, Los Angeles
THEODORE GAST, University of California, Los Angeles
CHENFANFU JIANG, University of Pennsylvania
JOSEPH TERAN, University of California, Los Angeles

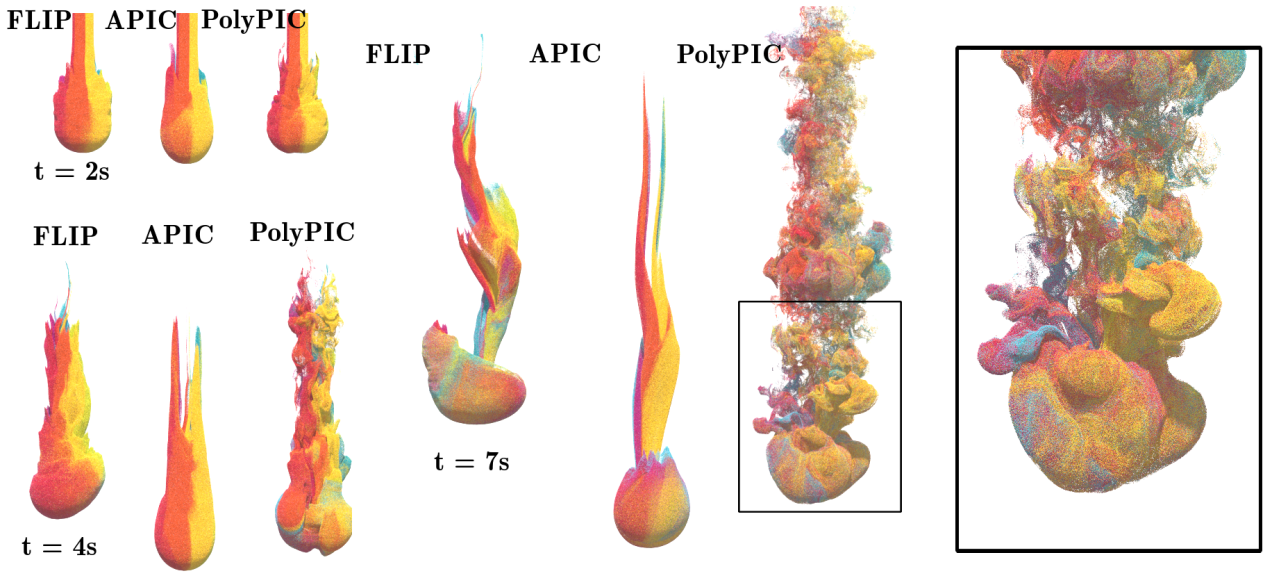


Fig. 1. **Ink drop.** We compare from left to right FLIP, APIC, and PolyPIC for an inkjet in an ambient incompressible fluid. PolyPIC more effectively resolves the vorticial details.

Recently the Affine Particle-In-Cell (APIC) Method was proposed by Jiang et al.[2015; 2017b] to improve the accuracy of the transfers in Particle-In-Cell (PIC) [Harlow 1964] techniques by augmenting each particle with a locally affine, rather than locally constant description of the velocity. This reduced the dissipation of the original PIC without suffering from the noise present in the historic alternative, Fluid-Implicit-Particle (FLIP) [Brackbill and Ruppel 1986]. We present a generalization of APIC by augmenting each particle with a more general local function. By viewing the grid-to-particle transfer as a linear and angular momentum conserving projection of the particle-wise local grid velocities onto a reduced basis, we greatly improve the energy and vorticity conservation over the original APIC. Furthermore, we show that the cost of the generalized projection is negligible over APIC when using a particular class of local polynomial functions. Lastly, we note that our method retains the filtering property of APIC and PIC and thus has similar robustness to noise.

CCS Concepts: • **Computing methodologies** → **Physical simulation**;

Additional Key Words and Phrases: APIC, PIC, FLIP

ACM Reference format:
Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. 2017. A Polynomial Particle-In-Cell Method. *ACM Trans. Graph.* 36, 6, Article 222 (November 2017), 12 pages.
DOI: 10.1145/3130800.3130878

DOI: 10.1145/3130800.3130878

1 INTRODUCTION

Simulation techniques for natural phenomena in computer graphics applications must accommodate a wide range of geometric domains and material behaviors. Lagrangian techniques accurately resolve transport phenomena and allow for simple rendering. However, Eulerian techniques naturally resolve topology change and contact/collision. Particle-In-Cell (PIC) [Harlow 1964] is a hybrid approach designed to attain the benefits of both views, without suffering from their drawbacks. Indeed PIC approaches like Fluid-Implicit-Particle (FLIP) [Brackbill and Ruppel 1986] for incompressible fluids and the Material Point Method (MPM) [Sulsky et al. 1994, 1995] for history dependent materials have proven very effective in graphics applications in recent years.

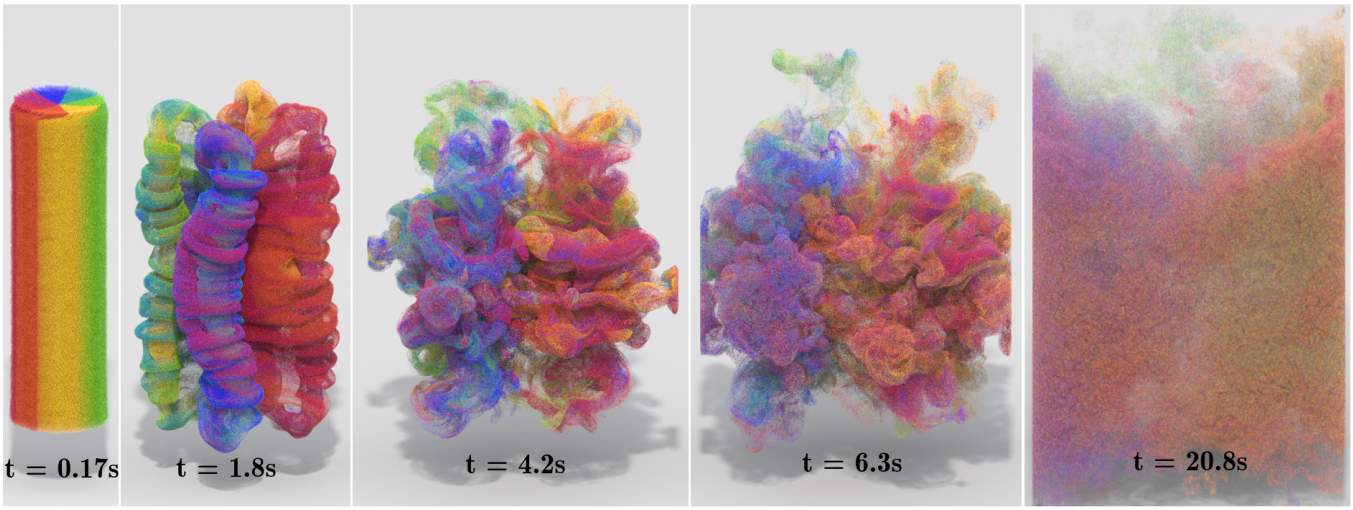


Fig. 2. **Rotating column of colored dust**. We demonstrate intricate vorticial patterns that arise from simple initial conditions with incompressible flow. PolyPIC achieves great detail with modest spatial grid resolution ($88 \times 132 \times 88$). The rightmost image shows that despite the energetic nature of our method, our simulations are stable at long runtimes.

While very powerful, PIC techniques have a number of well known artifacts. Particles store the primary state like mass and momentum, but the effect of internal stress on momentum is added on an Eulerian grid. This is reconciled by transfers back and forth between the particles and grid. There is generally a mismatch in the number of particle and grid degrees of freedom which can lead to errors during the frequent transfers between representations [Brackbill 1988]. The original PIC possesses a stabilizing filter property since particle velocities are interpolated from the grid after the stress response. However, this leads to excessive dissipation since particle modes are essentially overwritten by the generally lower resolution grid. FLIP removes this limitation by interpolating increments in velocity rather than velocity itself as in PIC; however this means that particle modes invisible to the grid persist despite not receiving a meaningful constitutive response. This can lead to particle artifacts like noise, instability, clumping and volume loss/gain. Recently, Jiang et al. [2015; 2017b] developed an Affine Particle-In-Cell (APIC) approach designed to prevent these artifacts, without incurring the excessive dissipation of PIC. The idea is to retain the filtering property, but to prevent dissipation by interpolating more information from the grid to the particles. By allowing particles to store both velocity and velocity derivative information, Jiang et al. design particle/grid transfers that conserve angular momentum and generally attain the benefits of both PIC and FLIP.

We present an improvement to this technique that allows for locally polynomial, rather than locally affine approximations to the grid velocity field: PolyPIC. Our generalization improves kinetic energy conservation during transfers which leads to better vorticity resolution in fluid simulations and less numerical damping in elastoplasticity simulations. Our transfers are designed to select particle-wise polynomial approximations to the grid velocity that are optimal in the local mass-weighted L^2 norm. This is equivalent to the reduced basis APIC derivation in [Jiang et al. 2017b] for affine modes, but generalized to polynomial modes. Indeed our notion of transfers reproduces the original PIC if only constants are used and APIC if only affine polynomials are used. Furthermore, we derive a polynomial basis that is mass-orthogonal to facilitate rapid solution of the optimality condition. By design, this reduces the projection to the polynomial basis to the solution of a *diagonal* linear system of size equal to the number of local polynomial modes. This has the added benefit of simplifying applications with staggered grids. The original APIC used a modified approximation to the linear part of the affine velocity in the case of staggered grids. We summarize our contributions as:

- A generalization of APIC from locally affine to locally polynomial representations that improves kinetic energy conservation in particle/grid transfers.
- A mass weighted L^2 optimality condition that achieves linear and angular momentum conservation.
- A mass-orthogonal class of polynomials for rapid solution of projection to the polynomial basis.
- Natural treatment of staggered and collocated grids.

We demonstrate the benefits of our technique in a number of representative applications of incompressible flow and MPM simulation of elastoplastic materials.

2 PREVIOUS WORK

PIC techniques have been widely adopted in computer graphics applications. Zhu and Bridson first used FLIP for sand and liquids in [Zhu and Bridson 2005]. Foster and Metaxas used particles for tracking liquids in [Foster and Metaxas 1996]. Most applications in computer graphics are for incompressible flows using a MAC-grid [Harlow and Welch 1965] pressure projection to enforce incompressibility in the grid momentum update [Batty et al. 2007; Batty and Bridson 2008; Boyd and Bridson 2012; Larionov et al. 2017; McAdams et al. 2009; Zhang et al. 2016]. Unilateral incompressibility, where a

divergence-inequality constraint replaces the divergence-free constraint, has been used with FLIP for a wide range of applications. Narain et al. use this for sand [2010] and crowd dynamics [2009]. Daviet et al. [2016] also use this technique for sand. Gerszewski and Bargteil use mass-full FLIP with a unilateral incompressibility constraint to resolve large-scale splashing liquids [Gerszewski and Bargteil 2013].

MPM has been used for a wide range of graphics applications involving more esoteric materials including snow [Stomakhin et al. 2013], sand [Daviet and Bertails-Descoubes 2016; Klár et al. 2016; Tampubolon et al. 2017], foam and complex fluids [Ram et al. 2015; Yue et al. 2015], large strain elasticity [Zhu et al. 2016], cloth and fiber collisions [Jiang et al. 2017a], and phase change [Stomakhin et al. 2014].

Various works have improved or modified aspects of the standard PIC techniques commonly used in graphics. Edwards and Bridson investigated higher-order accuracy [Edwards and Bridson 2012]. Narrow band and adaptive particle sampling techniques as well as adaptive/unstructured Eulerian grids dramatically increase efficiency [Ando et al. 2012, 2013; Ando and Tsuruno 2011; Ferstl et al. 2016; Hong et al. 2008, 2009]. Ando et al. [2015] use a stream function to enforce incompressibility, rather than the more commonly used MAC projection. Mercier et al. [2015] increase apparent resolution of FLIP with secondary surface wave simulation. Adaptive shallow water height field and FLIP coupling achieve impressive simulation rates [Chentanez and Muller 2014]. Smoothed-particle hydrodynamics (SPH) [Desbrun and Cani 1996] has proven very powerful for graphics applications. Several works couple SPH with PIC techniques to improve aspects like performance, memory usage and discrete incompressibility [Gao et al. 2009; Hong et al. 2008; Lee et al. 2009; Losasso et al. 2008; Raveendran et al. 2011; Zhu et al. 2010].

There are a number of recent PIC approaches designed to improve robustness to noise without sacrificing accurate energy and momentum conservation. Hammerquist and Nairn [2017] developed a PIC extension designed to reduce the noise of the FLIP by adding a smoothing term to the FLIP velocity. This strikes a good balance between noise reduction and energy preservation. Edwards and Bridson also add a regularization term to diminish particle noise [Edwards and Bridson 2012]. Gritton and Berzins [2017] reduce noise by filtering spatial gradients based on a local SVD approximation of the null space of the particle-to-grid transfer operator. Wallstedt and Guilkey use a locally-affine assumption as in [2015; 2017b], but they use FLIP grid-to-particle transfers that still suffer from noise [Wallstedt and Guilkey 2007]. Um et al. develop a particle repulsion force to improve particle bunching associated with the ringing instability [Um et al. 2014].

3 METHOD OUTLINE AND NOTATION

Our method is concerned with the update of the Lagrangian quantities in PIC calculations. We discuss this in detail and give an overview of each step in the process in Section (§6). However, we first motivate our generalized notion of velocity local to a particle in Section (§4) as well as the connection of our method to the very useful class of updated Lagrangian techniques in Section (§5).

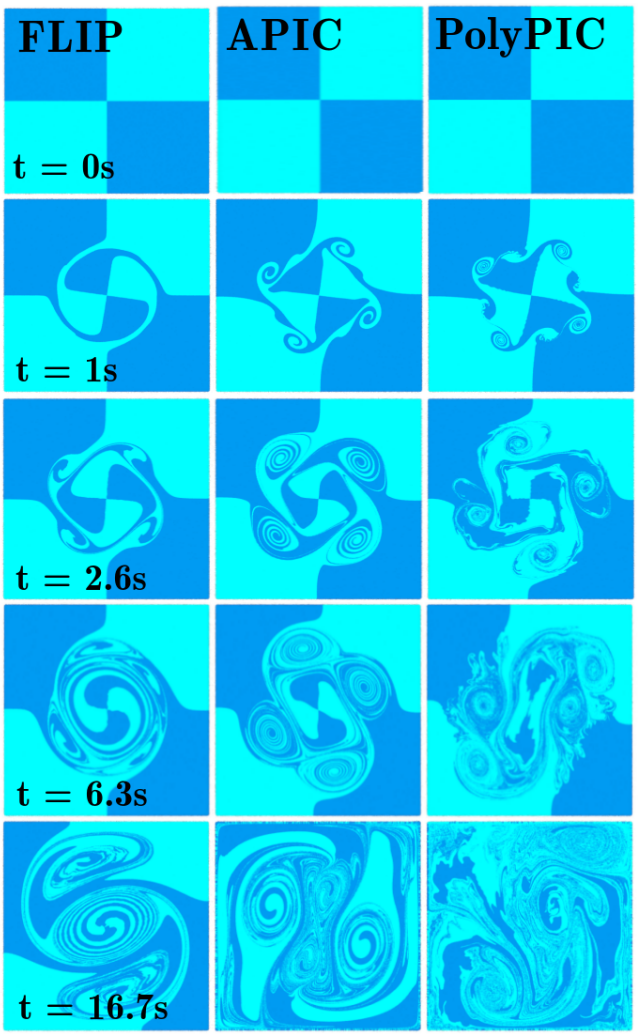


Fig. 3. **Vortex sheet.** We compare from left to right FLIP, APIC, and PolyPIC with 2D incompressible flow. The initial conditions are of a rotating circle surrounded by stationary fluid. This creates a vortex sheet which our method effectively resolves. The bottom row shows that despite the energetic nature of our method, our simulations are stable at long runtimes.

The Lagrangian state associated with particle p at time t^n consists of mass m_p, position $\mathbf{x}_p^n$, generalized velocity coefficients $\mathbf{c}_p^n$ and auxiliary quantities $\mathbf{A}_p^n$. Note that the mass does not change with time in accordance with conservation of mass. The auxiliary quantities in $\mathbf{A}_p^n$ are not relevant to our particle/grid transfers but we include them for completeness. E.g. in an MPM calculation the deformation gradient $\mathbf{F}_p^n$ is auxiliary to transfers and would be included in $\mathbf{A}_p^n$. We will generally consider the update of the auxiliary quantities to be outside the scope of the paper.

In order to update the Lagrangian state to obtain $\mathbf{x}_p^{n+1}$, $\mathbf{c}_p^{n+1}$ and $\mathbf{A}_p^{n+1}$, we first transfer mass and momentum from particle to grid (Section (§6.1)), then grid momentum is dynamically updated (Section (§6.2)) and finally, we transfer the generalized velocity

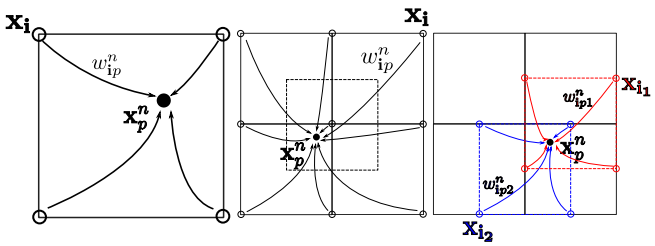
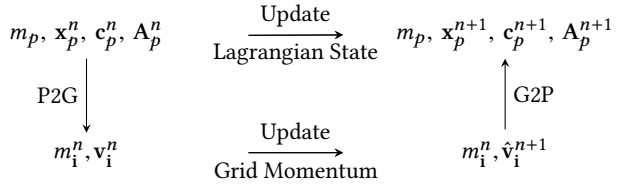


Fig. 4. **Grid interpolation**. We visualize the weights w^n_{ip} for multilinear ($N_B = 1$), collocated (left), multiquadratic ($N_B = 2$), collocated (center) and weights $w^n_{i_\alpha p}$ for linear ($N_B = 1$), MAC grids (right). We emphasize that the particle interpolates from $(N_B + 1)^d$ grid nodes.

information from grid to particle (Section (§6.3)). We use the notation $m^n_{\mathbf{i}}$ and $\mathbf{v}^n_{\mathbf{i}}$ to denote the mass and velocity transferred to the grid node $\mathbf{x_i}$ from the particles before the grid momentum update. We further use the notation $\hat{\mathbf{v}}^{n+1}_{\mathbf{i}}$ to denote the grid node velocity that is updated in grid momentum update. We use this convention to distinguish it from $\mathbf{v}^{n+1}_{\mathbf{i}}$, the velocity that is transferred to the grid in the next time step. Lastly, we use $\mathbf{x}^{n+1}_{\mathbf{i}} = \mathbf{x_i} + \Delta t \hat{\mathbf{v}}^{n+1}_{\mathbf{i}}$ to denote the position of the grid nodes if they move with the grid node velocity $\hat{\mathbf{v}}^{n+1}_{\mathbf{i}}$. This process is illustrated in following commutative diagram.



Grid-based interpolating functions $N(\mathbf{x} - \mathbf{x_i})$ provide the mechanism for the transfer of particle and grid quantities. As in many other recent approaches [Jiang et al. 2015; Steffen et al. 2008; Stomakhin et al. 2013], the grid interpolating functions are constructed from dyadic products of one-dimensional B-splines. We use the notation $w^n_{ip} = N(\mathbf{x_i} - \mathbf{x}^n_p)$ to denote the weight of interaction between node $\mathbf{x_i}$ and particle $\mathbf{x}^n_p$.

We note that a particle will interpolate from $(N_B + 1)^d$ grid nodes where N_B is the B-spline interpolating order (1 for linear, 2 for quadratic, etc) and $d = 2, 3$ is the spatial dimension. In other words, the particle with position $\mathbf{x}^n_p$ will only have non-zero weights w^n_{ip} for the $(N_B + 1)^d$ grid nodes most local to it. We will use the notation $\hat{\mathcal{V}}^{n+1}_p \in \mathbb{R}^{d(N_B+1)^d}$ to denote the vector of updated grid-node velocities $\hat{\mathbf{v}}^{n+1}_{\mathbf{i}^n_{kp}}$ corresponding to grid nodes $\mathbf{x}_{\mathbf{i}^n_{kp}}$ with non-zero weights $w^n_{\mathbf{i}^n_{kp}p}$

$$\hat{\mathcal{V}}^{n+1}_p = \begin{pmatrix} \hat{\mathbf{v}}^{n+1}_{\mathbf{i}^n_{1p}} \\ \hat{\mathbf{v}}^{n+1}_{\mathbf{i}^n_{2p}} \\ \vdots \\ \hat{\mathbf{v}}^{n+1}_{\mathbf{i}^n_{(N_B+1)^d p}} \end{pmatrix}.$$

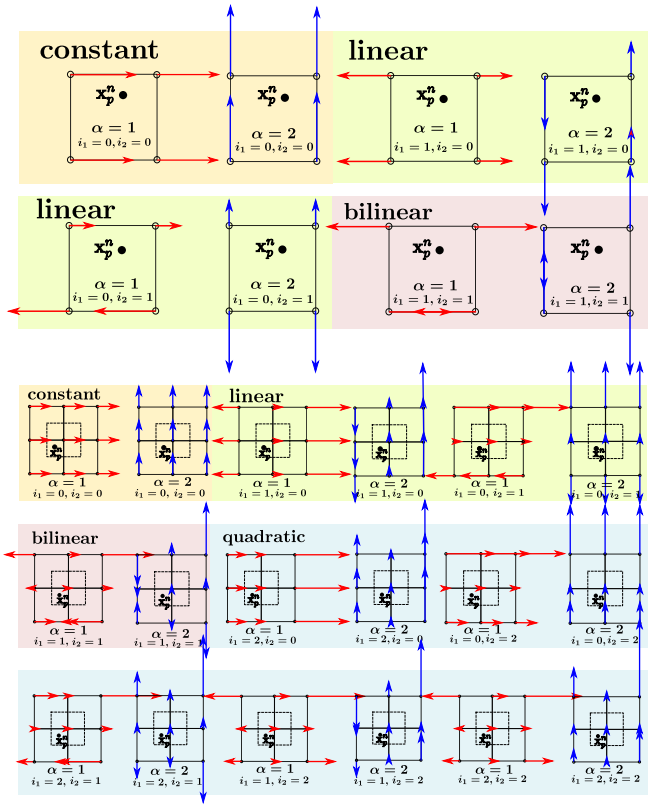


Fig. 5. **Velocity modes**. We visualize the component-wise velocity modes from Equation (2) in 2D. The top shows bilinear interpolation and the bottom shows biquadratic interpolation. Constant (peach), linear (green), bilinear (pink) and biquadratic (light blue) modes are depicted for x (red) and y (blue) components.

We use $\mathbf{i}^n_{kp}$ for $k = 1, 2, \ldots, (N_B + 1)^d$ as an index for nodes with non-zero weights $w^n_{\mathbf{i}^n_{kp}p}$. We illustrate this in Figure 4. When it is clear from context, we will use either $\mathbf{i}_k, w^n_{\mathbf{i}_k p}$ or even $\mathbf{i}_k, w^n_{ip}$ in lieu of the more descriptive $\mathbf{i}^n_{kp}, w^n_{\mathbf{i}^n_{kp}p}$ since the sub and super indices can become excessive in some expressions.

4 VELOCITY MODES

Our approach closely resembles that of Jiang et al. [2015; 2017b]. Our most fundamental difference is that instead of augmenting particles with affine velocities, we augment them with more general functions. In the APIC approaches advocated by Jiang et al. [2015; 2017b], the velocity local to the particle p at time t^n is approximated as

$$\mathbf{v}^n_p(\mathbf{x}) = \mathbf{v}^n_p + \mathbf{C}^n_p(\mathbf{x} - \mathbf{x}^n_p)$$

where $\mathbf{v}^n_p$ is the velocity of the particle and the matrix $\mathbf{C}^n_p \in \mathbb{R}^{d\times d}$ satisfies $\mathbf{C}^n_p = \mathbf{0}$ for PIC, $\mathbf{C}^n_p = -\left(\mathbf{C}^n_p\right)^T$ for RPIC (locally rigid PIC) and $\mathbf{C}^n_p$ is arbitrary for APIC.

In this paper, we improve the approach by considering the particle-wise local velocity to be of the form

$$\mathbf{v}_p^n(\mathbf{x}) = \sum_{r=1}^{N_r} \sum_{\alpha=1}^{d} s_r(\xi_p^n(\mathbf{x}) - \mathbf{x}_p^{n-1}) \mathbf{e}_\alpha c_{pr\alpha}^n \tag{1}$$

where the functions $s_r \mathbf{e}_\alpha : \mathbb{R}^d \to \mathbb{R}^d$ are generalized velocity modes, $\mathbf{e}_\alpha \in \mathbb{R}^d$ is the α^{th} standard basis vector and the $c_{pr\alpha}^n$ are the coefficients of the modes which are stored in the vector $\mathbf{c}_p^n \in \mathbb{R}^{dN_r}$. We build our generalized velocity modes component-by-component in terms of the scalar functions $s_r : \mathbb{R}^d \to \mathbb{R}$. N_r indicates the total number of scalar modes that we use. We illustrate these modes in Figure 5. The function ξ_p^n approximates the mapping from the time t^n configuration to the time t^{n-1} configuration local to the particle and represents the advection of the material (see Section (§5)). We note that in PIC and APIC it is simply given by $\xi_p^n(\mathbf{x}) = \mathbf{x} - \Delta t \mathbf{v}_p^{n-1}$ and $\xi_p^n(\mathbf{x}) - \mathbf{x}_p^{n-1} = \mathbf{x} - \mathbf{x}_p^n$.

By approximating the velocity local to particle $\mathbf{x}_p^n$ in terms of more general functions, we allow for a wider range of local behavior than in the original APIC. Notably, we can write APIC in this way by choosing affine functions for s_r. Similarly, we can write PIC in this way by choosing constant functions for the s_r. In either case we note that the coefficients $\mathbf{c}_p^n \in \mathbb{R}^{dN_r}$ are equivalent to the $\mathbf{v}_p^n$ and $\mathbf{C}_p^n$ in the original APIC and PIC. Note that for APIC, $dN_r = d^2 + d$ (d translations and d^2 linear functions) and similarly for PIC, $dN_r = d$.

We primarily use polynomial modes of the form

$$s(\mathbf{z}) = \prod_{\beta=1}^{d} z_\beta^{i_\beta}. \tag{2}$$

Here z_β is the β^{th} component of $\mathbf{z} \in \mathbb{R}^d$, the $i_\beta \in \mathbb{Z}^+$ are non-negative integer powers. We note that this reduces to the original PIC when $i_\beta = 0$ for $1 \le \beta \le d$. Furthermore, when we choose all s_r with exactly one of the $i_\beta = 1$ and the rest equal to zero, we obtain the affine modes and the method reduces to APIC. In general, we will modify the polynomial modes in Equation (2) slightly to ensure a mass-orthogonality condition that is essential for efficiency in the grid to particle transfer (see Section (§6.3)).

The particle-wise local velocity in Equation (1) is used in the particle-to-grid and grid-to-particle transfers. As in [Jiang et al. 2015, 2017b], it is used to define a particle's contribution to the grid node linear momentum in the particle-to-grid transfer (Section (§6.1)). In the grid-to-particle transfer (Section (§6.3)), the coefficients $\mathbf{c}_p^{n+1}$ are chosen so that

$$\hat{\mathbf{v}}_p^{n+1}(\mathbf{y}) = \sum_{r=1}^{N_r} \sum_{\alpha=1}^{d} s_r(\mathbf{y} - \mathbf{x}_p^n) \mathbf{e}_\alpha c_{pr\alpha}^{n+1} \approx \sum_{\mathbf{i}} \hat{\mathbf{v}}_{\mathbf{i}}^{n+1} N(\mathbf{y} - \mathbf{x}_{\mathbf{i}}) \tag{3}$$

for $\mathbf{y}$ near $\mathbf{x}_p^n$. However, this approximation is done with points $\mathbf{y}$ in the time t^n rather than t^{n+1} configuration of the material. This is a local approximation of the updated Lagrangian velocity. The updated Lagrangian velocity is related to the Eulerian velocity by the mapping $(\xi_p^{n+1})^{-1}$, which approximates the advection of the material local to the particle to the time t^{n+1} configuration. We discuss the significance of this mapping and the notion of updated Lagrangian velocity in the next section.

5 UPDATED LAGRANGIAN

Eulerian and Lagrangian methods can be characterized in terms of the flow map of the material $\boldsymbol{\phi}(\cdot, t) : \Omega^0 \to \Omega^t$ [Gonzalez and Stuart 2008]. Here $\Omega^0 \subset \mathbb{R}^d$ is the initial configuration of the material. Each point $\mathbf{X} \in \Omega^0$ is the initial position of a particle of material in the continuum and $\boldsymbol{\phi}(\mathbf{X}, t)$ is its location at time t. Ω^t is the time t configuration of the material consisting of the points $\mathbf{x} = \boldsymbol{\phi}(\mathbf{X}, t)$ for some $\mathbf{X} \in \Omega^0$. It is this mapping that defines the Lagrangian velocity and acceleration of each particle via $\mathbf{V}(\mathbf{X}, t) = \frac{\partial \boldsymbol{\phi}}{\partial t}(\mathbf{X}, t)$ and $\mathbf{A}(\mathbf{X}, t) = \frac{\partial \mathbf{V}}{\partial t}(\mathbf{X}, t)$. The Eulerian counterparts can be defined in terms of the inverse of the flow map $\boldsymbol{\phi}^{-1}(\cdot, t) : \Omega^t \to \Omega^0$ via $\mathbf{v}(\mathbf{x}, t) = \mathbf{V}(\boldsymbol{\phi}^{-1}(\mathbf{x}, t), t)$ and $\mathbf{a}(\mathbf{x}, t) = \mathbf{A}(\boldsymbol{\phi}^{-1}(\mathbf{x}, t), t)$ for all $\mathbf{x} \in \Omega^t$. Here $\mathbf{X} = \boldsymbol{\phi}^{-1}(\boldsymbol{\phi}(\mathbf{X}, t), t)$ for all particles $\mathbf{X} \in \Omega^0$ and $\mathbf{x} = \boldsymbol{\phi}(\boldsymbol{\phi}^{-1}(\mathbf{x}, t), t)$ for all points $\mathbf{x} \in \Omega^t$. Also, the Eulerian velocity and acceleration are related through the total derivative

$$\mathbf{a}(\mathbf{x}, t) = \frac{D\mathbf{v}}{Dt}(\mathbf{x}, t) = \frac{\partial \mathbf{v}}{\partial t}(\mathbf{x}, t) + \frac{\partial \mathbf{v}}{\partial \mathbf{x}}(\mathbf{x}, t)\mathbf{v}(\mathbf{x}, t). \tag{4}$$

We note that Levin et al. have developed a number of methods that make use of the inverse flow map [Fan et al. 2013; Levin et al. 2011; Teng et al. 2016].

In Lagrangian approaches, the initial configuration of the material Ω^0 serves as the domain of field functions like velocity and stress. This is sometimes referred to as a total Lagrangian approach. Updated Lagrangian approaches use a similar idea, but rather than using a single reference configuration Ω^0, the time t^n configuration Ω^{t^n} is used as the reference. With these approaches, it is convenient to define $\hat{\mathbf{v}}(\mathbf{y}, t) = \mathbf{V}(\boldsymbol{\phi}^{-1}(\mathbf{y}, t^n), t)$ and $\hat{\mathbf{a}}(\mathbf{y}, t) = \mathbf{A}(\boldsymbol{\phi}^{-1}(\mathbf{y}, t^n), t)$ for all $\mathbf{y} \in \Omega^{t^n}$. These are the time t velocity and acceleration defined over Ω^{t^n}. They are analogous to the Lagrangian acceleration and velocity, except the points $\mathbf{y} \in \Omega^{t^n}$ serve as the reference for each particle in the continuum, rather than the initial points $\mathbf{X} \in \Omega^0$. This is convenient because unlike the Eulerian velocity and acceleration in Equation (4) that relate through the total derivative, $\hat{\mathbf{v}}$ and $\hat{\mathbf{a}}$ relate through standard temporal differentiation

$$\hat{\mathbf{a}}(\mathbf{y}, t) = \frac{\partial \hat{\mathbf{v}}}{\partial t}(\mathbf{y}, t). \tag{5}$$

Thus, the updated Lagrangian velocity and acceleration have the same essential relation as their total Lagrangian counterparts.

PIC can be viewed as an updated Lagrangian approach. At time t^n the particles have positions $\mathbf{x}_p^n = \boldsymbol{\phi}(\mathbf{X}_p, t^n)$ and represent samples of Ω^{t^n}. When we transfer state to the grid (see Section (§6.2)), we obtain approximations to the velocity $\mathbf{v}_{\mathbf{i}}^n$ and mass $m_{\mathbf{i}}^n$ at grid nodes $\mathbf{x}_{\mathbf{i}} \in \Omega^{t^n}$. This provides an alternative approximation to the time t^n configuration that has the advantage of being defined over particles with structured (grid-aligned) locations, as opposed to the unstructured $\mathbf{x}_p^n$. The structured nature of their locations has many advantages, e.g. it is easy to interpolate data via regular grid interpolating functions. Indeed, the Eulerian velocity at time t^n can be approximated via interpolation from $\mathbf{v}(\mathbf{x}, t^n) \approx \sum_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^n N(\mathbf{x} - \mathbf{x}_{\mathbf{i}})$. In the grid momentum update step, we assume that Ω^{t^n} is the updated reference configuration and approximate the updated Lagrangian

acceleration in an essentially Lagrangian manner via

$$\hat{\mathbf{a}}(\mathbf{y}, t^{n+1}) \approx \sum_{\mathbf{i}} \frac{\hat{\mathbf{v}}_{\mathbf{i}}^{n+1} - \mathbf{v}_{\mathbf{i}}^{n}}{\Delta t} N(\mathbf{y} - \mathbf{x}_{\mathbf{i}}),\ \mathbf{y} \in \Omega^{t^n}.$$

Here, the new grid node velocities $\hat{\mathbf{v}}_{\mathbf{i}}^{n+1}$ approximate samples of $\hat{\mathbf{v}}(\mathbf{y}, t^{n+1})$ at grid nodes $\mathbf{y} = \mathbf{x}_{\mathbf{i}} \in \Omega^{t^n}$. Their interpolant approximates the updated Lagrangian velocity $\hat{\mathbf{v}}(\mathbf{y}, t^{n+1}) = \sum_{\mathbf{i}} \hat{\mathbf{v}}_{\mathbf{i}}^{n+1} N(\mathbf{y} - \mathbf{x}_{\mathbf{i}})$, i.e. the time t^{n+1} velocity but defined over $\mathbf{y} \in \Omega^{t^n}$.

By definition, the time t^{n+1} Eulerian velocity is related to the updated Lagrangian velocity through

$$\mathbf{v}(\mathbf{x}, t^{n+1}) = \hat{\mathbf{v}}(\boldsymbol{\xi}^{n+1}(\mathbf{x}), t^{n+1}) \tag{6}$$

for $\mathbf{x} \in \Omega^{t^{n+1}}$. Here we use $\boldsymbol{\xi}^{n+1}(\mathbf{x}) = \boldsymbol{\phi}(\boldsymbol{\phi}^{-1}(\mathbf{x}, t^{n+1}), t^n)$ to denote the mapping of material from the time t^{n+1} configuration to the time t^n configuration. Intuitively, $\boldsymbol{\phi}^{-1}$ maps the point $\mathbf{x} \in \Omega^{t^{n+1}}$ to its reference location $\mathbf{X} = \boldsymbol{\phi}^{-1}(\mathbf{x}, t^{n+1}) \in \Omega^0$ and $\boldsymbol{\phi}$ maps the reference location to the its position $\boldsymbol{\xi}^{n+1}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{X}, t^n) \in \Omega^{t^n}$. Thus $\mathbf{X} = \boldsymbol{\phi}^{-1}(\mathbf{x}, t^{n+1}) \in \Omega^0$ is the location in the reference configuration Ω^0 of the particle that occupies $\mathbf{x} \in \Omega^{t^{n+1}}$ and $\boldsymbol{\xi}^{n+1}(\mathbf{x}) = \boldsymbol{\phi}(\boldsymbol{\phi}^{-1}(\mathbf{x}, t^{n+1}), t^n)$ is the location in the time t^n configuration Ω^{t^n} of the particle that occupies $\mathbf{x} \in \Omega^{t^{n+1}}$. In other words, the mapping reverses the motion of material over the time step. Since the updated Lagrangian velocity is the time t^{n+1} velocity, defined over the time t^n configuration, the composition with this mapping in Equation (6) can be viewed as the key to defining the Eulerian velocity in a PIC calculation.

5.1 Particle-Wise Velocity Modes

The $\hat{\mathbf{v}}_p^{n+1}(\mathbf{y})$ from Equation (3) locally approximate the updated Lagrangian velocity $\hat{\mathbf{v}}(\mathbf{y}, t^{n+1})$ for $\mathbf{y} \in \Omega^{t^n}$ near $\mathbf{x}_p^n$. We use them to obtain the similar, but more useful approximation to the Eulerian velocity $\mathbf{v}_p^{n+1}(\mathbf{x})$ to $\mathbf{v}(\mathbf{x}, t^{n+1})$. The $\mathbf{v}_p^{n+1}(\mathbf{x})$ are required for the transfers from particle to grid at the beginning of time step t^{n+1} (see Section (§6.1)). We obtain them by composition with the local approximation $\boldsymbol{\xi}_p^{n+1}(\mathbf{x})$ to $\boldsymbol{\xi}^{n+1}(\mathbf{x})$ for $\mathbf{x} \in \Omega^{t^{n+1}}$ near $\mathbf{x}_p^{n+1}$

$$\mathbf{v}_p^{n+1}(\mathbf{x}) = \hat{\mathbf{v}}_p^{n+1}(\boldsymbol{\xi}_p^{n+1}(\mathbf{x})). \tag{7}$$

In our approach, as well as in PIC and APIC, particles move with the interpolated updated Lagrangian velocity

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \sum_{\mathbf{i}} \hat{\mathbf{v}}_{\mathbf{i}}^{n+1} N(\mathbf{x}_p^n - \mathbf{x}_{\mathbf{i}}).$$

This motion of the particles defines the material mapping from configuration Ω^{t^n} to configuration $\Omega^{t^{n+1}}$. Since $\boldsymbol{\xi}^{n+1}(\mathbf{x})$ is the inverse of this mapping, we know its value at each particle $\boldsymbol{\xi}^{n+1}(\mathbf{x}_p^{n+1}) = \mathbf{x}_p^n$. We can use this to approximate the mapping local to each particle.

5.1.1 Piecewise constant material motion. If we assume that $\boldsymbol{\xi}^{n+1}$ is approximately a simple translation near $\mathbf{x}_p^{n+1}$ and that $\boldsymbol{\xi}^{n+1}(\mathbf{x}_p^{n+1}) = \mathbf{x}_p^n$, then we obtain the local approximation

$$\boldsymbol{\xi}_p^{n+1}(\mathbf{x}) = \mathbf{x}_p^n + \left(\mathbf{x} - \mathbf{x}_p^{n+1}\right). \tag{8}$$

The PIC and APIC transfers can be derived from the local velocities in Equation (1) combined with the advection approximation in Equation (8). With PIC, the local updated Lagrangian velocity is constant $\hat{\mathbf{v}}_p^{n+1}(\mathbf{y}) = \mathbf{v}_p^{n+1}$ and thus for any local approximation $\boldsymbol{\xi}_p^{n+1}(\mathbf{x})$,

$$\mathbf{v}_{\mathrm{PIC},p}^{n+1}(\mathbf{x}) = \mathbf{v}_p^{n+1}.$$

With APIC, the local updated Lagrangian velocity is affine $\hat{\mathbf{v}}_p^{n+1}(\mathbf{y}) = \mathbf{v}_p^{n+1} + \mathbf{C}_p^{n+1}(\mathbf{y} - \mathbf{x}_p^n)$. If we combine this affine velocity via the composition in Equation (8) with the constant local approximation in Equation (7), we obtain

$$\mathbf{v}_{\mathrm{APIC},p}^{n+1}(\mathbf{x}) = \hat{\mathbf{v}}_p^{n+1}(\mathbf{x}_p^n + \left(\mathbf{x} - \mathbf{x}_p^{n+1}\right)) = \mathbf{v}_p^{n+1} + \mathbf{C}_p^{n+1}(\mathbf{x} - \mathbf{x}_p^{n+1}).$$

5.1.2 Piecewise affine material motion. It is evident that more accurate local approximations to $\boldsymbol{\xi}^{n+1}(\mathbf{x})$ are readily available. If we assume that the updated Lagrangian velocity is well approximated by $\hat{\mathbf{v}}_p^{n+1}(\mathbf{y})$ for particles $\mathbf{y} \in \Omega^{t^n}$ near $\mathbf{x}_p^n$, then particle trajectories will evolve locally as approximately $\mathbf{y}^{n+1} = \mathbf{y} + \Delta t \hat{\mathbf{v}}_p^{n+1}(\mathbf{y})$. This approximates the motion of the material from Ω^{t^n} to $\Omega^{t^{n+1}}$ for particles near $\mathbf{x}_p^n$. The inverse of the mapping local to the particle is then approximately given by $\boldsymbol{\xi}_p^{n+1}(\mathbf{x}) = \hat{\mathbf{x}}$ where $\hat{\mathbf{x}}$ is given by the solution to the implicit equation

$$\mathbf{x} = \hat{\mathbf{x}} + \Delta t \hat{\mathbf{v}}_p^{n+1}(\hat{\mathbf{x}}). \tag{9}$$

Intuitively, for particle $\mathbf{x} \in \Omega^{t^{n+1}}$, $\hat{\mathbf{x}} \in \Omega^{t^n}$ is its location at time t^n. For general functions $\hat{\mathbf{v}}_p^{n+1}$, Equation (9) can be solved using Newton's method; however if we approximate the updated Lagrangian velocity by its affine components $\hat{\mathbf{v}}_p^{n+1}(\hat{\mathbf{x}}) \approx \sum_{r=1}^{N_r} \sum_{\alpha=1}^{d} s_r(\hat{\mathbf{x}} - \mathbf{x}_p^n)\mathbf{e}_\alpha c_{pr\alpha}^{n+1}$ with $c_{pr\alpha}^{n+1}$ only non-zero for affine modes, then the system for $\hat{\mathbf{x}}$ is linear and we obtain

$$\boldsymbol{\xi}_p^{n+1}(\mathbf{x}) = \mathbf{x}_p^n + \left(\mathbf{I} + \Delta t \mathbf{C}_p^{n+1}\right)^{-1} \left(\mathbf{x} - \mathbf{x}_p^{n+1}\right) \tag{10}$$

where $\mathbf{C}_p^{n+1}$ is the linear part of the polynomial modes. When $\mathbf{C}_p^{n+1} = \mathbf{0}$, we obtain the constant approximation in Equation (8) and thus this can be seen as a higher-order approximation. We visualize the approximations to $\boldsymbol{\xi}_p^{n+1}$ in Equations (8) and (10) in Figure 6.

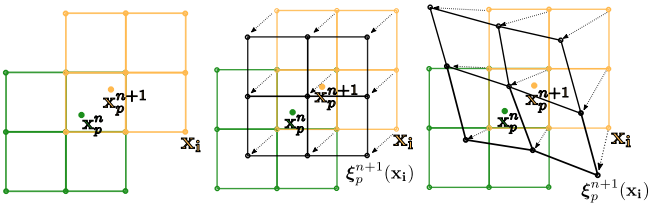


Fig. 6. **Updated Lagrangian.** Here we visualize the options for the mapping $\boldsymbol{\xi}_p^{n+1}$. As a particle moves, from $\mathbf{x}_p^n$ to $\mathbf{x}_p^{n+1}$, it selects different grid nodes $\mathbf{x}_{\mathbf{i}}$ to biquadratically interpolate from (green at t^n and yellow at t^{n+1}). The middle shows the $\boldsymbol{\xi}_p^{n+1}(\mathbf{x}_{\mathbf{i}})$ approximation from Equation (8) and the right from Equation (10).

6 METHOD

Here we detail all of the steps necessary for advancing the Lagrangian state from time t^n to t^{n+1} in a PIC calculation. We cover the necessary details for a MPM approach with elastoplastic materials as well as for incompressible Euler fluids with pressure projection on a MAC grid. This process consists of (1) the transfer from particle

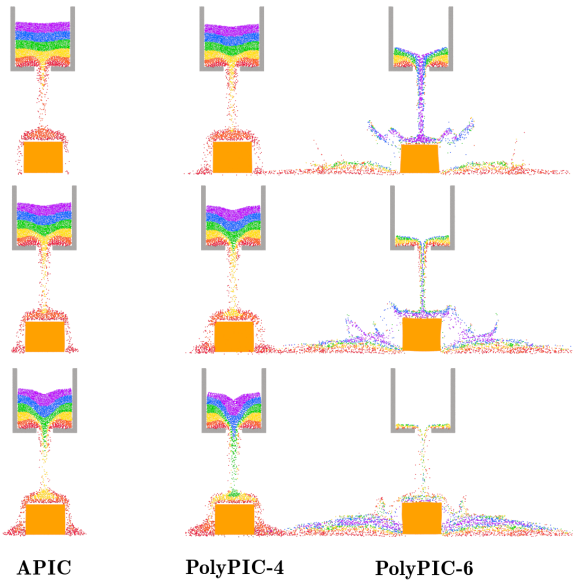


Fig. 7. **MPM elastoplasticity**. Rainbow colored sand is poured onto an elastic Jell-O square. We compare APIC (left) vs. PolyPIC with (from left to right) $N_r = 4$ and $N_r = 6$. Notice that increasing degrees of PolyPIC allow for more energetic sand flowing and Jell-O bouncing.

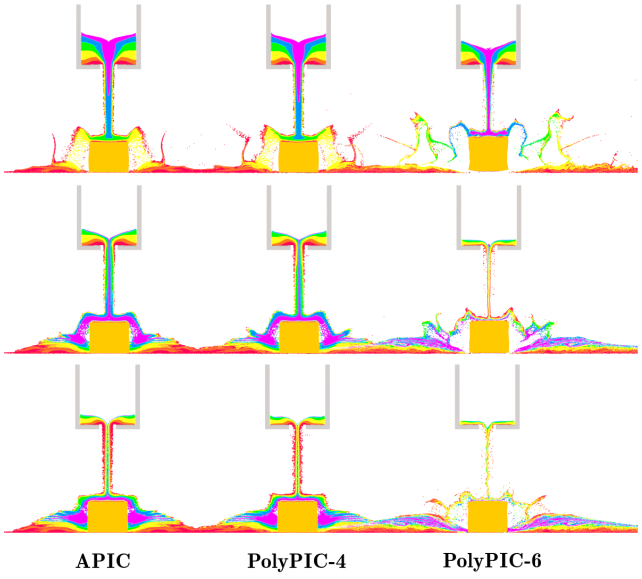


Fig. 8. **MPM elastoplasticity refinement**. We verify that the behavior exhibited by PolyPIC with $N_r = 6$ at lower resolution in Figure 7 is exhibited by PolyPIC with $N_r = 4$ and APIC under refinement.

to grid of mass and linear momentum (Section (§6.1)), (2) the grid based momentum update (Section (§6.2)) and (3) the transfer from grid to particle of generalized velocity coefficients (Section (§6.3)). The particle-wise local approximations discussed in Sections (§4) and (§5) are the keys to the particle/grid transfers. We provide the details of the grid momentum update for completeness but we note that it is not novel as our approach is not relevant to this step.

6.1 Transfer from Particle to Grid

The velocity local to the particle $\mathbf{v}_p^n : \mathbb{R}^d \to \mathbb{R}^d$ from Equation (1) is used to design the momentum transfer to the grid. We use the notation $(m\mathbf{v})_{\mathbf{i}p}^n = m_p w_{\mathbf{i}p}^n \mathbf{v}_p^n(\mathbf{x_i})$ to denote the particle's contribution to the momentum local to the node $\mathbf{x_i}$ and $(m\mathbf{v})_{\mathbf{i}}^n = \sum_p (m\mathbf{v})_{\mathbf{i}p}^n$ is the total momentum of grid node from the contribution of all particles. Similarly, the contribution of the particle's mass to the grid node $\mathbf{x_i}$ is $m_{\mathbf{i}p}^n = w_{\mathbf{i}p}^n m_p$ and the total grid node mass is the sum of the contributions from all particles $m_{\mathbf{i}}^n = \sum_p m_{\mathbf{i}p}^n$. Using this we can define the grid node velocity $\mathbf{v}_{\mathbf{i}}^n$ by dividing momentum by mass. In summary, this transfer consists of

$$\begin{aligned}(m\mathbf{v})_{\mathbf{i}p}^n &= m_{\mathbf{i}p}^n \sum_{r=1}^{N_r} \sum_{\alpha=1}^{d} s_r(\xi_p^n(\mathbf{x_i}) - \mathbf{x}_p^{n-1})\mathbf{e}_\alpha c_{pr\alpha}^n \\ (m\mathbf{v})_{\mathbf{i}}^n &= \sum_p (m\mathbf{v})_{\mathbf{i}p}^n,\ \mathbf{v}_{\mathbf{i}}^n = \frac{(m\mathbf{v})_{\mathbf{i}}^n}{m_{\mathbf{i}}^n}.\end{aligned} \tag{11}$$

Either local approximations $\xi_p^n(\mathbf{x_i})$ from Equations (8) or (10) can be used. We note that this is essentially the same transfer as in the original APIC approaches [Jiang et al. 2015, 2017b], with the only modification being the more general notions of the local velocity and the improved approximation to $\xi_p^n(\mathbf{x_i})$.

6.2 Update Grid Momentum

The grid momentum update is outside the scope of this paper. However, we include a generic description for representative cases that we used to generate our examples: incompressible Euler fluids and elastoplastic solids with MPM. In the case of the incompressible Euler, we used a MAC grid discretization of the pressure projection to update the fluid velocity. In the case of elastoplastic solids and MPM the update is from the elastic force (see [Fu et al. 2014] for more details).

$$\begin{aligned}\hat{\mathbf{v}}_{\mathbf{i}}^{n+1} &= \mathbf{v}_{\mathbf{i}}^n + \frac{\Delta t}{\rho}\nabla p, && \text{(Euler/MAC)} \\ \hat{\mathbf{v}}_{\mathbf{i}}^{n+1} &= \mathbf{v}_{\mathbf{i}}^n + \frac{\Delta t}{m_{\mathbf{i}}^n}(\mathbf{f} + \mathbf{g}), && \text{(elastoplastic/MPM)}\end{aligned}$$

where $\mathbf{f}$ is the elastic force and $\mathbf{g}$ is the gravitational acceleration.

6.3 Transfer from Grid to Particle

The transfer from grid to particle is achieved by choosing the generalized velocity coefficients $\mathbf{c}_p^{n+1} \in \mathbb{R}^{dN_r}$ so that the approximation in Equation (3) is optimal in the appropriate sense. Here we show that we can solve a linear system for the coefficients $\mathbf{c}_p^{n+1} \in \mathbb{R}^{dN_r}$, and that by design our approach (1) is equivalent to PIC and APIC if only constant or affine modes are used, (2) conserves linear and angular momentum (see [Fu et al. 2014]) and (3) has a *diagonal* system matrix in the equation for the $\mathbf{c}_p^{n+1} \in \mathbb{R}^{dN_r}$.

s_r \ s_t	1	x	y	xy	x^2	y^2	x^2y	xy^2	x^2y^2
1	X				X	X			X
x		X			X			X	X
y			X			X	X		X
xy				X			X	X	X
x^2	X	X			X	X		X	X
y^2	X		X		X	X	X		X
x^2y			X	X		X	X	X	X
xy^2		X		X	X		X	X	X
x^2y^2	X	X	X	X	X	X	X	X	X

Table 1. **Sparsity pattern: unmodified.** We illustrate the sparsity pattern of the matrix $(\mathbf{S}_p^n)^T \mathbf{m}_p^n \mathbf{S}_p^n$ for dimension $d = 2$ with scalar modes $s = x^{i_1} y^{i_2}$. X indicates a non-zero entry in the matrix. Note that the multilinear modes (indicated in red) are mass-orthogonal to one another, but that the multiquadratic modes couple extensively.

We choose the coefficients $\mathbf{c}_p^{n+1}$ to minimize the mass-weighted distance $d_p^n(\mathbf{c}_p^{n+1})$ between local velocities at the grid nodes and the updated grid-node velocities

$$d_p^n(\mathbf{c}_p^{n+1}) = \sum_{\mathbf{i}} m_{\mathbf{i}p}^n \left| \hat{\mathbf{v}}_{\mathbf{i}}^{n+1} - \hat{\mathbf{v}}_p^{n+1}(\mathbf{x}_{\mathbf{i}}) \right|^2 = \sum_{\mathbf{i}} m_{\mathbf{i}p}^n \left| \hat{\mathbf{v}}_{\mathbf{i}}^{n+1} - \sum_{r=1}^{N_r} \sum_{\alpha=1}^{d} s_r(\mathbf{x}_{\mathbf{i}} - \mathbf{x}_p^n)\mathbf{e}_\alpha c_{pr\alpha}^{n+1} \right|^2$$

where $m_{\mathbf{i}p}^n = m_p w_{\mathbf{i}p}^n$ is the mass that the particle $\mathbf{x}_p^n$ transfers to the grid node $\mathbf{i}$. The minimizer of this mass weighted distance can be expressed more concisely in terms of the grid node locations that received non-zero mass from the particle $\mathbf{x}_p^n$. Recall that the particle will have non-zero weights $w_{\mathbf{i}p}^n$ for precisely the $(N_B+1)^d$ grid nodes in closest proximity to the particle and that $\hat{\mathcal{V}}_p^{n+1} \in \mathbb{R}^{d(N_B+1)^d}$ is the vector of velocities of the grid nodes with non-zero weights. Similarly, we use the notation $\mathbf{Q}_p^n = \left[Q_{p11}^n, Q_{p12}^n, \ldots, Q_{pN_r d}^n \right] \in \mathbb{R}^{d(N_B+1)^d \times dN_r}$ where the columns $Q_{pr\alpha}^n$ of $\mathbf{Q}_p^n$ are analogous to $\hat{\mathcal{V}}_p^{n+1}$ and have entries equal to the particle-wise local modes $s_r(\mathbf{x}_{\mathbf{i}} - \mathbf{x}_p^n)\mathbf{e}_\alpha$ at the grid nodes with non-zero weights

$$Q_{pr\alpha}^n = \begin{pmatrix} s_r(\mathbf{x}_{\mathbf{i}_1} - \mathbf{x}_p^n)\mathbf{e}_\alpha \\ s_r(\mathbf{x}_{\mathbf{i}_2} - \mathbf{x}_p^n)\mathbf{e}_\alpha \\ \vdots \\ s_r(\mathbf{x}_{\mathbf{i}_{(N_B+1)^d}} - \mathbf{x}_p^n)\mathbf{e}_\alpha \end{pmatrix}.$$

The optimal coefficients $\mathbf{c}_p^{n+1}$ can be expressed in terms of these vectors as

$$\mathbf{c}_p^{n+1} = \underset{\mathbf{c} \in \mathbb{R}^{dN_r}}{\operatorname{argmin}} \; d_p^n(\mathbf{c}) = \left((\mathbf{Q}_p^n)^T \mathbf{M}_p^n \mathbf{Q}_p^n \right)^{-1} (\mathbf{Q}_p^n)^T \mathbf{M}_p^n \hat{\mathcal{V}}_p^{n+1} \tag{12}$$

s_r \ s_t	1	x	y	xy	$g_1(x)$	$g_2(y)$	$g_1(x)y$	$xg_2(y)$	$g_1(x)g_2(y)$
1	a								
x		b							
y			b						
xy				c					
$g_1(x)$					$d(x)$				
$g_2(y)$						$d(y)$			
$g_1(x)y$							$e(x)$		
$xg_2(y)$								$e(y)$	
$g_1(x)g_2(y)$									$f(x,y)$

Table 2. **Sparsity pattern: modified.** We illustrate the sparsity pattern of the matrix $(\mathbf{S}_p^n)^T \mathbf{m}_p^n \mathbf{S}_p^n$ for dimension $d = 2$ with the modified quadratic modes given by Equation (14). $a = 1$, $b = \frac{\Delta x^2}{4}$, $c = \frac{\Delta x^2}{16}$, $d(z) = \frac{(\Delta x^2 - 4z^2)^2 (3\Delta x^2 - 4z^2)}{16 \Delta x^2}$, $e(z) = \frac{(\Delta x^2 - 4z^2)^2 (3\Delta x^2 - 4z^2)}{64}$, $f(x,y) = \frac{(\Delta x^2 - 4x^2)^2 (3\Delta x^2 - 4x^2)(\Delta x^2 - 4y^2)^2 (3\Delta x^2 - 4y^2)}{256 \Delta x^4}$.

where the matrix $\mathbf{M}_p^n \in \mathbb{R}^{d(N_B+1)^d \times d(N_B+1)^d}$ is diagonal and consists of $(N_B + 1)$ diagonal blocks $m_{\mathbf{i}p}^n \mathbf{I}_d$. Here $\mathbf{I}_d \in \mathbb{R}^{d \times d}$ is the $d-$dimensional identity.

6.3.1 Dimension-by-dimension decoupling. Our approach is only efficient if the linear system for $\mathbf{c}_p^{n+1}$ in Equation (12) can be solved quickly. Fortunately, the matrix $(\mathbf{Q}_p^n)^T \mathbf{M}_p^n \mathbf{Q}_p^n \in \mathbb{R}^{dN_r \times dN_r}$ has remarkable properties for polynomial velocity modes of the type in Equation (2). First, because the matrix $\mathbf{M}_p^n$ is diagonal,

$$\mathbf{M}_p^n Q_{pt\beta}^n = \begin{pmatrix} m_{\mathbf{i}_1 p}^n s_t(\mathbf{x}_{\mathbf{i}_1} - \mathbf{x}_p^n)\mathbf{e}_\beta \\ m_{\mathbf{i}_2 p}^n s_t(\mathbf{x}_{\mathbf{i}_2} - \mathbf{x}_p^n)\mathbf{e}_\beta \\ \vdots \\ m_{\mathbf{i}_{(N_B+1)^d} p}^n s_t(\mathbf{x}_{\mathbf{i}_{(N_B+1)^d}} - \mathbf{x}_p^n)\mathbf{e}_\beta \end{pmatrix}.$$

Therefore, the entries in $\mathbf{M}_p^n Q_{pt\beta}^n$ are proportionate to the entries in $Q_{pr\alpha}^n$. Thus, since they have the same dimension-by-dimension sparsity as a consequence of the $\mathbf{e}_\beta$ and $\mathbf{e}_\alpha$ terms, the individual dimensions decouple when we take the dot products $Q_{pr\alpha}^n \cdot \left(\mathbf{M}_p^n Q_{pt\beta}^n \right)$. The exact expression for the entries in the matrix $(\mathbf{Q}_p^n)^T \mathbf{M}_p^n \mathbf{Q}_p^n$ are then

$$Q_{pr\alpha}^n \cdot \left(\mathbf{M}_p^n Q_{pt\beta}^n \right) = \sum_{\mathbf{i}} m_{\mathbf{i}}^n s_r(\mathbf{x}_{\mathbf{i}} - \mathbf{x}_p^n) s_t(\mathbf{x}_{\mathbf{i}} - \mathbf{x}_p^n) \mathbf{e}_\alpha \cdot \mathbf{e}_\beta.$$

Here we use r, α to index the rows and t, β to index the columns of the matrix $(\mathbf{Q}_p^n)^T \mathbf{M}_p^n \mathbf{Q}_p^n$. Furthermore, r, t index the mode type while α, β index the dimension. Therefore the matrix entries $Q_{pr\alpha}^n \cdot \left(\mathbf{M}_p^n Q_{pt\beta}^n \right) = 0$ when $\alpha \neq \beta$ since $\mathbf{e}_\alpha \cdot \mathbf{e}_\beta = 0$ when $\alpha \neq \beta$. Thus the coefficients $c_{pr\alpha}^n$ are decoupled in α and the matrix $(\mathbf{Q}_p^n)^T \mathbf{M}_p^n \mathbf{Q}_p^n$ is block diagonal with d identical diagonal blocks associated with the dimension-by-dimension velocity modes.

The d non-zero diagonal blocks of $(\mathbf{Q}_p^n)^T \mathbf{M}_p^n \mathbf{Q}_p^n \in \mathbb{R}^{dN_r \times dN_r}$ are each equal to $(\mathbf{S}_p^n)^T \mathbf{m}_p^n \mathbf{S}_p^n \in \mathbb{R}^{N_r \times N_r}$. Here $\mathbf{m}_p^n \in \mathbb{R}^{(N_B+1)^d \times (N_B+1)^d}$ is diagonal with entries equal to $m_{\mathbf{i}p}^n$. Furthermore, the matrix $\mathbf{S}_p^n$ consists of columns analogous to $\hat{\mathcal{V}}_p^{n+1}$ and $Q_{pr\alpha}^n$, but with entries equal to the *scalar* particle-wise local modes $s_r(\mathbf{x}_{\mathbf{i}} - \mathbf{x}_p^n)$ at the grid nodes with non-zero weights. $\mathbf{S}_p^n = \left[\mathcal{S}_{p1}^n, \ldots, \mathcal{S}_{pN_r}^n \right] \in \mathbb{R}^{(N_B+1)^d \times N_r}$ with

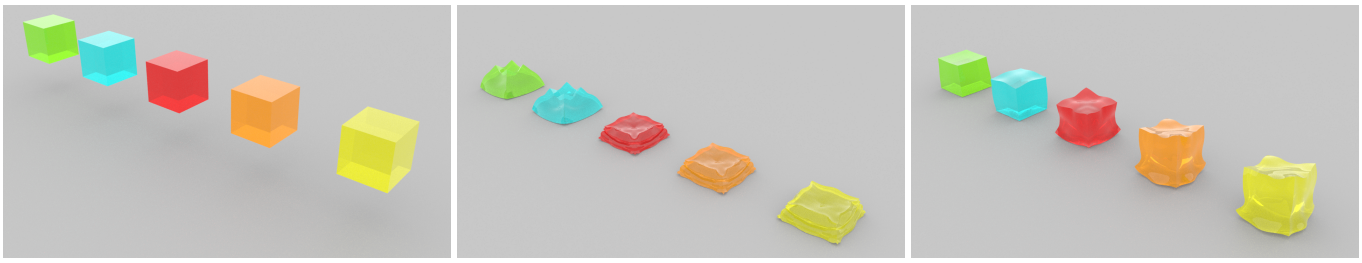


Fig. 9. **MPM hyperelasticity**. We compare from left to right APIC (green) and PolyPIC with $N_r = 8$ (blue), $N_r = 11$ (red), $N_r = 14$ (orange), $N_r = 18$ (yellow). PolyPIC better conserves total energy which results in less numerical damping of the deformable motion.

columns

$$\mathcal{S}^n_{pr} = \begin{pmatrix} s_r(\mathbf{x}_{\mathbf{i}_1} - \mathbf{x}^n_p) \\ s_r(\mathbf{x}_{\mathbf{i}_2} - \mathbf{x}^n_p) \\ \vdots \\ s_r(\mathbf{x}_{\mathbf{i}_{(N_B+1)^d}} - \mathbf{x}^n_p) \end{pmatrix} \in \mathbb{R}^{(N_B+1)^d}.$$

With this convention, $Q^n_{pr\alpha} \cdot \left(\mathbf{M}^n_p Q^n_{pt\beta}\right) = \mathcal{S}^n_{pr} \cdot \left(\mathbf{m}^n_p \mathcal{S}^n_{pt}\right) \mathbf{e}_\alpha \cdot \mathbf{e}_\beta$ and the dimension-by-dimension decoupled equations for the optimal coefficients $\mathbf{c}^{n+1}_p$ are

$$\begin{aligned} \sum_{t=1}^{N_r} \mathcal{S}^n_{pr} \cdot \left(\mathbf{m}^n_p \mathcal{S}^n_{pt}\right) c^{n+1}_{pt\alpha} &= Q^n_{pr\alpha} \cdot \left(\mathbf{M}^n_p \hat{\mathcal{V}}^{n+1}_p\right) \\ &= \sum_{\mathbf{i}} m^n_{\mathbf{i}p} s_r(\mathbf{x}_\mathbf{i} - \mathbf{x}^n_p) \hat{v}^{n+1}_{\mathbf{i}\alpha} \end{aligned} \tag{13}$$

for $1 \le r \le N_r$, where $\hat{v}^{n+1}_{\mathbf{i}\alpha}$ is the α^{th} component of $\hat{\mathbf{v}}^{n+1}_\mathbf{i}$.

6.3.2 Mass-orthogonal polynomial modes. The individual blocks $(\mathbf{S}^n_p)^T \mathbf{m}^n_p \mathbf{S}^n_p \in \mathbb{R}^{N_r \times N_r}$ have further favorable sparsity structure. If we assume that we number the modes with increasing degree (e.g. in 2D, constant modes first: $s_1 = 1$, followed by linear $s_2 = x, s_3 = y$, then multilinear: $s_4 = xy$, etc) and if we use modes s_r with $r \le N_r \le 2^d$, the matrix $(\mathbf{S}^n_p)^T \mathbf{m}^n_p \mathbf{S}^n_p$ is diagonal. This can be verified directly using Mathematica [2016] and we provide Mathematica code in [Fu et al. 2014]. Notably, this means that constant modes ($r \le 1$), linear modes ($1 < r \le d$) and multilinear modes ($d < r \le 2^d$) are mass-orthogonal and therefore the coefficients in Equation (13) can be obtained through the solution of a diagonal system.

In general for $2^d < r \le N_r \le (N_B+1)^d$, the matrix $(\mathbf{S}^n_p)^T \mathbf{m}^n_p \mathbf{S}^n_p$ is not diagonal. We illustrate this in Table 1 with $d = 2$ for brevity. However, we can obtain a diagonal system with a modified Gram-Schmidt approach that takes into account the inner product defined by $\mathbf{m}^n_p$. This amounts to simple modifications of the quadratic scalar modes $2^d < r \le N_r \le (N_B+1)^d$ in Equation (2). Remarkably, the Gram-Schmidt mass-orthogonalization does not modify any of the constant, linear or multilinear modes. Only the quadratic modes are modified and the change is very simple: each quadratic term z^2_β in Equation (2) is replaced with $g_\beta(z_\beta)$ given by

$$g_\beta(w) = w^2 - \frac{x^n_{p\beta}\left(\Delta x^2 - 4(x^n_{p\beta})^2\right)}{\Delta x^2} w - \frac{\Delta x^2}{4}. \tag{14}$$

E.g. the mode $s_5 = g_1(x)$ replaces x^2, $s_6 = g_2(y)$ replaces y^2, $s_7 = g_1(x)y$ replaces $x^2 y$, etc. This trivial modification yields a diagonal $(\mathbf{S}^n_p)^T \mathbf{m}^n_p \mathbf{S}^n_p$ whose entries we enumerate in Table 2. We give expressions for the individual entries in the solution $\mathbf{c}^{n+1}_p$ to Equation (13) with diagonal basis in the supplementary technical document [Fu et al. 2014].

We note that $(N_B+1)^d$ is a natural upper bound on the number of reduces modes N_r since the minimization in Equation (12) is over determined for $N_r > (N_B+1)^d$.

7 MAC GRID

For clarity of exposition, we only consider the case of collocated grids in Sections (§6.1) and (§6.3). For incompressible Euler we transfer to and from staggered velocity MAC grids [Harlow and Welch 1965]. Using $\mathbf{i}_\alpha$, $1 \le \alpha \le d$ to denote the face index for each of the staggered grids, MAC transfers are done component-wise (see Figure 4). Particle $\mathbf{x}^n_p$ transfers mass $m^n_{\mathbf{i}_\alpha p}$ to each α face grid from $m^n_{\mathbf{i}_\alpha p} = m^n_p w^n_{\mathbf{i}_\alpha p}$. The total mass on each grid face $m^n_{\mathbf{i}_\alpha}$ is equal to the sum of the contribution from each particle $m^n_{\mathbf{i}_\alpha} = \sum_p m^n_{\mathbf{i}_\alpha p}$. The weight of interaction $w^n_{\mathbf{i}_\alpha p} = N(\mathbf{x}_{\mathbf{i}_\alpha} - \mathbf{x}^n_p)$ is between the particle $\mathbf{x}^n_p$ and the MAC face $\mathbf{x}_{\mathbf{i}_\alpha}$. The component-wise particle-to-grid momentum transfer is

$$\begin{aligned} (mv)^n_{\mathbf{i}_\alpha p} &= m^n_{\mathbf{i}_\alpha p} \sum_r s_r(\xi^n_p(\mathbf{x}_{\mathbf{i}_\alpha}) - \mathbf{x}^{n-1}_p) c^n_{pr\alpha} \\ (mv)^n_{\mathbf{i}_\alpha} &= \sum_p (mv)^n_{\mathbf{i}_\alpha p}, \; v^n_{\mathbf{i}_\alpha} = \frac{(mv)^n_{\mathbf{i}_\alpha}}{m^n_{\mathbf{i}_\alpha}}. \end{aligned} \tag{15}$$

These transfers are very similar to those in Equation (11); however each face grid gets its own mass and respective component of the momentum. This is slightly more costly since mass must be transferred d times with the MAC grid instead of just one with the collocated grid.

The transfers from grid to particle are also trivially done component-wise since as discussed in Section (§6.3.1), the system in Equation (13) decouples component-wise. However, unlike in the collocated case, the mass matrix and scalar mode vectors will be different on each of the velocity face grids. We use the notation $\mathbf{m}^n_{p\alpha}$ and $\mathcal{S}^n_{pt\alpha}$ to denote this, where the appearance of α emphasizes that they vary with each face grid. With this convention we can write the system

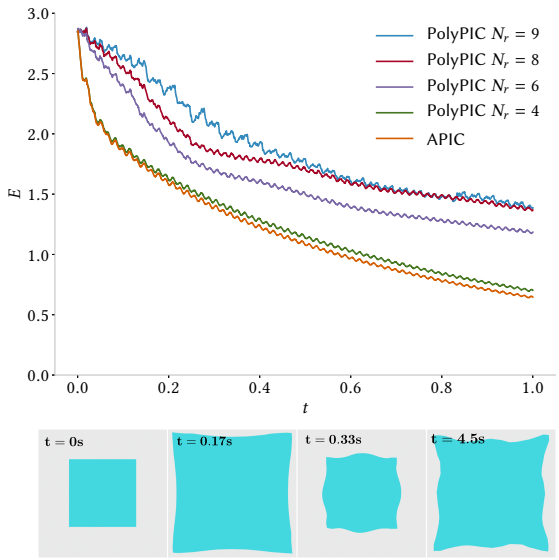


Fig. 10. **Energy conservation**. We plot of the total energy as a function of time for an elastic square with initial compressive dilation. The energy is calculated as the sum of the elastic potential energy on the particles and the kinetic energy on the grid.

for the reduced mode components $\mathbf{c}_p^{n+1} \in \mathbb{R}^{dN_r}$ as

$$\sum_{t=1}^{N_r} \mathcal{S}_{pr\alpha}^n \cdot \left(\mathbf{m}_{p\alpha}^n \mathcal{S}_{pt\alpha}^n\right) c_{pt\alpha}^{n+1} = \sum_{k=\mathbf{i}_\alpha} m_{\mathbf{i}_\alpha p}^n s_r(\mathbf{x}_{\mathbf{i}_\alpha} - \mathbf{x}_p^n) \hat{v}_{\mathbf{i}_\alpha}^{n+1} \qquad (16)$$

for $1 \le r \le N_r$. These systems are very similar to those in Equation (13). However, in the case of the MAC grid the matrices that appear in the left-hand side (whose entries $\mathcal{S}_{pr\alpha}^n \cdot \left(\mathbf{m}_{p\alpha}^n \mathcal{S}_{pt\alpha}^n\right)$ are indexed by $1 \le r, t \le N_r$) of Equation (16) are different on each of the α grids. In Equation 13 there is only one matrix on the left-hand side, independent of α.

8 RESULTS

We demonstrate our method on a number of examples with incompressible flow and MPM elastoplasticity. We compare PolyPIC with APIC and FLIP in a number of representative scenarios. All incompressible flow simulations were done using Manta Flow [Thuerey and Pfaff 2016]. In a few of our incompressible examples, we use passive advected particles as a post-process to aid in visualization. We note that these are simply advected in the flow for post-process visualization and do not use PolyPIC transfers. Also, all grid interpolation is multilinear for the incompressible flow examples. All grid interpolation is multiquadratic for the MPM elastoplasticity examples.

8.1 Incompressible Flow

In Figure 3 we simulate a vortex sheet by setting the velocity inside a circle to be initially rotating relative to a stationary surrounding fluid. The discontinuity in the velocity induces vorticity at the interface which produces intricate flow patterns. We compare PolyPIC to FLIP and APIC and see that it better resolves the vorticial flow.

In Figure 1 we simulate an ink droplet in an ambient incompressible fluid by dropping liquid onto a free surface. We only render the particles in the jet. Note that the ink and water are both simulated as the same incompressible fluid. We compare PolyPIC to FLIP and APIC and see that it again better captures the transition to turbulence. We note that PolyPIC works well even when the grid resolution is rather low. Figure 1 was run with a relatively low grid resolution $64 \times 256 \times 64$. We used 8 simulated particles per cell, and 8000 passively advected tracer particles per cell in a post-process for visualization.

Figure 2 demonstrates a 3D version of the vortex sheet. The cylinder is initially rotating about its axis relative to a stationary ambient fluid. It was also run on a low resolution grid ($88 \times 132 \times 88$) with 8 simulated particles per cell for simulation and 216 passively advected tracer particles per cell in a post-process for visualization. Despite the low resolution simulation, intricate flow patterns are observed.

For all incompressible flow examples we use constant, linear and multilinear modes (i.e. $N_r = 2^d$) with PolyPIC. This is the maximum number of modes we can use because the grid interpolation in the incompressible flow solver is multilinear ($N_B = 1$) and, as discussed in Section (§6.3.2), the number of reduced modes is bounded by $N_r \le (N_B + 1)^d$.

8.2 MPM elastoplasticity

In Figure 7 we demonstrated the increasingly energetic nature of PolyPIC elastoplasticity simulations as we add more polynomial modes. Note that with $N_r = 6$ modes the sand flows more freely and splashes off the jello more dramatically, while the Jell-O bounces more readily.

In Figure 10 we demonstrate the improved energy conservation of our method over APIC. In this scenario, a 2D hyperelastic square is initially compressed. The total energy of the system should be conserved with these initial and boundary conditions (zero traction). As we add more polynomial modes, the energy preservation improves. In Figure 9, we demonstrate how the increased energy retention affects the dynamics of a Jell-O cube dropped on the ground.

8.3 Accuracy and the number of modes

We verify that adding additional modes increases the accuracy of the simulation. In Figures 7 and 8 we examine the case of granular sand flowing from a container onto Jell-0. In Figure 7 we see that PolyPIC with $N_r = 4$ and APIC are less energetic than PolyPIC with $N_r = 6$. The flow of the sand in the container suffers from more numerical friction with PolyPIC $N_r = 4$ and APIC, therefore sand flows out of the container much slower. We can see this because the containers are still quite full in the final frame with PolyPIC $N_r = 4$ and APIC compared to PolyPIC with $N_r = 6$. In Figure 8 we rerun the same simulations but with higher grid and particle resolution. At this resolution, the PolyPIC $N_r = 4$ and APIC containers are all nearly empty in the final frame and as a result all flows are similarly energetic, indicating that PolyPIC with more modes gives a more accurate result since it is more predictive of the refined behavior.

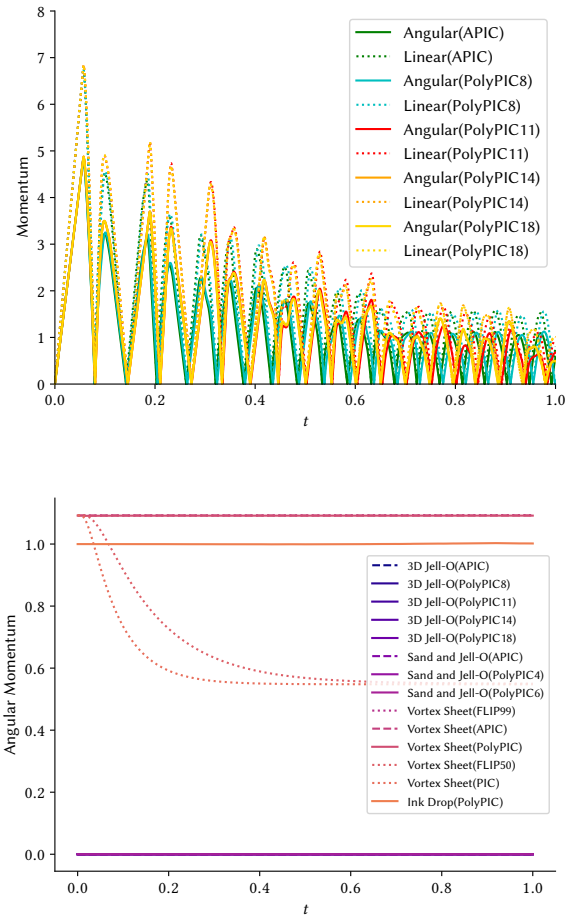


Fig. 11. **Momentum conservation**. The top figure plots the linear and angular momenta for the falling Jell-O's in Figure 9. The bottom illustrates the angular momentum loss resulting form transfers. We plot the momenta $\hat{\mathbf{l}}^n$ and $\hat{\mathbf{p}}^n$ from Equation (17) to monitor the transfers effects on conservation. APIC and PolyPIC preserve angular momentum during transfers, however the FLIP/PIC blends are commonly used in incompressible flow simulations do not. We illustrate this by comparing with increasing amounts of PIC.

8.4 Momentum conservation

We verify the angular momentum conservation properties of the PolyPIC transfers. In Figure 11 we plot the linear and angular momentum over the course of the time step for the falling Jell-O example shown in Figure 9. Even though the PolyPIC transfers conserve the momenta, the grid momentum update and the application of boundary conditions (Section (§6.2)) are not momentum conserving. To illustrate the conservation of the momentum in the transfers, we can monitor

$$\hat{\mathbf{l}}^n = \mathbf{l}^n_{P2G} + \sum_{m=1}^{n-1} \mathbf{l}^m_{\text{grid}} - \mathbf{l}^m_{P2G}, \; \hat{\mathbf{p}}^n = \mathbf{p}^n_{P2G} + \sum_{m=1}^{n-1} \mathbf{p}^m_{\text{grid}} - \mathbf{p}^m_{P2G} \quad (17)$$

	Seconds/Frame	$\Delta t_{\max}$	Particles	Cores
Ink Drop(FLIP99)	20.569	5×10^{-2}	3.64M	16
Ink Drop(APIC)	23.188	5×10^{-2}	3.64M	16
Ink Drop(PolyPIC)	31.466	5×10^{-2}	3.64M	16
Cylinder(PolyPIC)	146.744	2×10^{-1}	7.86M	12
Vortex Sheet(FLIP99)	2.367	1×10^{-1}	0.97M	20
Vortex Sheet(APIC)	2.739	1×10^{-1}	0.97M	12
Vortex Sheet(PolyPIC)	2.760	1×10^{-1}	0.97M	20
Sand & Jello(APIC)	11.582	4×10^{-5}	59.7K	12
Sand & Jello(PolyPIC4)	12.616	4×10^{-5}	59.7K	12
Sand & Jello(PolyPIC6)	17.682	4×10^{-5}	59.7K	12
Jello(APIC)	4.882	2×10^{-4}	17.5K	48
Jello(PolyPIC8)	5.713	2×10^{-4}	17.5K	48
Jello(PolyPIC11)	5.562	2×10^{-4}	17.5K	48
Jello(PolyPIC14)	5.512	2×10^{-4}	17.5K	48
Jello(PolyPIC18)	5.852	2×10^{-4}	17.5K	48

Table 3. We list the time step sizes, run times, particle counts and number of cores used for our simulations. We note that the Jell-O examples demonstrate that increasing the number of reduced modes N_r in PolyPIC only moderately increases the computational cost over APIC.

where $\mathbf{l}^n = \sum_{\mathbf{i}} \mathbf{x}_{\mathbf{i}} \times m_{\mathbf{i}} \mathbf{v}^n_{\mathbf{i}}$ and $\mathbf{p}^n = \sum_{\mathbf{i}} m_{\mathbf{i}} \mathbf{v}^n_{\mathbf{i}}$ are the angular and linear momenta on the grid. $\mathbf{l}^m_{P2G}$ $\mathbf{l}^m_{P2G}$ are computed after the transfer from particle to grid (Section (§6.1)) and $\mathbf{l}^m_{\text{grid}}$ and $\mathbf{p}^m_{\text{grid}}$ are computed after the grid momentum update (Section (§6.2)). The quantities $\mathbf{l}^m_{\text{grid}} - \mathbf{l}^m_{P2G}$ and $\mathbf{p}^m_{\text{grid}} - \mathbf{p}^m_{P2G}$ are the momenta lost during the grid momentum update at time step t^m. This is the only source of angular momentum loss for APIC and PolyPIC and thus the quantities in Equation (17) should be constant for those methods. We visualize the angular momentum loss from transfers in Figure 11. The straight lines indicate conservation.

9 DISCUSSION AND LIMITATIONS

While our method is a natural extension to the APIC approaches in [Jiang et al. 2015, 2017b], it has some apparent drawbacks. Adding more polynomial modes helps to increase the energy conservation during transfers which reduces numerical dissipation. However, numerical dissipation is often desirable. Most everyday examples of elasticity involve some type of damping term and numerical dissipation is often an acceptable approximation to this. Also, numerical dissipation in the transfers can help to stabilize the method. At large time steps PIC calculations will go unstable and numerical dissipation can increase the critical time step size at which instability dominates. Indeed we found that introducing too many modes for the hyperelastic simulations can lead to small time steps. In general, we set $\Delta t = \min(\frac{\text{CFL}\Delta x}{\text{max\_}v}, \Delta t_{\max})$ where max_v is the magnitude of the maximum velocity, 0<CFL<1 and $\Delta t_{\max}$ is typically about 1e-3 (see Table 3). If simulations go unstable, we shrink $\Delta t_{\max}$. We notice that $\Delta t_{\max}$ will decrease to around 1e-5 if we use larger numbers of modes and this can lead to longer run times. We typically use explicit symplectic Euler (SE) integration for the grid momentum update and energy loss in transfers helps to stabilize it.

Our approach incurs storage proportionate to the number of modes since each particle must store the coefficient of the polynomial bases used to locally represent the velocity field, however in the case of affine polynomials this is equivalent to storing the velocity and velocity derivative, thus for $Nr \approx d + 1$ the storage is approximately that of original APIC. Similarly, our transfers have computational cost that is linear in the number of reduced modes.

Therefore, the run time will increase slightly with additional modes. We demonstrate this by progressively adding more modes in the Jell-O examples in Figure 9. Run times are given in Table 3.

The number of nodes with nonzero weights implies a threshold on the number of velocity modes $N_r \leq (N_B+1)^d$. For larger N_r the system for $\mathbf{c}_p^{n+1}$ is overdetermined since there would be more modes than grid node velocity values to determine them from. In principle, our method would still work however we did not investigate this possibility. Interestingly, we noticed that with $N_r = (N_B+1)^d$ the transfer from grid to particle then back to grid is lossless (neglecting motion of the particles) (see [Fu et al. 2014]).

ACKNOWLEDGMENTS

The work is supported by NSF CCF-1422795, ONR (N000141110719, N000141210834), DOD (W81XWH-15-1-0147), Intel STC-Visual Computing Grant (20112360). UPenn authors additionally gratefully acknowledge the GPU donation from NVIDIA Corporation and a gift from Awowd, Inc.

REFERENCES

R. Ando, N. Thuerey, and C. Wojtan. 2015. A Stream Function Solver for Liquid Simulations. *ACM Trans Graph* 34 (2) (August 2015), 8.

R. Ando, N. Thurey, and R. Tsuruno. 2012. Preserving Fluid Sheets with Adaptively Sampled Anisotropic Particles. *IEEE Trans Vis Comp Graph* 18, 8 (Aug. 2012), 1202–1214.

R. Ando, N. Thurey, and C. Wojtan. 2013. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans Graph* 32, 4 (2013), 103:1–103:10.

R. Ando and R. Tsuruno. 2011. A particle-based method for preserving fluid sheets. In *Proc ACM SIGGRAPH/Eurographics Symp Comp Anim (SCA '11)*. 7–16.

C. Batty, F. Bertails, and R. Bridson. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Trans Graph* 26, 3 (2007).

C. Batty and R. Bridson. 2008. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim* (2008), 219–228.

L. Boyd and R. Bridson. 2012. MultiFLIP for energetic two-phase fluid simulation. *ACM Trans Graph* 31, 2 (2012), 16:1–16:12.

J. Brackbill. 1988. The ringing instability in Particle-In-Cell calculations of low-speed flow. *J Comp Phys* 75, 2 (1988), 469–492.

J. Brackbill and H. Ruppel. 1986. FLIP: A method for adaptively zoned, Particle-In-Cell calculations of fluid flows in two dimensions. *J Comp Phys* 65 (1986), 314–343.

N. Chentanez and M. Muller. 2014. Coupling 3D Eulerian, height field and particle methods for the simulation of large scale liquid phenomena. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim (SCA '14)*.

G. Daviet and F. Bertails-Descoubes. 2016. A Semi-implicit Material Point Method for the Continuum Simulation of Granular Materials. *ACM Trans Graph* 35, 4 (2016), 102:1–102:13.

M. Desbrun and M. Cani. 1996. Smoothed Particles: A new paradigm for animating highly deformable bodies. In *Eurographics Workshop on Computer Animation and Simulation (EGCAS)*, R. Boulic and G. Hegron (Eds.). Springer-Verlag, 61–76.

E. Edwards and R. Bridson. 2012. A high-order accurate Particle-In-Cell method. *Int J Numer Meth Eng* 90 (2012), 1073–1088.

Y. Fan, J. Litven, D. Levin, and D. Pai. 2013. Eulerian-on-lagrangian Simulation. *ACM Trans Graph* 32, 3 (2013), 22:1–22:9.

F. Ferstl, R. Ando, C. Wojtan, R. Westermann, and N. Thuerey. 2016. Narrow Band FLIP for Liquid Simulations. *Comp Graph For* 35(2) (May 2016), 8.

N. Foster and D. Metaxas. 1996. Realistic animation of liquids. *Graph Mod Imag Proc* 58 (1996), 471–483.

C. Fu, Q. Guo, T. Gast, C. Jiang, and J. Teran. 2014. Material Point Method for Coupled Hydromechanical Problems. *Supplementary Technical Document* (2014).

Y. Gao, C. Li, S. Hu, and B. Barsky. 2009. Simulating gaseous fluids with low and high speeds. *Comp Graph Forum* 28, 28 (2009), 1845–1852.

D. Gerszewski and A. Bargteil. 2013. Physics-based Animation of Large-scale Splashing Liquids. *ACM Trans Graph* 32, 6 (2013), 185:1–185:6.

O. Gonzalez and A. Stuart. 2008. *A first course in continuum mechanics*. Cambridge University Press.

C. Gritton and M. Berzins. 2017. Improving accuracy in the MPM method using a null space filter. *Comp Part Mech* 4, 1 (2017), 131–142.

C. Hammerquist and J. Nairn. 2017. A new method for material point method particle updates that reduces noise and enhances stability. *Comp Meth App Mech Eng* 318 (2017), 724 – 738.

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

F. Harlow and E. Welch. 1965. Numerical Calculation of Time Dependent Viscous Flow of Fluid with a Free Surface. *Phys Fluid* 8, 12 (1965), 2182–2189.

W. Hong, D. House, and J. Keyser. 2008. Adaptive particles for incompressible fluid simulation. *Vis Comp* 24, 7 (2008), 535–543.

W. Hong, D. House, and J. Keyser. 2009. An adaptive sampling approach to incompressible particle-based fluid. *Theory Pract Comp Graph* (2009), 69–76.

C. Jiang, T. Gast, and J. Teran. 2017a. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Trans Graph* 36, 4 (2017).

C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. 2015. The Affine Particle-In-Cell Method. *ACM Trans Graph* 34, 4 (2015), 51:1–51:10.

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

G. Klár, T. Gast, A. Pradhana, C. Fu, C. Schroeder, C. Jiang, and J. Teran. 2016. Drucker-prager Elastoplasticity for Sand Animation. *ACM Trans Graph* 35, 4 (2016), 103:1–103:12.

E. Larionov, C. Batty, and R. Bridson. 2017. Variational Stokes: A Unified Pressure-Viscosity Solver for Accurate Viscous Liquids. *ACM Trans Graph* 36, 4 (2017).

H. Lee, J. Hong, and C. Kim. 2009. Interchangeable SPH and level set method in multiphase fluids. *Vis Comp* 25, 5 (2009), 713–718.

D. Levin, J. Litven, G. Jones, S. Sueda, and D. Pai. 2011. Eulerian Solid Simulation with Contact. *ACM Trans Graph* 30, 4 (2011), 36:1–36:10.

F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. 2008. Two-way coupled SPH and Particle Level Set fluid simulation. *IEEE Trans Vis Comp Graph* 14 (2008), 797–804.

A. McAdams, A. Selle, K. Ward, E. Sifakis, and J. Teran. 2009. Detail Preserving Continuum Simulation of Straight Hair. *ACM Trans Graph* 28, 3 (2009), 62:1–62:6.

O. Mercier, C. Beauchemin, N. Thuerey, T. Kim, and D. Nowrouzezahrai. 2015. Surface Turbulence for Particle-Based Liquid Simulations. *ACM Trans Graph* 34(6) (Nov 2015), 10.

R. Narain, A. Golas, S. Curtis, and M. Lin. 2009. Aggregate Dynamics for Dense Crowd Simulation. *ACM Trans Graph* 28, 5 (2009), 122:1–122:8.

R. Narain, A. Golas, and M. Lin. 2010. Free-flowing granular materials with two-way solid coupling. *ACM Trans Graph* 29, 6 (2010), 173:1–173:10.

D. Ram, T. Gast, C. Jiang, C. Schroeder, A. Stomakhin, J. Teran, and P. Kavehpour. 2015. A material point method for viscoelastic fluids, foams and sponges. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*. 157–163.

K. Raveendran, C. Wojtan, and G. Turk. 2011. Hybrid SPH. In *Proc 2011 ACM SIGGRAPH/Eurograp Symp Comp Anim (SCA '11)*. 33–42.

Wolfram Research. 2016. Mathematica 11.0. (2016).

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

A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle. 2013. A Material Point Method for snow simulation. *ACM Trans Graph* 32, 4 (2013), 102:1–102:10.

A. Stomakhin, C. Schroeder, C. Jiang, L. Chai, J. Teran, and A. Selle. 2014. Augmented MPM for phase-change and varied materials. *ACM Trans Graph* 33, 4 (2014), 138:1–138:11.

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

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

A. P. Tampubolon, T. Gast, G. Klár, C. Fu, J. Teran, C. Jiang, and K. Museth. 2017. Multi-species simulation of porous sand and water mixtures. *ACM Trans Graph* 36, 4 (2017).

Y. Teng, D. Levin, and T. Kim. 2016. Eulerian Solid-fluid Coupling. *ACM Trans Graph* 35, 6 (2016), 200:1–200:8.

N. Thuerey and T. Pfaff. 2016. MantaFlow. (2016). *http://mantaflow.com*.

K. Um, S. Baek, and J. Han. 2014. Advanced hybrid particle-grid method with sub-grid particle correction. *Comp Graph Forum* 33 (2014), 209–218.

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

Y. Yue, B. Smith, C. Batty, C. Zheng, and E. Grinspun. 2015. Continuum foam: a material point method for shear-dependent flows. *ACM Trans Graph* 34, 5 (2015), 160:1–160:20.

X. Zhang, M. Li, and R. Bridson. 2016. Resolving fluid boundary layers with particle strength exchange and weak adaptivity. *ACM Trans Graph (TOG)* 35, 4 (2016), 76.

B. Zhu, X. Yang, and Y. Fan. 2010. Creating and preserving vortical details in SPH fluid. *Comp Graph Forum* 29, 7 (2010), 2207–2214.

F. Zhu, J. Zhao, S. Li, Y. Tang, and G. Wang. 2016. Dynamically Enriched MPM for Invertible Elasticity. In *Comp Graph For*. Wiley Online Library.

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