An angular momentum conserving Affine-Particle-In-Cell method

Chenfanfu Jiang, Craig Schroeder, Joseph Teran[1]

Department of Mathematics
University of California Los Angeles

Abstract

We present a new technique for transferring momentum and velocity between particles and grid with Particle-In-Cell (PIC) [1] calculations which we call Affine-Particle-In-Cell (APIC). APIC represents particle velocities as locally affine, rather than locally constant as in traditional PIC. We show that this representation allows APIC to conserve linear and angular momentum across transfers while also dramatically reducing numerical diffusion usually associated with PIC. Notably, conservation is achieved with lumped mass, as opposed to the more commonly used Fluid Implicit Particle (FLIP) [2, 3] transfers which require a "full" mass matrix for exact conservation. Furthermore, unlike FLIP, APIC retains a filtering property of the original PIC and thus does not accumulate velocity modes on particles as FLIP does. In particular, we demonstrate that APIC does not experience velocity noise that is characteristic of FLIP in a number of Material Point Method (MPM) hyperelasticity calculations. Lastly, we demonstrate that when combined with the midpoint rule for implicit update of grid momentum that linear and angular momentum are exactly conserved.

Keywords: PIC, FLIP, MPM, APIC, hybrid Lagrangian/Eulerian, particle-grid

1. Introduction

PIC methods have been used for decades to simulate many different physical phenomena. Examples include compressible flow, incompressible flow, plasma physics, computational solids and many more [4]. PIC utilizes a hybrid particle/grid representation of material to retain the accuracy of Lagrangian techniques without sacrificing the robustness of Eulerian techniques. In all cases, the hybrid nature of the approach requires the transfer of state to and from Lagrangian particles and Eulerian grid. Unfortunately, this frequent remapping can introduce significant error and noise. The most apparent error is excessive dissipation incurred from double interpolation. The FLIP approach of Brackbill et al. [2, 3] was developed to reduce the dissipation by transferring *changes* in grid quantities to particles, rather than directly interpolating as in PIC. This also greatly improved the angular momentum conservation properties of the particle/grid transfers [5, 6, 7]. However, as pointed out in [7] exact conservation with FLIP is only possible with the use of the "full"

[1] jteran@math.ucla.edu

mass matrix. Unfortunately, since the full mass matrix can be singular for certain particle configurations, it is necessary in practice to interpolate between a mass-lumped and full mass matrix to avoid issues caused by a poorly conditioned mass matrix [7]. However, even with mass lumping, FLIP greatly reduces the angular momentum losses from transfers in the original PIC.

While all PIC approaches suffer to some degree from finite grid [8, 9] (or ringing [10, 11]) instabilities, FLIP appears to exacerbate null modes in the transfer operator from particle to grid. This is particularly true with MPM [12, 13] PIC techniques for simulating history dependent materials. The problems arise from the mismatch in particle and grid degrees of freedom. Typically there are many more particles than grid nodes and thus information is lost in the particle to grid transfer. While the original PIC transfers can be seen as a filter of particle degrees of freedom by modes resolvable on the grid, FLIP does not have this property. FLIP transfers can be shown to cause unpredictable behavior since certain particle velocity modes persist, invisible to the dynamics on the grid and reappear after particle movement. Notably, the particle velocities are not used to move the particle positions. Particle positions are directly interpolated from the grid, which is equivalent to using an interpolated, PIC velocity for position updates. This idea has also been used for example in [14]. However, while this reduces the effect of the velocity modes greatly, it does not completely remove the problem. We illustrate this in Figure 1. Other recent results in MPM have noted similar issues with noise in FLIP transfers [15, 16, 17, 18].

The typical PIC transfer of particle velocities $\mathbf{v}_p$ to grid velocities $\mathbf{v}_i$ is done by first transferring mass and momentum from particle to grid and then dividing out mass to get velocity as

$$m_i = \sum_p m_p N(\mathbf{x}_p - \mathbf{x}_i), \quad (m\mathbf{v})_i = \sum_p m_p \mathbf{v}_p N(\mathbf{x}_p - \mathbf{x}_i), \quad \mathbf{v}_i = \frac{1}{m_i}(m\mathbf{v})_i \tag{1}$$

where $\mathbf{x}_p$ and $\mathbf{x}_i$ are particle and grid node locations and $N(\mathbf{x}_p - \mathbf{x}_i)$ represent interpolating functions defined on the grid. After a physical update of the momentum is done on the grid, new grid velocities $\tilde{\mathbf{v}}_i$ are then directly interpolated to particles as

$$\tilde{\mathbf{v}}_p = \sum_i \tilde{\mathbf{v}}_i N(\mathbf{x}_p - \mathbf{x}_i). \tag{2}$$

With this simple convention, linear and angular momentum are conserved in the transfer from particle to grid as long as the interpolating functions satisfy a partition of unity property. In the transfer from grid to particle, linear momentum is conserved, but angular momentum is not. Notably, these transfers are linear operations, and since there are typically many more particles than grid degrees of freedom, there are particle velocity null modes that are lost when transferring to grid. Since the PIC transfer from grid to particle is just interpolation, this process can be seen as filtering out particle velocity modes that are not seen on the grid. The loss of the kinetic energy in these modes is what leads to the excessive dissipation of PIC.

The energy loss in PIC style transfers is unacceptable for many application areas, and FLIP style transfers can be used instead. FLIP uses the same transfer from particles to grid as PIC, however with FLIP, velocities

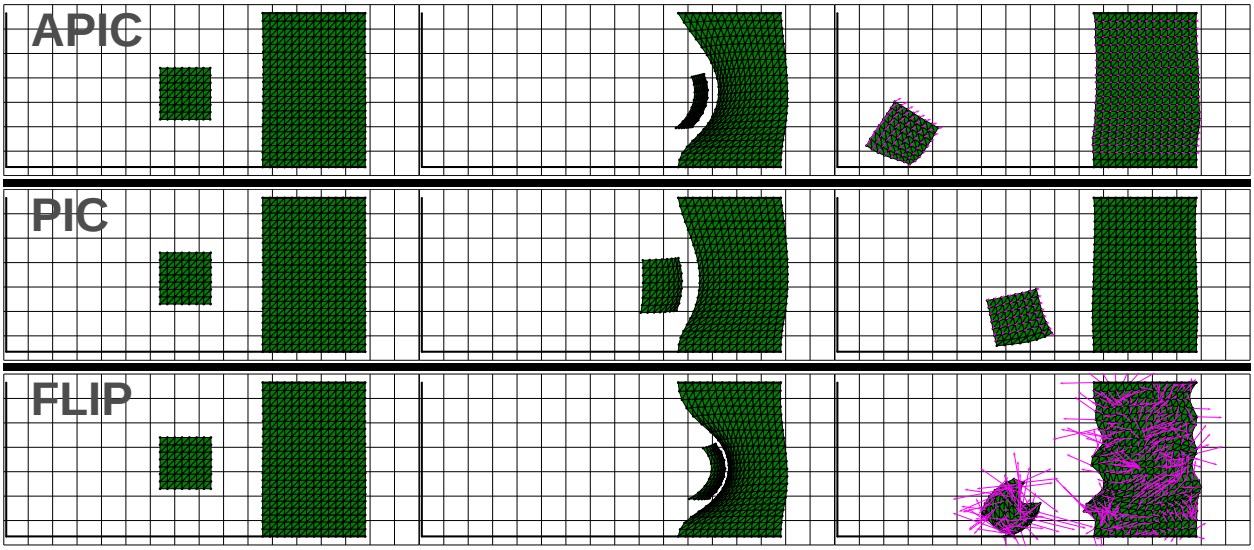

Figure 1: Ringing test. An elastic cube with initial velocity pointing to the right hits an elastic wall with its top and bottom fixed. (See section 6.7 for more details.) Particle velocities are drawn to illustrate the noisy modes persistent with FLIP transfers. PIC and APIC transfers do not suffer from this due to the filtering property. However, APIC is not excessively damped like PIC. The time points shown here are $t = \frac{2}{24}\,s$, $t = \frac{5}{24}\,s$, and $t = \frac{40}{24}\,s$. The target has almost stopped moving by the last frame in each case, so the final resting configuration for the target in each case closely resembles the last frame. Notably the particle velocities are non-zero when the simulation comes to rest, and thus consist of purely transfer null modes.

are incremented by interpolated differences in grid velocities (rather than directly interpolated as in PIC) when transferring from grid to particles

$$\tilde{\mathbf{v}}_p = \mathbf{v}_p + \sum_i (\tilde{\mathbf{v}}_i - \mathbf{v}_i) N(\mathbf{x}_p - \mathbf{x}_i). \tag{3}$$

Since velocities are incremented, rather than overwritten with information from the grid, energy in particle null modes is not lost and thus the excessive dissipation is avoided. However, these modes are still invisible to the grid, since the transfer from particle to grid is the same in PIC and FLIP. Thus, although these modes are not lost, they have no direct effect on the governing physics which can lead to unpredictable behavior like those discussed in [8, 9, 10, 11, 19, 20, 15, 16, 17, 18].

We present a new technique designed to retain the filtering property of the original PIC transfers to guarantee stable behavior. We do this by storing both velocity and velocity derivatives on each particle. This can be thought of as adding an extra Taylor series term to increase accuracy. This idea was also used in [21], however they used FLIP transfers and thus do not attain the filtering property. We show that by representing particle velocities as locally affine, rather than locally constant, particle/grid transfers can be defined that: (1) filter out null modes invisible to the grid, (2) have dissipation comparable to that of FLIP and (3) conserve angular and linear momentum (both from particle to grid and grid to particle). Furthermore, this is all done with simple mass lumping foregoing the need for poorly conditioned full mass matrices.

There are a few existing approaches that use similar ideas to what we propose. Our work builds on that of Jiang et al [22]. The transfers used there are discretely angular momentum conserving only for explicit symplectic Euler integration. For any other integration scheme, angular momentum may be gradually lost. Wallstedt and Guilkey also augment particles with derivatives of the field variables from grid to reduce dissipation in [21]. However they still use FLIP style incremental updates and thus still suffer from null mode persistence. Furthermore, their transfer from grid to particle is not angular momentum conserving. Gritton et al [18, 17] have recently developed methods to resolve the noise introduced by transfer null modes by taking the SVD of the transfer operator. Nairn [15] use a linear combination of the FLIP and PIC transfers to reduce noise. Hammerquist and Nairn [16] also develop a modified PIC technique designed to reduce the noise of FLIP transfers by retaining the filtering property of PIC. Also, our approach is similar to some aspects of the Constrained Interpolation Profile (CIP) methods which also store derivative information to reduce diffusion and improve conservation, but for semi-Lagrangian interpolation [23].

The choice of interpolating functions is very important in MPM applications. Indeed this is also true in our development of particle/grid transfers and we discuss some requirements in Section 3.2. As pointed out in [24], discontinuities in the derivatives of interpolating functions lead to discontinuties in the flow map derivatives (or deformation gradient), and as a consequence, discontinuties in stress as particles move across interpolation support boundaries. GIMP interpolting functions developed by Bardenhagen and Kober

[25] can be used to fix this. Steffen et al [24] show that B-spline interpolation also works. We adopt this convention and use qudratic or cubic B-spline interolation in all of our examples. In some examples we additionally use the Lagrangian FEM update of deformation gradient outlined in [22], which Nguyen et al [26] note is equivalent to convected particle domain interpolation (CPDI) [27].

2. Momentum transfers

Our method is a mathematical means for switching between particle based samples of momenta and grid based samples of momenta. It is agnostic to the physical behavior of the material. It simply serves as alternative means of performing a process that is common to most if not all PIC calculations. As with [21], we note that the momenta transfer processes can be improved by augmenting particles with velocity derivatives. This addition requires an appropriate modification to the traditional PIC and FLIP transfer strategies. Here, we summarize the main technical ideas behind our grid/particle transfers. The purpose of this section is to give the intuition for the development of the transfers. The formulas for the transfers we advocate are slightly different and are given in Sections 3.3 and 3.5. For example, in this section we do not consider the motion of the grid positions over the time step to simplify the notation.

The particle momentum state is represented by a sample of mass m_p, velocity $\mathbf{v}_p$ and velocity derivative $\mathbf{C}_p$. The grid momentum state is represented by node samples of mass m_i and velocity $\mathbf{v}_i$. Our transfer from particle to grid determines the grid quantities m_i and $\mathbf{v}_i$ in terms of the particle quantities $m_p, \mathbf{v}_p$ and $\mathbf{C}_p$. We then assume, as is generally the case, that grid momenta are updated according so some arbitrary physical equations. We also assume that grid mass is not changed during the update and thus grid momenta change as $\mathbf{v}_i$ become $\tilde{\mathbf{v}}_i$. Thus, we describe the process of transferring from grid to particle in terms of determining new particle state $\tilde{\mathbf{v}}_p$ and $\tilde{\mathbf{C}}_p$ in terms of the updated grid velocities $\tilde{\mathbf{v}}_i$.

Note that as in [21] we do not transfer velocities gradients $\mathbf{C}_p$ from particle to grid, rather we use them to define grid momenta more accurately. In fact, our transfer from particle to grid is the same as that in [21]. However, where they use a FLIP transfer of grid velocities to particles, we design a grid to particle transfer that retains the filtering property of the original PIC.

In the discussion that follows we use conservation of linear and angular momenta when transfering from grid to particle and vice verse as a motivation. We empahsize that this conservation is only related to the mathematical change of state from particle mass, velocity and veloctiy gradient, to grid mass and velocity (and vice verse). This conservation is unrelated to any such behavior that arises physically and is simply a desirable property to attain when switching between discrete represnetations of the same material. Furthermore, we note that while other transfers may not achieve this perfect conservation property, their errors are typically proportionate to grid resolution and decrease under refinement of discrete space and time.

2.1. Rigid-Particle-In-Cell (RPIC)

The intuition for our transfers is largely derived from a simpler case: particle-wise rigid-body velocity. This can be thought of as defining a velocity field local to $\mathbf{x}_p$ as $\mathbf{v}(\mathbf{x}) = \mathbf{v}_p + \mathbf{C}_p(\mathbf{x} - \mathbf{x}_p)$ with skew symmetric $\mathbf{C}_p$. That is, $\mathbf{C}_p = \boldsymbol{\omega}_p^*$ where $\boldsymbol{\omega}_p$ is the angular velocity of the rigid body and $\mathbf{C}_p = \boldsymbol{\omega}_p^*$ is the skew symmetric matrix equivalent to $\mathbf{C}_p\mathbf{x} = \boldsymbol{\omega}_p \times \mathbf{x}$ for arbitrary vector $\mathbf{x}$. We refer to this case as RPIC. We do not recommend using RPIC since it ultimately suffers from excessive dissipation similar to PIC [22]. Nonetheless, it provides most of the insights needed for making transfers with general affine conservative so we present them here.

2.1.1. Particle to grid

With a piecewise rigid assumption, we conceptualize particle $\mathbf{x}_p$ as a rigid body consisting of point masses that the particle distributes to the grid with a standard PIC transfer: $m_{ip} = m_p N(\mathbf{x}_p - \mathbf{x}_i)$. That is, rigid body p consists of point masses m_{ip} located at $\mathbf{x}_i$. Note that this rigid body then has inertia tensor $\mathbf{K}_p = \sum_i m_{ip}(\mathbf{x}_i - \mathbf{x}_p)^*(\mathbf{x}_i - \mathbf{x}_p)^{*T}$. Also note that the standard PIC grid mass is then $m_i = \sum_p m_{ip}$. With this idealization, the linear momenta of the points in the rigid body are then $(m\mathbf{v})_{ip} = m_{ip}(\mathbf{v}_p + \mathbf{C}_p(\mathbf{x}_i - \mathbf{x}_p))$ where again $\mathbf{C}_p$ is assumed to be skew symmetric to represent rigid body velocity. We can thus define the grid linear momenta to be the sum of the contributions from all rigid bodies p: $(m\mathbf{v})_i = \sum_p (m\mathbf{v})_{ip}$. Notably, this transfer is equivalent to the particle to grid transfer formulas used in [21].

The transfer conserves linear and angular momenta in the following sense. Define the total linear momentum of all particles as $\mathbf{p}^P = \sum_p m_p \mathbf{v}_p$ and the total angular momentum (about the origin) as $\mathbf{L}^P = \sum_p \mathbf{K}_p\boldsymbol{\omega}_p + \mathbf{x}_p \times m_p\mathbf{v}_p$ (see Section 5.1 for justification of these definitions). After the transfer from particle to grid, we have $\mathbf{p}^G = \sum_i (m\mathbf{v})_i$ and $\mathbf{L}^G = \sum \mathbf{x}_i \times (m\mathbf{v})_i$ as the analogous quantities defined over the grid. It can be shown that $\mathbf{p}^P = \mathbf{p}^G$ and $\mathbf{L}^P = \mathbf{L}^G$ (see Section 5.1 for details). That is, we can say that the linear and angular momentum of the grid state is the same as that of the particle rigid body state after the transfer from particle to grid.

2.1.2. Grid to particle

The transfer from grid to particle is done after a momentum update on the grid. Using $\tilde{\mathbf{v}}_i$ to denote the grid velocities after the grid momentum update (again assuming that the grid m_i do not change over the step), we design transfers of $\tilde{\mathbf{v}}_i$ to get $\tilde{\mathbf{v}}_p$ and skew $\tilde{\mathbf{C}}_p$ that give a rigid body state whose linear and angular momentum are consistent with that of the updated grid state. That is, we want $\tilde{\mathbf{v}}_p$ and skew $\tilde{\mathbf{C}}_p = \tilde{\boldsymbol{\omega}}_p^*$ such that the new linear momentum is conserved $\tilde{\mathbf{p}}^G = \sum_i m_i\tilde{\mathbf{v}}_i = \sum_p m_p\tilde{\mathbf{v}}_p = \tilde{\mathbf{p}}^P$ and new angular momentum is conserved $\tilde{\mathbf{L}}^G = \sum \mathbf{x}_i \times m_i\tilde{\mathbf{v}}_i = \sum_p (\mathbf{K}_p\tilde{\boldsymbol{\omega}}_p + \mathbf{x}_p \times m_p\tilde{\mathbf{v}}_p) = \tilde{\mathbf{L}}^P$. If we define the transfer of the linear velocity as with standard PIC, $\tilde{\mathbf{v}}_p = \sum_i \tilde{\mathbf{v}}_i N(\mathbf{x}_p - \mathbf{x}_i)$, then linear momentum is conserved, as with PIC. However, with this transfer alone, angular momentum is lost. Specifically, it can be shown that local to

particle p, $\tilde{\mathbf{l}}_p = \sum_i (\mathbf{x}_i - \mathbf{x}_p) \times m_{ip}\tilde{\mathbf{v}}_i$ is lost. This arises from representing the information in the grid state $m_{ip}\tilde{\mathbf{v}}_i$ as only $m_p\mathbf{v}_p$. Clearly, one particle can not represent the angular momentum seen on the grid in $m_{ip}\tilde{\mathbf{v}}_i$. The idea is to represent that angular momentum in a rigid body, rather than a simply translating body to prevent the loss. Thus, if we define angular velocity $\tilde{\boldsymbol{\omega}}_p$ to be $\tilde{\boldsymbol{\omega}}_p = \mathbf{K}_p^{-1}\tilde{\mathbf{l}}_p$ (and $\tilde{\mathbf{C}}_p = \tilde{\boldsymbol{\omega}}_p^*$), then a simple argument shows that both linear and angular momentum are conserved in the transfer from grid to particle. That is, the transfers give a rigid body state whose linear and angular momentum are consistent with that of the updated grid state. See Section 5.1 for proofs of these claims. Also, we again note that we have neglected to consider the motion of the grid nodes and the particles over the step in our discussion of conservation. These should be taken into account, and we do consider this for the transfers advocated in Section 3.5. We only omit this in the precededing discussion for simplification of notation, it does not effect the main idea of the transfers essentially.

2.2. Affine-Particle-In-Cell (APIC)

While the RPIC treatment of angular velocity does greatly improve the angular momentum conservation properties of the original PIC, a full velocity gradient is needed to reduce dissipation sufficiently. We refer to this strategy as APIC. For APIC, we will extend the particle-wise, local velocity field to be an arbitrary affine function as $\mathbf{v}(\mathbf{x}) = \mathbf{v}_p + \mathbf{C}_p(\mathbf{x} - \mathbf{x}_p)$. Here the matrix $\mathbf{C}_p$ is fully arbitrary, unlike the skew symmetric view in RPIC. The problem then is to determine the transfers from particle to grid and vice versa. This can be done in a manner directly analogous to what was presented in Section 2.1, and we provide those details in Section 5.6. However, when developing a scheme that is perfectly conservative over the entire time step (i.e., both transfers and grid updates are conservative), a more general notion of transfer is useful. The discussion of transfers so far has assumed that information will be transferred from particles to the grid and then immediately back to particles without any other changes in grid or particle positions. While we show that these transfers can be made perfectly conservative, this is typically not enough in practice. The point of hybrid particle/grid schemes is that part of the evolution will occur on the grid. This introduces an element of time into the conservation problem. For example, immediately following the transfer from particle to grid, the angular momentum should be computed as $\mathbf{L}^G = \sum_i \mathbf{x}_i^n \times m_i^n \mathbf{v}_i^n$. Before the transfer back to particles, the grid state will have changed, and angular momentum will be computed as $\mathbf{L}^G = \sum_i \mathbf{x}_i^{n+1} \times m_i^n \mathbf{v}_i^{n+1}$. We introduce a degree of flexibility into the definition of the APIC transfers to account for this. When transferring to the particles, we have access to $\mathbf{x}_p^n$, $\mathbf{x}_i^n$, $\mathbf{x}_p^{n+1}$, and $\mathbf{x}_i^{n+1}$, which gives us more possible options. We are also free to choose the state that we store. For example with RPIC, rather than storing angular velocity $\boldsymbol{\omega}_p$ as state, we could store rotational angular momentum $\tilde{\mathbf{l}}_p$. This additional flexibility is very useful, since it allows us to obtain additional properties from the method. We require our transfers to be generally of the form described above, subject to the additional flexibility that has been noted.

Now that we have broadened our search space of possible transfers, we need to narrow down the possibilities. We narrow the field of choices down to a single scheme by enforcing three properties:

1. A globally affine velocity field should be preserved across transfers from particles to the grid and back when moving particles and moving grids are ignored (for example when $\Delta t = 0$).
2. The transfers should conserve linear and angular momentum, even when the complications of grid-based evolution, moving grids, and moving particles are taken into account.
3. A simulation with a single particle is stable but non-dissipative when moving grids and moving particles are taken to account but additional grid-based influences (forces, etc.) are ignored.

Property 1 is what it means to be an APIC scheme; it is a PIC-style transfer that preserves affine velocity fields. Note that this property should only be enforced under very strict circumstances ($\Delta t = 0$), since affine velocity fields should be able to change due to advection. Property 2 ensures that the entire scheme will conserve linear and angular momentum provided that the grid-based scheme also conserves these quantities.

Property 3 is a non-obvious but crucial requirement. The other properties do not uniquely determine a transfer; they only narrow it down to a one-parameter family of transfers. These transfers tend to behave similarly *except* when one particle moves far enough from other particles that it is able to evolve in isolation. For one particular member of this family, a lone particle will evolve by not changing. For the rest of the members of this family, part of the particle's state tends to explode or decay exponentially when the particle evolves in isolation. Exponential decay is not desirable, and exponential growth is intolerable. This leads us to choose the stability criterion to narrow the possibilities down to one set of transfers. We present these transfers in the context of the MPM method in which we use them in Sections 3.3 and 3.5. Also, we present a derivation of the transfers from the properties 1-3 in Section 5.8.

3. Method

Here we outline the governing equations for an MPM discretization of hyperelastic materials and we establish some notation used throughout the exposition. We emphasize that the focus of our paper is on the transfers between the grid and particle representations of momentum. The integration scheme applied on the grid, and indeed the constitutive physical behavior of the material itself is independent of this. We chose a hyperelastic formulation because it simplifies the implementation, exposition and analysis. Also, it is representative of many MPM applications. A different constitutive law could be substituted without change to the transfers. Similarly, while we focus on the midpoint rule for grid momentum updates because of its favorable conservation properties, it could be replaced with another integration scheme without impacting our discussion of transfers. If the alternate constitutive law and grid-based integration scheme conserve linear or angular momentum, the overall numerical scheme will as well.

3.1. Equations

Let $\mathbf{x} = \boldsymbol{\phi}(\mathbf{X}, t)$ be the mapping from material coordinates $\mathbf{X}$ to world coordinates $\mathbf{x}$. Let $\mathbf{V}$ and $\mathbf{v}$ be the Lagrangian and Eulerian velocities. $\mathbf{F}$ is the deformation gradient, and J is its determinant. That is,

$$\mathbf{V}(\mathbf{X}, t) = \frac{\partial \mathbf{x}}{\partial t}(\mathbf{X}, t) \tag{4}$$

$$\mathbf{v}(\mathbf{x}, t) = \mathbf{V}(\boldsymbol{\phi}^{-1}(\mathbf{x}, t), t) \tag{5}$$

$$\mathbf{F}(\mathbf{X}, t) = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}(\mathbf{X}, t) \tag{6}$$

$$J = \det(\mathbf{F}) \tag{7}$$

With these definitions, the evolution equations are

$$\rho \frac{D\mathbf{v}}{Dt} = \nabla \cdot \boldsymbol{\sigma}, \tag{8}$$

where the Cauchy stress $\boldsymbol{\sigma}$ is related to the first Piola-Kirchhoff stress $\mathbf{P}$ and hyperelastic energy density Ψ through

$$\boldsymbol{\sigma} = \frac{1}{J}\mathbf{P}\mathbf{F}^T \tag{9}$$

$$\mathbf{P} = \frac{\partial \Psi}{\partial \mathbf{F}} \tag{10}$$

The state of stress in hyperelastic materials is simply related to $\mathbf{F}$ as $\Psi(\mathbf{F})$ and $\mathbf{P}(\mathbf{F})$ where the total internal potential energy Φ is

$$\Phi(t) = \int_{\Omega_0} \Psi(\mathbf{F}(\mathbf{X}, t))\, d\mathbf{X}. \tag{11}$$

Since we will not have access to a reference configuration, we must evolve our deformation gradient according to

$$\frac{\partial \mathbf{F}}{\partial t}(\mathbf{X}, t) = \frac{\partial \mathbf{v}}{\partial \mathbf{x}}(\boldsymbol{\phi}(\mathbf{X}, t), t)\mathbf{F}(\mathbf{X}, t). \tag{12}$$

We seek to conserve total momentum $\mathbf{p}(t)$ and total angular momentum $\mathbf{l}(t)$, which are given by

$$\mathbf{p}(t) = \int_{\Omega} \rho(\mathbf{x}, t)\mathbf{v}(\mathbf{x}, t)\, d\mathbf{x} \tag{13}$$

$$\mathbf{l}(t) = \int_{\Omega} \mathbf{x} \times \rho(\mathbf{x}, t)\mathbf{v}(\mathbf{x}, t)\, d\mathbf{x} \tag{14}$$

For completeness, kinetic energy is

$$T(t) = \int_{\Omega} \rho(\mathbf{x}, t)\|\mathbf{v}(\mathbf{x}, t)\|^2\, d\mathbf{x}, \tag{15}$$

and total energy is $E = T + \Phi$.

3.2. Interpolation weights

As with PIC, we use weights to transfer information between the two representations. While the choice of weights is flexible, we require them to satisfy some important properties. Let $N(\mathbf{x})$ be an interpolation kernel, which must be chosen to satisfy

$$\sum_i N(\mathbf{x}_p^n - \mathbf{x}_i^n) = 1 \tag{16}$$

$$\sum_i \mathbf{x}_i^n N(\mathbf{x} - \mathbf{x}_i^n) = \mathbf{x} \tag{17}$$

for any $\mathbf{x}$. The kernel $N(\mathbf{x})$ is used to define interpolation weights and weight gradients as

$$w_{ip}^n = N(\mathbf{x}_p^n - \mathbf{x}_i^n) \tag{18}$$

$$\nabla w_{ip}^n = \nabla N(\mathbf{x}_p^n - \mathbf{x}_i^n). \tag{19}$$

The properties above lead to properties for w_{ip}^n and ∇w_{ip}^n.

$$\sum_i w_{ip}^n = 1 \tag{20}$$

$$\sum_i w_{ip}^n \mathbf{x}_i^n = \mathbf{x}_p^n \tag{21}$$

$$\sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n) = \mathbf{0} \tag{22}$$

$$\sum_i \mathbf{x}_i^n (\nabla w_{ip}^n)^T = \mathbf{I} \tag{23}$$

With these weights defined, we can start describing the method. We note that we use either quadratic or cubic B-splines as in Steffen et al[24] as they satisfy the above criteria and are continuously differentiable.

3.3. Transfer to grid

Each particle $\mathbf{x}_p^n$ stores mass m_p, velocity $\mathbf{v}_p^n$, and the additional matrix $\mathbf{B}_p^n$. As we are using MPM, we also store a deformation gradient $\mathbf{F}_p^n$ on particles. Note that particle masses m_p do not have a time superscript because they are constant (and thus never updated from the grid) to account for conservation of mass. We first use our weights to interpolate mass and momentum to the grid.

$$m_i^n = \sum_p m_p w_{ip}^n \tag{24}$$

$$\mathbf{D}_p^n = \sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n)(\mathbf{x}_i^n - \mathbf{x}_p^n)^T \tag{25}$$

$$m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p (\mathbf{v}_p^n + \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n)) \tag{26}$$

The velocity $\mathbf{v}_i^n$ is obtained by division. Note that unlike with m_p, we specify a time superscript on grid mass m_i^n, since it will change each time step. The additional matrix $\mathbf{D}_p^n$ used in the transfer is similar to an inertia

tensor (but for an affine rather than rigid motion). Similarly, $\mathbf{B}_p^n$ contains angular momentum information and the local affine velocity field is conceptually $\mathbf{v}_p^n + \mathbf{B}_p^n(\mathbf{D}_p^n)^{-1}(\mathbf{x}_i^n - \mathbf{x}_p^n)$ with matrix $\mathbf{C}_p^n = \mathbf{B}_p^n(\mathbf{D}_p^n)^{-1}$. We will elaborate on these properties later when we prove conservation.

3.4. Grid evolution

At this point, we have transferred state from particle to grid, and we are ready to apply forces and perform our grid-based evolution. We must update grid velocity $\tilde{\mathbf{v}}_i^{n+1}$, position $\tilde{\mathbf{x}}_i^{n+1}$, and deformation gradient $\mathbf{F}_p^{n+1}$. The update of grid positions to $\tilde{\mathbf{x}}_i^{n+1}$ is purely conceptual. Our implementation uses fixed Cartesian grids.

An important aspect of allowing for exact conservation of linear and angular momentum during particle/grid transfers is that conservation of the entire method can be achieved by combining with one of the many conservative integrators used for updating the grid state [28, 29, 30, 31, 32, 33]. We introduce a parameter λ, which allows us to consider an entire family of methods that conserve linear and angular momentum. This family contains two notable members: symplectic Euler ($\lambda = 0$) and midpoint rule ($\lambda = \frac{1}{2}$). The schemes $\lambda = 0$ and $\lambda = 1$ are both explicit; the rest are implicit. We use midpoint rule for all of our examples. Note that schemes such as forward Euler, backward Euler, and trapezoid rule do not conserve angular momentum and thus are not suitable for our purposes. Our family of grid-based updates is

$$\mathbf{F}_p^{n+1} = \left(\mathbf{I} + \sum_i (\tilde{\mathbf{x}}_i^{n+1} - \mathbf{x}_i^n)(\nabla w_{ip}^n)^T\right)\mathbf{F}_p^n \tag{27}$$

$$\mathbf{F}_p^{n+\lambda} = (1-\lambda)\mathbf{F}_p^n + \lambda\mathbf{F}_p^{n+1} \tag{28}$$

$$\tilde{\mathbf{v}}_i^{n+1} = \mathbf{v}_i^n + \frac{\Delta t}{m_i^n}\mathbf{f}_i^{n+\lambda} \tag{29}$$

$$\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + \Delta t(\lambda\mathbf{v}_i^n + (1-\lambda)\tilde{\mathbf{v}}_i^{n+1}) \tag{30}$$

The velocity update rule uses forces $\mathbf{f}_i^{n+\lambda}$, which we define from a potential energy function $\Phi^{n+\lambda}$, which we compute from an energy density $\Psi_p(\mathbf{F}_p)$. Our rules for computing potential energy $\Phi^{n+\lambda}$, force $\mathbf{f}_i^{n+\lambda}$, and

product by force derivatives are

$$\Psi_p^{n+\lambda} = \Psi_p\big(\mathbf{F}_p^{n+\lambda}\big) \tag{31}$$

$$\Phi^{n+\lambda} = \sum_p V_p \Psi_p^{n+\lambda} \tag{32}$$

$$\mathbf{P}_p^{n+\lambda} = \mathbf{P}_p\big(\mathbf{F}_p^{n+\lambda}\big) \tag{33}$$

$$\mathbf{f}_i^{n+\lambda} = \sum_p V_p \mathbf{P}_p^{n+\lambda} (\mathbf{F}_p^n)^T \nabla w_{ip}^n \tag{34}$$

$$\mathbf{A}_p = \frac{\partial \mathbf{P}_p}{\partial \mathbf{F}_p} : \left(\sum_i \Delta \mathbf{v}_i (\nabla w_{ip}^n)^T \mathbf{F}_p^n \right) \tag{35}$$

$$\sum_j \left(\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j} \right) \Delta \mathbf{v}_j = \sum_p V_p \mathbf{A}_p (\mathbf{F}_p^n)^T \nabla w_{ip}^n \tag{36}$$

Here, $\mathbf{P}_p$ is the first Piola-Kirchhoff stress tensor. Defining forces through an energy ensures angular momentum conservation; the particular constitutive model does not matter.

3.5. Transfer to particles

With grid evolution completed, we have updated grid locations $\tilde{\mathbf{x}}_i^{n+1}$ and velocities $\tilde{\mathbf{v}}_i^{n+1}$. What remains is to transfer this information back to particles. We do this using the transfers

$$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} \tag{37}$$

$$\mathbf{B}_p^{n+1} = \frac{1}{2} \sum_i w_{ip}^n \big(\tilde{\mathbf{v}}_i^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n + \tilde{\mathbf{x}}_i^{n+1} - \mathbf{x}_p^{n+1})^T + (\mathbf{x}_i^n - \mathbf{x}_p^n - \tilde{\mathbf{x}}_i^{n+1} + \mathbf{x}_p^{n+1}) (\tilde{\mathbf{v}}_i^{n+1})^T \big) \tag{38}$$

$$\mathbf{x}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1} \tag{39}$$

$$\mathbf{F}_p^{n+1} = \left(\mathbf{I} + \sum_i (\tilde{\mathbf{x}}_i^{n+1} - \mathbf{x}_i^n)(\nabla w_{ip}^n)^T \right) \mathbf{F}_p^n \tag{40}$$

This completes the specification of our angular-momentum-conserving family of APIC schemes.

4. Implementation details

4.1. Implicit midpoint as minimization problem

The grid update is in general implicit, including the midpoint rule ($\lambda = \frac{1}{2}$). Since this is the member that we implemented and recommend using, we restrict our attention here to this case. We also demonstrate symplectic Euler and backward Euler as grid update schemes for comparison in some of our numerical experiments. Symplectic Euler is explicit and does not require the optimization treatment that follows. Backward Euler is not a member of the family described in this paper; we compare against it for reference.

We solve the resulting nonlinear systems of equations following an optimization-stabilized Newton-Raphson solver framework [34, 35]. The implicit midpoint scheme for MPM grid nodes is

$$\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + \Delta t\left(\frac{\mathbf{v}_i^n + \tilde{\mathbf{v}}_i^{n+1}}{2}\right),$$
$$\tilde{\mathbf{v}}_i^{n+1} = \mathbf{v}_i^n + \frac{\Delta t}{m_i^n}\mathbf{f}_i\left(\frac{\mathbf{x}_i^n + \tilde{\mathbf{x}}_i^{n+1}}{2}\right).$$

Eliminating $\tilde{\mathbf{x}}_i^{n+1}$ gives

$$\begin{aligned} m_i^n \frac{\tilde{\mathbf{v}}_i^{n+1} - \mathbf{v}_i^n}{\Delta t} &= \mathbf{f}_i\left(\frac{\mathbf{x}_i^n + \mathbf{x}_i^n + \Delta t\left(\frac{\mathbf{v}_i^n + \tilde{\mathbf{v}}_i^{n+1}}{2}\right)}{2}\right) \\ &= \mathbf{f}_i\left(\mathbf{x}_i^n + \frac{\Delta t}{4}(\mathbf{v}_i^n + \tilde{\mathbf{v}}_i^{n+1})\right). \end{aligned}$$

Changing to the variable $\Delta\mathbf{v}_i = \tilde{\mathbf{v}}_i^{n+1} - \mathbf{v}_i^n$,

$$m_i^n \Delta\mathbf{v}_i = \Delta t\mathbf{f}_i\left(\mathbf{x}_i^n + \frac{\Delta t}{2}\mathbf{v}_i^n + \frac{\Delta t}{4}\Delta\mathbf{v}_i\right). \tag{41}$$

The corresponding minimization objective function is

$$E(\Delta\mathbf{v}_i) = \sum_i \frac{m_i^n}{8}\left\|\Delta\mathbf{v}_i\right\|^2 + \Phi\left(\mathbf{x}_i^n + \frac{\Delta t}{2}\mathbf{v}_i^n + \frac{\Delta t}{4}\Delta\mathbf{v}_i\right).$$

This is similar to the corresponding objective for backward Euler, which is

$$E_{be}(\Delta\mathbf{v}_i) = \sum_i \frac{m_i^n}{2}\left\|\Delta\mathbf{v}_i\right\|^2 + \Phi\big(\mathbf{x}_i^n + \Delta t\mathbf{v}_i^n + \Delta t^2\Delta\mathbf{v}_i\big).$$

The minimum of E occurs when

$$\begin{aligned} \mathbf{g}_i &= \frac{\partial E}{\partial \Delta\mathbf{v}_i} \\ &= \frac{\partial}{\partial \Delta\mathbf{v}_i}\left(\sum_j \frac{m_j}{8}\left\|\Delta\mathbf{v}_j\right\|^2 + \Phi\left(\mathbf{x}_j^n + \frac{\Delta t}{2}\mathbf{v}_j^n + \frac{\Delta t}{4}\Delta\mathbf{v}_j\right)\right) \\ &= \frac{m_i^n}{4}\Delta\mathbf{v}_i - \frac{\Delta t}{4}\mathbf{f}_i\left(\mathbf{x}_i^n + \frac{\Delta t}{2}\mathbf{v}_i^n + \frac{\Delta t}{4}\Delta\mathbf{v}_i\right), \end{aligned}$$

Note that $\mathbf{g}(\Delta\mathbf{v}_i) = \mathbf{0}$ is just (41), so minimizing E is equivalent to solving (41). Multiplying the derivative of $\mathbf{g}$ by some vector $\delta\mathbf{u}_i$ will be necessary.

$$\begin{aligned} \sum_j \frac{\partial \mathbf{g}_i}{\partial \Delta\mathbf{v}_j}\delta\mathbf{u}_j &= \sum_j \frac{\partial}{\partial \Delta\mathbf{v}_j}\left(\frac{m_i^n}{4}\Delta\mathbf{v}_i - \frac{\Delta t}{4}\mathbf{f}_i\left(\mathbf{x}_i^n + \frac{\Delta t}{2}\mathbf{v}_i^n + \frac{\Delta t}{4}\Delta\mathbf{v}_i\right)\right)\delta\mathbf{u}_j \\ &= \frac{m_i^n}{4}\delta\mathbf{u}_i - \frac{\Delta t^2}{16}\sum_j \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j}\left(\mathbf{x}_i^n + \frac{\Delta t}{2}\mathbf{v}_i^n + \frac{\Delta t}{4}\Delta\mathbf{v}_i\right)\delta\mathbf{u}_j \end{aligned}$$

This in turn requires a matrix-vector multiply by the force derivative, which is done using (35) and (36).

4.2. Momentum conservation on incomplete convergence

The conservation properties of our method (see Section 5.2) depend on solving (41) to convergence. If this is not done, conservation will be only approximate. We note, however, that this is not a fundamental problem. One way to track down the source of the problem is to label every vector a *velocity-like* or *force-like*. Assume initial velocity is zero and all forces are momentum-conserving. Then, we can note some rules about how these types of vector should behave:

1. A force-like vector will sum to zero.
2. A velocity-like vector will sum to zero when scaled by mass.
3. Scaling a velocity-like vector by mass produces a force-like vector.
4. Scaling a force-like vector by inverse mass produces a velocity-like vector.
5. Scaling a vector by a constant preserves its type.
6. Adding vectors is only permitted if they have the same type; the type is preserved.
7. In the matrix-vector multiply $\delta\mathbf{f}_i = \sum_j \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j}\delta\mathbf{u}_j$, $\delta\mathbf{u}_j$ must be velocity-like, and $\delta\mathbf{f}_i$ will be force-like.
8. Dot product is only allowed if one vector is force-like and the other is velocity-like. (This is done, for example, when computing kinetic energy.)

As long as these rules are followed, the velocity will be velocity-like, which implies conservation of linear momentum (the last rule is not strictly required, but we can enforce it anyway). Propagating these labels through the algorithm (Newton's method, line searches, conjugate gradient, etc.) is straightforward and breaks down only inside the conjugate gradient solver. The source of the problem is that $\mathbf{p}$, $\mathbf{r}$, and $\mathbf{s}$ must be of the same type (see Algorithm 1), so that $\mathbf{s} \leftarrow \mathbf{A}\mathbf{p}$ means the operator $\mathbf{A}$ must take and produce the same type of vector. The system we are solving takes the general form

$$\mathbf{A}_1\delta\mathbf{v} = \delta\mathbf{f} \qquad \mathbf{A}_1 = \mathbf{M} + \zeta\frac{\partial \mathbf{f}}{\partial \mathbf{x}},$$

where $\mathbf{M}$ is a diagonal mass matrix, ζ is a scalar, $\delta\mathbf{v}$ is a velocity-like vector, $\delta\mathbf{f}$ is a force-like vector. The operator $\mathbf{A}_1$ takes velocity-like vectors and produces force-like vectors, which is a problem. We can avoid that problem by rewriting

$$\mathbf{A}_2\delta\mathbf{v} = \mathbf{M}^{-1}\delta\mathbf{f} \qquad \mathbf{A}_1 = \mathbf{I} + \mathbf{M}^{-1}\zeta\frac{\partial \mathbf{f}}{\partial \mathbf{x}}.$$

Now, $\mathbf{A}_2$ takes velocity-like vectors and returns velocity-like vectors. Unfortunately, this $\mathbf{A}_2$ is not symmetric.

The conjugate gradient operates on vectors in only a few ways: matrix-vector multiply, vector operations, and inner product. Note that the inner product used does not need to be the standard inner product: $\mathbf{A}$ is only required to be symmetric with respect to the inner product chosen. That is, $\langle \mathbf{A}\mathbf{u}, \mathbf{v}\rangle = \langle \mathbf{u}, \mathbf{A}\mathbf{v}\rangle$ for any $\mathbf{u}$ and $\mathbf{v}$. Note that $\mathbf{A}_2$ is symmetric with respect to the mass inner product $\langle \mathbf{u}, \mathbf{v}\rangle = \mathbf{u}^T\mathbf{M}\mathbf{v}$. Using this

Algorithm 1 Conjugate Gradient

procedure CONJUGATE–GRADIENT($\mathbf{A}$, $\mathbf{x}$, $\mathbf{b}$)
  $\mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}$
  $\mathbf{p} \leftarrow \mathbf{r}$
  $\gamma \leftarrow \langle \mathbf{r}, \mathbf{r} \rangle$
  while not converged **do**
    $\mathbf{s} \leftarrow \mathbf{A}\mathbf{p}$
    $\alpha \leftarrow \dfrac{\gamma}{\langle \mathbf{p}, \mathbf{s} \rangle}$
    $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
    $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{s}$ ▷ $\mathbf{r}$ and $\mathbf{s}$ have the same type
    $\kappa \leftarrow \langle \mathbf{r}, \mathbf{r} \rangle$
    $\beta \leftarrow \frac{\kappa}{\gamma}$
    $\mathbf{p} \leftarrow \mathbf{r} + \beta\mathbf{p}$ ▷ $\mathbf{r}$ and $\mathbf{p}$ have the same type
    $\gamma \leftarrow \kappa$
  end while
end procedure

modified system and a mass inner product for conjugate gradient is a perfectly acceptable means of solving the linear system. Furthermore, all vectors in the conjugate gradient algorithm are now velocity-like, which allows us to label all of our vectors. This in turn guarantees conservation of momentum, even if our solver is not fully converged.

4.3. CFL condition

We choose our time step size Δt so that no particle will travel more than the grid spacing Δx in one time step. We approximate this by assuming that these particles travel with the initial grid velocity $\mathbf{v}_i^n$. While this does not take into account the potentially dramatic affect of forces, we note that our method is implicit

and can tolerate such errors.

$$
\begin{aligned}
m_i^n \mathbf{v}_i^n &= \sum_p w_{ip}^n m_p (\mathbf{v}_p^n + \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n)) \\
m_i^n \|\mathbf{v}_i^n\| &= \left\| \sum_p w_{ip}^n m_p (\mathbf{v}_p^n + \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n)) \right\| \\
&\leq \left\| \sum_p w_{ip}^n m_p \mathbf{v}_p^n \right\| + \left\| \sum_p w_{ip}^n m_p \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n) \right\| \\
&\leq \sum_p w_{ip}^n m_p \|\mathbf{v}_p^n\| + \sum_p w_{ip}^n m_p \|\mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n)\| \\
&\leq \sum_p w_{ip}^n m_p \|\mathbf{v}_p^n\| + \sum_p w_{ip}^n m_p \|\mathbf{B}_p^n\|_F \|(\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n)\|
\end{aligned}
$$

Interpolation stencil support is bounded by $\|\mathbf{x}_i^n - \mathbf{x}_p^n\| \leq \kappa \Delta x$. If we also assume $D_p = k\mathbf{I}$, then $\|(\mathbf{D}_p^n)^{-1}(\mathbf{x}_i^n - \mathbf{x}_p^n)\| \leq \frac{\kappa}{k} \Delta x$.

$$
\begin{aligned}
m_i^n \|\mathbf{v}_i^n\| &\leq \sum_p w_{ip}^n m_p \|\mathbf{v}_p^n\| + \sum_p w_{ip}^n m_p \|\mathbf{B}_p^n\|_F \|(\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n)\| \\
&\leq \sum_p w_{ip}^n m_p \left(\|\mathbf{v}_p^n\| + \frac{\kappa}{k} \Delta x \|\mathbf{B}_p^n\|_F \right) \\
&\leq \left(\sum_p w_{ip}^n m_p \right) \max_p \left(\|\mathbf{v}_p^n\| + \frac{\kappa}{k} \Delta x \|\mathbf{B}_p^n\|_F \right) \\
&= m_i^n \max_p \left(\|\mathbf{v}_p^n\| + \frac{\kappa}{k} \Delta x \|\mathbf{B}_p^n\|_F \right) \\
\|\mathbf{v}_i^n\| &\leq \max_p \left(\|\mathbf{v}_p^n\| + \frac{\kappa}{k} \Delta x \|\mathbf{B}_p^n\|_F \right)
\end{aligned}
$$

In the case of both quadratic and cubic interpolation, $\frac{\kappa}{k} \Delta x = \frac{6\sqrt{d}}{\Delta x}$, where d is the dimension. A reasonable CFL condition is then

$$
\Delta t \leq \frac{\nu \Delta x}{\max_p \left(\|\mathbf{v}_p^n\| + \frac{\kappa}{k} \Delta x \|\mathbf{B}_p^n\|_F \right)}.
$$

We use $\nu = 1$ for our examples.

5. Notes and analysis

Here we discuss a number of aspects and useful properties of the schemes we have proposed.

5.1. RPIC transfer properties

The RPIC transfers outlined in Sections 2.1.1 from particle momenta to grid momenta are

$$
\boldsymbol{\omega}_p^n = \mathbf{K}_p^{-1} \mathbf{l}_p^n \tag{42}
$$

$$
m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p (\mathbf{v}_p^n + \boldsymbol{\omega}_p^n \times (\mathbf{x}_i^n - \mathbf{x}_p^n)). \tag{43}
$$

The transfers from updated grid velocities $\tilde{\mathbf{v}}_i^{n+1}$ to new particle velocities $\mathbf{v}_p^{n+1}$ and angular momenta $\mathbf{l}_p^{n+1}$ as outlined in Section 2.1.2 are

$$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} \tag{44}$$

$$\mathbf{l}_p^{n+1} = \sum_i (\mathbf{x}_i^n - \mathbf{x}_p^n) \times m_p w_{ip}^n \tilde{\mathbf{v}}_i^{n+1}. \tag{45}$$

These transfers conserve total linear and angular momenta. To define the total linear and angular momenta of the particles, we can think of them as a collection of rigid bodies, each made up of individual point masses $m_{ip} = m_p w_{ip}^n$ located at the grid nodes $\mathbf{x}_i^n$. Then the total momenta of the collection of rigid bodies is the sum of the contributions from each respective point mass. That is, the total linear momentum $\mathbf{p}^P$ of the particles is

$$\mathbf{p}^P = \sum_p \sum_i m_{ip} \big(\mathbf{v}_p^n + \boldsymbol{\omega}_p^n \times (\mathbf{x}_i^n - \mathbf{x}_p^n)\big) = \sum_p m_p \mathbf{v}_p^n. \tag{46}$$

That is, $\boldsymbol{\omega}_p^n$ contributes no net linear momentum. The total angular momentum (about the origin) is

$$\mathbf{L}^P = \sum_p \sum_i \mathbf{x}_i^n \times m_{ip} \big(\mathbf{v}_p^n + \boldsymbol{\omega}_p^n \times (\mathbf{x}_i^n - \mathbf{x}_p^n)\big) = \sum_p \mathbf{x}_p^n \times m_p \mathbf{v}_p^n + \sum_p \mathbf{l}_p^n. \tag{47}$$

The equivalence of the momentum definitions follows from

$$\mathbf{p}^P = \sum_p \sum_i m_{ip} \big(\mathbf{v}_p^n + \boldsymbol{\omega}_p^n \times (\mathbf{x}_i^n - \mathbf{x}_p^n)\big) \tag{48}$$

$$= \sum_p \sum_i m_p w_{ip}^n \big(\mathbf{v}_p^n + \boldsymbol{\omega}_p^n \times (\mathbf{x}_i^n - \mathbf{x}_p^n)\big) \tag{49}$$

$$= \sum_p m_p \left(\mathbf{v}_p^n \sum_i w_{ip}^n + \boldsymbol{\omega}_p^n \times \sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n) \right) \tag{50}$$

$$= \sum_p m_p \mathbf{v}_p^n. \tag{51}$$

The equivalence of the angular momentum definitions follows from

$$\mathbf{L}^P = \sum_p \sum_i \mathbf{x}_i^n \times m_{ip}\big(\mathbf{v}_p^n + \boldsymbol{\omega}_p^n \times (\mathbf{x}_i^n - \mathbf{x}_p^n)\big) \tag{52}$$

$$= \sum_p \sum_i w_{ip}^n \mathbf{x}_i^n \times m_p\big(\mathbf{v}_p^n + \boldsymbol{\omega}_p^n \times (\mathbf{x}_i^n - \mathbf{x}_p^n)\big) \tag{53}$$

$$= \sum_p \sum_i w_{ip}^n \mathbf{x}_i^n \times m_p \mathbf{v}_p^n + \sum_p \sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n) \times m_p(\boldsymbol{\omega}_p^n \times (\mathbf{x}_i^n - \mathbf{x}_p^n)) \tag{54}$$

$$+ \sum_p \sum_i w_{ip}^n \mathbf{x}_p^n \times m_p(\boldsymbol{\omega}_p^n \times (\mathbf{x}_i^n - \mathbf{x}_p^n)) \tag{55}$$

$$= \sum_p \mathbf{x}_p^n \times m_p \mathbf{v}_p^n + \sum_p \left(m_p \sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n)^* (\mathbf{x}_i^n - \mathbf{x}_p^n)^{*T} \right) \boldsymbol{\omega}_p^n \tag{56}$$

$$+ \sum_p \mathbf{x}_p^n \times m_p \left(\boldsymbol{\omega}_p^n \times \sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n) \right) \tag{57}$$

$$= \sum_p \mathbf{x}_p^n \times m_p \mathbf{v}_p^n + \sum_p \mathbf{K}_p^n \boldsymbol{\omega}_p^n \tag{58}$$

$$= \sum_p \mathbf{x}_p^n \times m_p \mathbf{v}_p^n + \sum_p \mathbf{l}_p^n. \tag{59}$$

Recall we use $\mathbf{x}^*$ to denote the matrix that expresses $\mathbf{x} \times \mathbf{y} = \mathbf{x}^* \mathbf{y}$. Note that both $\mathbf{p}^P$ and $\mathbf{L}^P$ can be defined entirely in terms of particle state. These quantities are defined in a more obvious manner on the grid as

$$\mathbf{p}^G = \sum_i m_i^n \mathbf{v}_i^n \quad \text{and} \quad \mathbf{L}^G = \sum_i \mathbf{x}_i^n \times m_i^n \mathbf{v}_i^n. \tag{60}$$

In the following, we will show that after the transfer from particle to grid, $\mathbf{p}^P = \mathbf{p}^G$ and $\mathbf{L}^P = \mathbf{L}^G$ and after the transfer from grid to particle $\tilde{\mathbf{p}}^P = \tilde{\mathbf{p}}^G$ and $\tilde{\mathbf{L}}^P = \tilde{\mathbf{L}}^G$

5.1.1. Particle to grid: conservation of linear momentum

The total linear momenta are equal after the transfer, which can be seen simply from

$$\begin{aligned} \mathbf{p}^P &= \sum_p \sum_i m_p w_{ip}^n \big(\mathbf{v}_p^n + \boldsymbol{\omega}_p^n \times (\mathbf{x}_i^n - \mathbf{x}_p^n)\big) \\ &= \sum_i \sum_p m_p w_{ip}^n \big(\mathbf{v}_p^n + \boldsymbol{\omega}_p^n \times (\mathbf{x}_i^n - \mathbf{x}_p^n)\big) = \sum_i m_i^n \mathbf{v}_i^n = \mathbf{p}^G. \end{aligned} \tag{61}$$

5.1.2. Particle to grid: conservation of angular momentum

The transfer also conserves total angular momentum since

$$\mathbf{L}^P = \sum_p \sum_i \mathbf{x}_i \times m_p w_{ip}^n \big(\mathbf{v}_p + \mathbf{C}_p(\mathbf{x}_i^n - \mathbf{x}_p^n)\big) \tag{62}$$

$$= \sum_i \mathbf{x}_i \times \sum_p m_p w_{ip}^n \big(\mathbf{v}_p + \mathbf{C}_p(\mathbf{x}_i^n - \mathbf{x}_p^n)\big) = \sum_i \mathbf{x}_i \times m_i^n \mathbf{v}_i = \mathbf{L}^G. \tag{63}$$

5.1.3. Grid to particle: conservation of linear momentum

The transfers in Equation 44 conserve linear momentum since

$$\begin{aligned}\tilde{\mathbf{p}}^P &= \sum_p m_p \mathbf{v}_p^{n+1} = \sum_p m_p \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} \\ &= \sum_i \sum_p m_p w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} = \sum_i m_i^n \tilde{\mathbf{v}}_i^{n+1} = \tilde{\mathbf{p}}^G\end{aligned} \tag{64}$$

where we use the particle state expression for $\tilde{\mathbf{p}}^P$.

5.1.4. Grid to particle: conservation of angular momentum

The transfers in Equation 44 also conserve angular momentum. We can show this using Equation 44 to express the new total particle angular momentum as

$$\tilde{\mathbf{L}}^P = \sum_p \mathbf{x}_p^{n+1} \times m_p \mathbf{v}_p^{n+1} + \sum_p \mathbf{l}_p^{n+1} \tag{65}$$

$$= \sum_p \mathbf{x}_p^{n+1} \times m_p \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} + \sum_p \sum_i (\mathbf{x}_i^n - \mathbf{x}_p^n) \times m_p w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} \tag{66}$$

$$= \sum_i \mathbf{x}_i \times \sum_p m_p w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} = \sum_i \mathbf{x}_i \times m_i^n \tilde{\mathbf{v}}_i^{n+1} = \tilde{\mathbf{L}}^G. \tag{67}$$

5.2. APIC conservation of linear momentum

The APIC scheme is naturally divided into three steps; we show that each step independently conserves linear momentum. The first step is the transfer of information from particle to grid. We see that the initial particle momentum $\mathbf{p}^{P,n}$ is equal to the grid momentum after the transfer $\mathbf{p}^{G,n}$.

$$\begin{aligned}\mathbf{p}^{G,n} &= \sum_i m_i^n \mathbf{v}_i^n \\ &= \sum_i \sum_p w_{ip}^n m_p (\mathbf{v}_p^n + \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n)) \\ &= \sum_i \sum_p w_{ip}^n m_p \mathbf{v}_p^n + \sum_i \sum_p w_{ip}^n m_p \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n) \\ &= \sum_p m_p \mathbf{v}_p^n \sum_i w_{ip}^n + \sum_p m_p \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} \sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n) \\ &= \sum_p m_p \mathbf{v}_p^n \\ &= \mathbf{p}^{P,n}\end{aligned}$$

Once mass and momentum are on the grid, grid positions and velocities are updated. We note that initial grid momentum matches the final grid momentum $\tilde{\mathbf{p}}^{G,n+1}$.

$$\begin{aligned}
\tilde{\mathbf{p}}^{G,n+1} &= \sum_i m_i^n \tilde{\mathbf{v}}_i^{n+1} \\
&= \sum_i m_i^n \left(\mathbf{v}_i^n + \frac{\Delta t}{m_i^n} \mathbf{f}_i^{n+\lambda} \right) \\
&= \sum_i m_i^n \mathbf{v}_i^n + \Delta t \sum_i \mathbf{f}_i^{n+\lambda} \\
&= \mathbf{p}^G + \Delta t \sum_i \sum_p V_p \mathbf{P}_p^{n+\lambda} (\mathbf{F}_p^n)^T \nabla w_{ip}^n \\
&= \mathbf{p}^G + \Delta t \sum_p V_p \mathbf{P}_p^{n+\lambda} (\mathbf{F}_p^n)^T \sum_i \nabla w_{ip}^n \\
&= \mathbf{p}^{G,n}
\end{aligned}$$

The final step is transferring information back to particles. This step is also conservative since

$$\begin{aligned}
\mathbf{p}^{P,n+1} &= \sum_p m_p \mathbf{v}_p^{n+1} \\
&= \sum_p m_p \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} \\
&= \sum_i \tilde{\mathbf{v}}_i^{n+1} \sum_p m_p w_{ip}^n \\
&= \sum_i m_i^n \tilde{\mathbf{v}}_i^{n+1} \\
&= \tilde{\mathbf{p}}^{G,n+1}
\end{aligned}$$

Finally, the entire scheme conserves momentum since $\mathbf{p}^{P,n+1} = \mathbf{p}^{P,n}$.

5.3. APIC conservation of angular momentum

We use the permutation tensor in this section. To make these portions easier to read, we take the convention that $\mathbf{A} : \boldsymbol{\epsilon}$ denotes $A_{\alpha\beta}\epsilon_{\alpha\beta\gamma}$. The manipulation $\mathbf{u} \times \mathbf{v} = (\mathbf{v}\mathbf{u}^T)^T : \boldsymbol{\epsilon}$ is used to transition from a cross product into the permutation tensor.

5.3.1. Transfer to grid

Our approach to demonstrating angular momentum conservation follows the same three steps. In this case, we show that $\mathbf{L}^{P,n} = \mathbf{L}^{G,n} = \tilde{\mathbf{L}}^{G,n+1} = \mathbf{L}^{P,n+1}$, though the individual steps are more involved. We

begin with the transfer from particles to the grid.

$$\begin{aligned}
\mathbf{L}^{G,n} &= \sum_i \mathbf{x}_i^n \times m_i^n \mathbf{v}_i^n \\
&= \sum_p \sum_i \mathbf{x}_i^n \times m_p w_{ip}^n (\mathbf{v}_p^n + \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n)) \\
&= \sum_p \sum_i \mathbf{x}_i^n \times m_p w_{ip}^n \mathbf{v}_p^n + \sum_p \mathbf{l}_p^{\mathbf{B}} \\
&= \sum_p \mathbf{x}_p^n \times m_p \mathbf{v}_p^n + \sum_p m_p (\mathbf{B}_p^n)^T : \boldsymbol{\epsilon} \\
&= \mathbf{L}^{P,n}
\end{aligned}$$

where use has been made from

$$\begin{aligned}
\mathbf{l}_p^{\mathbf{B}} &= \sum_i \mathbf{x}_i^n \times m_p w_{ip}^n \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n) \\
&= \sum_i \left(m_p w_{ip}^n \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n) (\mathbf{x}_i^n)^T \right)^T : \boldsymbol{\epsilon} \\
&= \left(m_p \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} \sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n) (\mathbf{x}_i^n)^T \right)^T : \boldsymbol{\epsilon} \\
&= \left(m_p \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} \left(\sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n) (\mathbf{x}_i^n - \mathbf{x}_p^n)^T + \sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n) (\mathbf{x}_p^n)^T \right) \right)^T : \boldsymbol{\epsilon} \\
&= \left(m_p \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{D}_p^n + \mathbf{0}) \right)^T : \boldsymbol{\epsilon} \\
&= m_p (\mathbf{B}_p^n)^T : \boldsymbol{\epsilon}
\end{aligned}$$

Note that this expression for $\mathbf{L}^{P,n}$ can be taken to be the definition of total angular momentum on particles, with $\mathbf{l}_p^{\mathbf{B}}$ being the angular momentum contribution of particle p due to $\mathbf{B}_p$.

5.3.2. Grid update

The next step is the grid update. Let $\mathbf{G}_p = \sum_i \tilde{\mathbf{x}}_i^{n+1} (\nabla w_{ip}^n)^T$. Then,

$$\begin{aligned}
\mathbf{F}_p^{n+1} &= \left(\mathbf{I} + \sum_i (\tilde{\mathbf{x}}_i^{n+1} - \mathbf{x}_i^n) (\nabla w_{ip}^n)^T \right) \mathbf{F}_p^n \\
&= (\mathbf{I} + \mathbf{G}_p - \mathbf{I}) \mathbf{F}_p^n \\
&= \mathbf{G}_p \mathbf{F}_p^n
\end{aligned}$$

For the grid update portion, we will use the following manipulations to replace cross products with permutation tensors.

$$\begin{aligned}
\sum_i \mathbf{x}_i^n \times \mathbf{A}_p \nabla w_{ip}^n &= \sum_i \left(\mathbf{A}_p \nabla w_{ip}^n (\mathbf{x}_i^n)^T\right)^T : \boldsymbol{\epsilon} \\
&= \left(\mathbf{A}_p \sum_i \nabla w_{ip}^n (\mathbf{x}_i^n)^T\right)^T : \boldsymbol{\epsilon} \\
&= (\mathbf{A}_p \mathbf{I})^T : \boldsymbol{\epsilon} \\
&= \mathbf{A}_p^T : \boldsymbol{\epsilon} \\
\sum_i \tilde{\mathbf{x}}_i^{n+1} \times \mathbf{A}_p \nabla w_{ip}^n &= \sum_i \left(\mathbf{A}_p \nabla w_{ip}^n (\tilde{\mathbf{x}}_i^{n+1})^T\right)^T : \boldsymbol{\epsilon} \\
&= \left(\mathbf{A}_p \sum_i \nabla w_{ip}^n (\tilde{\mathbf{x}}_i^{n+1})^T\right)^T : \boldsymbol{\epsilon} \\
&= \left(\mathbf{A}_p \mathbf{G}_p^T\right)^T : \boldsymbol{\epsilon}
\end{aligned}$$

With this, we note the identity

$$\begin{aligned}
\sum_i (\lambda \tilde{\mathbf{x}}_i^{n+1} + (1-\lambda)\mathbf{x}_i^n) \times \mathbf{f}_i^{n+\lambda} &= \sum_i (\lambda \tilde{\mathbf{x}}_i^{n+1} + (1-\lambda)\mathbf{x}_i^n) \times \sum_p V_p \mathbf{P}_p^{n+\lambda} (\mathbf{F}_p^n)^T \nabla w_{ip}^n \\
&= \sum_i (\lambda \tilde{\mathbf{x}}_i^{n+1} + (1-\lambda)\mathbf{x}_i^n) \times \sum_p V_p \mathbf{F}_p^{n+\lambda} \mathbf{S}_p^{n+\lambda} (\mathbf{F}_p^n)^T \nabla w_{ip}^n \\
&= \sum_i \sum_p V_p \left(\mathbf{F}_p^{n+\lambda} \mathbf{S}_p^{n+\lambda} (\mathbf{F}_p^n)^T ((1-\lambda)\mathbf{I} + \lambda \mathbf{G}_p)^T\right)^T : \boldsymbol{\epsilon} \\
&= \sum_i \sum_p V_p \left(\mathbf{F}_p^{n+\lambda} \mathbf{S}_p^{n+\lambda} ((1-\lambda)\mathbf{F}_p^n + \lambda \mathbf{G}_p \mathbf{F}_p^n)^T\right)^T : \boldsymbol{\epsilon} \\
&= \sum_i \sum_p V_p \left(\mathbf{F}_p^{n+\lambda} \mathbf{S}_p^{n+\lambda} ((1-\lambda)\mathbf{F}_p^n + \lambda \mathbf{F}_p^{n+1})^T\right)^T : \boldsymbol{\epsilon} \\
&= \sum_i \sum_p V_p \left(\mathbf{F}_p^{n+\lambda} \mathbf{S}_p^{n+\lambda} (\mathbf{F}_p^{n+\lambda})^T\right)^T : \boldsymbol{\epsilon} \\
&= \mathbf{0}
\end{aligned}$$

from which it follows that

$$\sum_i (\lambda \tilde{\mathbf{x}}_i^{n+1} + (1-\lambda)\mathbf{x}_i^n) \times m_i^n (\tilde{\mathbf{v}}_i^{n+1} - \mathbf{v}_i^n) = \mathbf{0}.$$

With this identity, it is finally possible to show that angular momentum is conserved across the grid update.

$$\begin{aligned}
\tilde{\mathbf{L}}^{G,n+1} - \mathbf{L}^{G,n} &= \sum_i \tilde{\mathbf{x}}_i^{n+1} \times m_i^n \tilde{\mathbf{v}}_i^{n+1} - \sum_i \mathbf{x}_i^n \times m_i^n \mathbf{v}_i^n \\
&= \sum_i \tilde{\mathbf{x}}_i^{n+1} \times m_i^n \tilde{\mathbf{v}}_i^{n+1} - \sum_i \mathbf{x}_i^n \times m_i^n \mathbf{v}_i^n - \sum_i (\lambda \tilde{\mathbf{x}}_i^{n+1} + (1-\lambda)\mathbf{x}_i^n) \times m_i^n (\tilde{\mathbf{v}}_i^{n+1} - \mathbf{v}_i^n) \\
&= \sum_i (\tilde{\mathbf{x}}_i^{n+1} - \mathbf{x}_i^n) \times m_i^n ((1-\lambda)\tilde{\mathbf{v}}_i^{n+1} + \lambda \mathbf{v}_i^n) \\
&= \sum_i \Delta t ((1-\lambda)\tilde{\mathbf{v}}_i^{n+1} + \lambda \mathbf{v}_i^n) \times m_i^n ((1-\lambda)\tilde{\mathbf{v}}_i^{n+1} + \lambda \mathbf{v}_i^n) \\
&= \mathbf{0}
\end{aligned}$$

5.3.3. Transfer to particles

Using

$$\begin{aligned}
\mathbf{l}_p^{\mathbf{B}} &= m_p (\mathbf{B}_p^{n+1})^T : \boldsymbol{\epsilon} \\
&= m_p \left(\frac{1}{2} \sum_i w_{ip}^n \big(\tilde{\mathbf{v}}_i^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n + \tilde{\mathbf{x}}_i^{n+1} - \mathbf{x}_p^{n+1})^T + (\mathbf{x}_i^n - \mathbf{x}_p^n - \tilde{\mathbf{x}}_i^{n+1} + \mathbf{x}_p^{n+1})(\tilde{\mathbf{v}}_i^{n+1})^T \big) \right)^T : \boldsymbol{\epsilon} \\
&= \frac{m_p}{2} \sum_i w_{ip}^n \big((\mathbf{x}_i^n - \mathbf{x}_p^n + \tilde{\mathbf{x}}_i^{n+1} - \mathbf{x}_p^{n+1}) \times \tilde{\mathbf{v}}_i^{n+1} + \tilde{\mathbf{v}}_i^{n+1} \times (\mathbf{x}_i^n - \mathbf{x}_p^n - \tilde{\mathbf{x}}_i^{n+1} + \mathbf{x}_p^{n+1}) \big) \\
&= \frac{m_p}{2} \sum_i w_{ip}^n \big((\tilde{\mathbf{x}}_i^{n+1} - \mathbf{x}_p^{n+1}) \times \tilde{\mathbf{v}}_i^{n+1} + \tilde{\mathbf{v}}_i^{n+1} \times (-\tilde{\mathbf{x}}_i^{n+1} + \mathbf{x}_p^{n+1}) \big) \\
&= m_p \sum_i w_{ip}^n (\tilde{\mathbf{x}}_i^{n+1} - \mathbf{x}_p^{n+1}) \times \tilde{\mathbf{v}}_i^{n+1} \\
&= m_p \sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1} \times \tilde{\mathbf{v}}_i^{n+1} - m_p \sum_i w_{ip}^n \mathbf{x}_p^{n+1} \times \tilde{\mathbf{v}}_i^{n+1} \\
&= m_p \sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1} \times \tilde{\mathbf{v}}_i^{n+1} - \mathbf{x}_p^{n+1} \times m_p \mathbf{v}_p^{n+1}
\end{aligned}$$

we have

$$\begin{aligned}
\mathbf{L}^{P,n+1} &= \sum_p \mathbf{x}_p^{n+1} \times m_p \mathbf{v}_p^{n+1} + \sum_p m_p (\mathbf{B}_p^{n+1})^T : \boldsymbol{\epsilon} \\
&= \sum_p \mathbf{x}_p^{n+1} \times m_p \mathbf{v}_p^{n+1} + \sum_p \left(m_p \sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1} \times \tilde{\mathbf{v}}_i^{n+1} - \mathbf{x}_p^{n+1} \times m_p \mathbf{v}_p^{n+1} \right) \\
&= \sum_p m_p \sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1} \times \tilde{\mathbf{v}}_i^{n+1} \\
&= \sum_i \tilde{\mathbf{x}}_i^{n+1} \times \tilde{\mathbf{v}}_i^{n+1} \sum_p w_{ip}^n m_p \\
&= \sum_i \tilde{\mathbf{x}}_i^{n+1} \times m_i^n \tilde{\mathbf{v}}_i^{n+1} \\
&= \tilde{\mathbf{L}}^{G,n+1}
\end{aligned}$$

This completes the proof of angular momentum conservation.

5.4. Stability

It is possible to construct a transfer that conserves angular momentum and retains affine fields but is unstable. This instability was observed to occur when variations in the transfer are considered. The instability conveniently manifests when a particle is isolated, so the problem is easy to avoid. We require that an isolated particle experiencing no forces should translate uniformly with no change in $\mathbf{v}_p^n$ or $\mathbf{B}_p^n$. We now show that our scheme has this property.

Consider that there is only one particle, which experiences no forces ($\mathbf{f}_i^{n+\lambda} = \mathbf{0}$). Then, the update rules for $\mathbf{v}_i^n$, $\tilde{\mathbf{v}}_i^{n+1}$, and $\tilde{\mathbf{x}}_i^{n+1}$ reduce to

$$\mathbf{v}_i^n = \mathbf{v}_p^n + \mathbf{B}_p^n(\mathbf{D}_p^n)^{-1}(\mathbf{x}_i^n - \mathbf{x}_p^n) \tag{68}$$

$$\tilde{\mathbf{v}}_i^{n+1} = \mathbf{v}_i^n \tag{69}$$

$$\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + \Delta t \mathbf{v}_i^n \tag{70}$$

With these, the final particle velocity is

$$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} \tag{71}$$

$$= \sum_i w_{ip}^n (\mathbf{v}_p^n + \mathbf{B}_p^n(\mathbf{D}_p^n)^{-1}(\mathbf{x}_i^n - \mathbf{x}_p^n)) \tag{72}$$

$$= \mathbf{v}_p^n \sum_i w_{ip}^n + \mathbf{B}_p^n(\mathbf{D}_p^n)^{-1} \sum_i w_{ip}^n(\mathbf{x}_i^n - \mathbf{x}_p^n) \tag{73}$$

$$= \mathbf{v}_p^n \tag{74}$$

The final position is

$$\mathbf{x}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1} \tag{75}$$

$$= \sum_i w_{ip}^n (\mathbf{x}_i^n + \Delta t \mathbf{v}_i^n) \tag{76}$$

$$= \sum_i w_{ip}^n \mathbf{x}_i^n + \Delta t \sum_i w_{ip}^n \mathbf{v}_i^n \tag{77}$$

$$= \mathbf{x}_p^n + \Delta t \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} \tag{78}$$

$$= \mathbf{x}_p^n + \Delta t \mathbf{v}_p^{n+1} \tag{79}$$

$$= \mathbf{x}_p^n + \Delta t \mathbf{v}_p^n \tag{80}$$

Finally, $\mathbf{B}_p^{n+1}$ is now

$$\mathbf{B}_p^{n+1} = \frac{1}{2}\sum_i w_{ip}^n \big(\tilde{\mathbf{v}}_i^{n+1}(\mathbf{x}_i^n - \mathbf{x}_p^n + \tilde{\mathbf{x}}_i^{n+1} - \mathbf{x}_p^{n+1})^T + (\mathbf{x}_i^n - \mathbf{x}_p^n - \tilde{\mathbf{x}}_i^{n+1} + \mathbf{x}_p^{n+1})(\tilde{\mathbf{v}}_i^{n+1})^T\big) \tag{81}$$

$$= \frac{1}{2}\sum_i w_{ip}^n \big(\mathbf{v}_i^n(2\mathbf{x}_i^n - 2\mathbf{x}_p^n + \Delta t\mathbf{v}_i^n - \Delta t\mathbf{v}_p^n)^T + (\Delta t\mathbf{v}_p^n - \Delta t\mathbf{v}_i^n)(\mathbf{v}_i^n)^T\big) \tag{82}$$

$$= \frac{1}{2}\sum_i w_{ip}^n \big(\mathbf{v}_i^n(2\mathbf{x}_i^n - 2\mathbf{x}_p^n - \Delta t\mathbf{v}_p^n)^T + \Delta t\mathbf{v}_p^n(\mathbf{v}_i^n)^T\big) \tag{83}$$

$$= \sum_i w_{ip}^n \mathbf{v}_i^n(\mathbf{x}_i^n - \mathbf{x}_p^n)^T - \Delta t\left(\sum_i w_{ip}^n \mathbf{v}_i^n\right)(\mathbf{v}_p^n)^T + \Delta t\mathbf{v}_p^n\left(\sum_i w_{ip}^n \mathbf{v}_i^n\right)^T \tag{84}$$

$$= \sum_i w_{ip}^n \mathbf{v}_i^n(\mathbf{x}_i^n - \mathbf{x}_p^n)^T - \Delta t\mathbf{v}_p^n(\mathbf{v}_p^n)^T + \Delta t\mathbf{v}_p^n(\mathbf{v}_p^n)^T \tag{85}$$

$$= \sum_i w_{ip}^n \mathbf{v}_i^n(\mathbf{x}_i^n - \mathbf{x}_p^n)^T \tag{86}$$

$$= \sum_i w_{ip}^n (\mathbf{v}_p^n + \mathbf{B}_p^n(\mathbf{D}_p^n)^{-1}(\mathbf{x}_i^n - \mathbf{x}_p^n))(\mathbf{x}_i^n - \mathbf{x}_p^n)^T \tag{87}$$

$$= \mathbf{B}_p^n(\mathbf{D}_p^n)^{-1}\sum_i w_{ip}^n(\mathbf{x}_i^n - \mathbf{x}_p^n)(\mathbf{x}_i^n - \mathbf{x}_p^n)^T + \mathbf{v}_p^n\sum_i w_{ip}^n(\mathbf{x}_i^n - \mathbf{x}_p^n)^T \tag{88}$$

$$= \mathbf{B}_p^n(\mathbf{D}_p^n)^{-1}\mathbf{D}_p^n \tag{89}$$

$$= \mathbf{B}_p^n \tag{90}$$

This guarantees stability in the case of one particle. In practice, the scheme is observed to be stable with any number of particles when using a quadratic or cubic basis. It is not, however, stable for a multilinear basis, as noted in Section 5.9.

5.5. Affine round trip

One of the original motivations behind the original APIC scheme is that, in some reasonable sense, it should preserve affine velocity fields. Particles represent an affine velocity field when $\mathbf{v}_p^n = \mathbf{v} + \mathbf{C}\mathbf{x}_p$ and $\mathbf{B}_p^n = \mathbf{C}\mathbf{D}_p^n$ for some vector $\mathbf{v}$ and matrix $\mathbf{C}$. We require that such a velocity field be preserved in the limit when an arbitrarily small time step is taken, so that we may assume $\Delta t = 0$. The assumption $\Delta t = 0$ immediately implies $\tilde{\mathbf{v}}_i^{n+1} = \mathbf{v}_i^n$ and $\tilde{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n$, from which $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n$, $w_{ip}^{n+1} = w_{ip}^n$, and $\mathbf{D}_p^{n+1} = \mathbf{D}_p^n$

follow. The transfer to the grid simplifies to

$$m_i^n = \sum_p m_p w_{ip}^n \tag{91}$$

$$m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p (\mathbf{v}_p^n + \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n)) \tag{92}$$

$$= \sum_p w_{ip}^n m_p (\mathbf{v} + \mathbf{C}\mathbf{x}_p + \mathbf{C}\mathbf{D}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n)) \tag{93}$$

$$= \sum_p w_{ip}^n m_p (\mathbf{v} + \mathbf{C}\mathbf{x}_i^n) \tag{94}$$

$$= m_i^n (\mathbf{v} + \mathbf{C}\mathbf{x}_i^n) \tag{95}$$

$$\mathbf{v}_i^n = \mathbf{v} + \mathbf{C}\mathbf{x}_i^n \tag{96}$$

so that the grid velocity field is produced by the same affine velocity field.

$$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} \tag{97}$$

$$= \sum_i w_{ip}^n \mathbf{v}_i^n \tag{98}$$

$$= \sum_i w_{ip}^n (\mathbf{v} + \mathbf{C}\mathbf{x}_i^n) \tag{99}$$

$$= \mathbf{v} \sum_i w_{ip}^n + \mathbf{C} \sum_i w_{ip}^n \mathbf{x}_i^n \tag{100}$$

$$= \mathbf{v} + \mathbf{C}\mathbf{x}_p^n \tag{101}$$

$$= \mathbf{v} + \mathbf{C}\mathbf{x}_p^{n+1} \tag{102}$$

$$\mathbf{B}_p^{n+1} = \frac{1}{2} \sum_i w_{ip}^n \big(\tilde{\mathbf{v}}_i^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n + \tilde{\mathbf{x}}_i^{n+1} - \mathbf{x}_p^{n+1})^T + (\mathbf{x}_i^n - \mathbf{x}_p^n - \tilde{\mathbf{x}}_i^{n+1} + \mathbf{x}_p^{n+1}) (\tilde{\mathbf{v}}_i^{n+1})^T \big) \tag{103}$$

$$= \sum_i w_{ip}^n \mathbf{v}_i^n (\mathbf{x}_i^n - \mathbf{x}_p^n)^T \tag{104}$$

$$= \sum_i w_{ip}^n (\mathbf{v} + \mathbf{C}\mathbf{x}_i^n) (\mathbf{x}_i^n - \mathbf{x}_p^n)^T \tag{105}$$

$$= \mathbf{v} \sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n)^T + \mathbf{C} \sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n)(\mathbf{x}_i^n - \mathbf{x}_p^n)^T + \mathbf{C}\mathbf{x}_p^n \sum_i w_{ip}^n (\mathbf{x}_i^n - \mathbf{x}_p^n)^T \tag{106}$$

$$= \mathbf{C}\mathbf{D}_p^n \tag{107}$$

$$= \mathbf{C}\mathbf{D}_p^{n+1} \tag{108}$$

The new particle state corresponds to the same affine velocity field, so the field has been preserved across the transfers.

286 *5.6. Unifying PIC, RPIC and APIC*

For each of PIC, RPIC and APIC the transfer from particle to grid can be written as $m_{ip} = m_p N(\mathbf{x}_p - \mathbf{x}_i)$, $m_i = \sum_p m_{ip}$, $(m\mathbf{v})_{ip} = m_{ip}(\mathbf{v}_p + \mathbf{C}_p(\mathbf{x}_i - \mathbf{x}_p))$ and $(m\mathbf{v})_i = \sum_p (m\mathbf{v})_{ip}$, where the $\mathbf{C}_p$ is zero, skew or a full matrix to distinguish PIC, RPIC and APIC respectively. However, when designing the transfer back from grid to particle, the details are less obviously related. There is, in fact, a description that unifies RPIC, APIC and PIC. It starts with the alternative notation

$$\mathbf{v}_{ip} = \sum_{j=1}^{N_r} s_{jp} \mathbf{b}_{jpi}$$

to describe the velocity field local to the particle. Here, the $\mathbf{b}_{jpi} \in \mathbb{R}^3$, $\mathbf{b}_{jp} \in \mathbb{R}^{3N_g}$ with N_g equal to the number of grid nodes and

$$\mathbf{b}_{jp} = \begin{pmatrix} \mathbf{b}_{jp1} \\ \mathbf{b}_{jp2} \\ \vdots \\ \mathbf{b}_{jpN_G} \end{pmatrix}.$$

The $\mathbf{b}_{jp}$ form a reduced basis for the grid velocity field $\mathbf{v}_{ip}$ local to particle p. That is, the $\mathbf{b}_{jp} \in \mathbb{R}^{3N_g}$ are individual modes defined over the grid and the s_{jp} describe the local particle state, e.g. they are equivalent to $\mathbf{v}_p$ and $\mathbf{C}_P$ for APIC. The choice of the basis vectors $\mathbf{b}_{jp} \in \mathbb{R}^{3N_g}$ is what distinguishes PIC from RPIC from APIC etc. For example, PIC uses $N_r = 3$ and

$$\mathbf{b}_{jp} = \begin{pmatrix} \mathbf{e}_j \\ \mathbf{e}_j \\ \vdots \\ \mathbf{e}_j \end{pmatrix} \quad \text{and} \quad \mathbf{v}_p = \begin{pmatrix} s_{1p} \\ s_{2p} \\ s_{3p} \end{pmatrix}$$

for $j = 1, 2, 3 = N_r$ with $\mathbf{e}_j \in \mathbb{R}^3$ the j^{th} standard basis vector for $\mathbb{R}^3$. RPIC uses $N_r = 6$ with the same $\mathbf{b}_{jp}$ as PIC for $j = 1, 2, 3$ and

$$\mathbf{b}_{jp} = \begin{pmatrix} \sum_{k=1}^3 \epsilon_{j-3k1} r_{1pk} \\ \sum_{k=1}^3 \epsilon_{j-3k2} r_{1pk} \\ \sum_{k=1}^3 \epsilon_{j-3k3} r_{1pk} \\ \sum_{k=1}^3 \epsilon_{j-3k1} r_{2pk} \\ \vdots \\ \sum_{k=1}^3 \epsilon_{j-3k1} r_{N_g pk} \\ \sum_{k=1}^3 \epsilon_{j-3k2} r_{N_g pk} \\ \sum_{k=1}^3 \epsilon_{j-3k3} r_{N_g pk} \end{pmatrix}$$

for $j = 4, 5, 6 = N_r$ where ϵ_{ijk} is the permutation tensor (such that the k^{th} component of $\mathbf{a}\times\mathbf{b}$ is $\sum_{i,j} \epsilon_{ijk} a_i b_j$) and r_{ipk} is the k^{th} component of $\mathbf{r}_{ip} = \mathbf{x}_i - \mathbf{x}_p$. With this convention,

$$\mathbf{v}_p = \begin{pmatrix} s_{1p} \\ s_{2p} \\ s_{3p} \end{pmatrix} \quad \text{and} \quad \boldsymbol{\omega}_p = \begin{pmatrix} s_{4p} \\ s_{5p} \\ s_{6p} \end{pmatrix}.$$

Lastly, APIC uses the same $\mathbf{b}_{jp}$ as RPIC for $j = 1, 2, \ldots, 6$ and

$$\mathbf{b}_{jp} = \begin{pmatrix} \sum_{k=1}^{3} |\epsilon_{j-6k1}| r_{1pk} \\ \sum_{k=1}^{3} |\epsilon_{j-6k2}| r_{1pk} \\ \sum_{k=1}^{3} |\epsilon_{j-6k3}| r_{1pk} \\ \sum_{k=1}^{3} |\epsilon_{j-6k1}| r_{2pk} \\ \vdots \\ \sum_{k=1}^{3} |\epsilon_{j-6k1}| r_{N_g pk} \\ \sum_{k=1}^{3} |\epsilon_{j-6k2}| r_{N_g pk} \\ \sum_{k=1}^{3} |\epsilon_{j-6k3}| r_{N_g pk} \end{pmatrix}$$

for $j = 7, 8, 9$ (which represent symmetric matrices with zero diagonal) and

$$\mathbf{b}_{jp} = \begin{pmatrix} \mathbf{e}_{j-9}\mathbf{e}_{j-9}^T \mathbf{r}_{1p} \\ \mathbf{e}_{j-9}\mathbf{e}_{j-9}^T \mathbf{r}_{2p} \\ \vdots \\ \mathbf{e}_{j-9}\mathbf{e}_{j-9}^T \mathbf{r}_{N_G p} \end{pmatrix}$$

for $j = 10, 11, 12 = N_r$ which represent the diagonal matrices. With this convection,

$$\mathbf{v}_p = \begin{pmatrix} s_{1p} \\ s_{2p} \\ s_{3p} \end{pmatrix} \quad \text{and} \quad \mathbf{C}_p = \begin{pmatrix} s_{10p} & s_{7p} - s_{4p} & s_{8p} - s_{5p} \\ s_{4p} + s_{7p} & s_{11p} & s_{9p} - s_{6p} \\ s_{5p} + s_{8p} & s_{6p} + s_{9p} & s_{12p} \end{pmatrix}.$$

5.6.1. Transfer from particle to grid

With this notation, the transfer from particle to grid is $m_{ip} = m_p N(\mathbf{x}_p - \mathbf{x}_i)$, $m_i = \sum_p m_{ip}$, $(m\mathbf{v})_{ip} = m_{ip} \sum_{j=1}^{N_r} s_{jp} \mathbf{b}_{jpi}$ and $(m\mathbf{v})_i = \sum_p (m\mathbf{v})_{ip}$. That is, the grid momenta are just the sum of the momenta modes local to each particle. Notably, this describes the PIC, RPIC and APIC transfers in one description. If we define the total linear momentum of the particle state to be the sum of the total linear momenta from each local particle state, and the total angular momentum of particle state to be the sum of the total angular momenta from each local particle state (computed about the particle) plus the angular momenta of the particles, then the transfer conserves linear and angular momentum by reasoning analogous to that in Section 5.1.

5.6.2. Transfer from grid to particle

Using this notation, the transfer from grid to particle is done by determining $\tilde{s}_{jp}$ from the updated grid velocities $\tilde{\mathbf{v}}_i$. We can do this in a way that conserves linear, angular momenta, as well as generalized moments directly by solving the system

$$\mathbf{b}_{jp}^T \begin{pmatrix} m_{1p}\tilde{\mathbf{v}}_1 \\ m_{2p}\tilde{\mathbf{v}}_2 \\ \vdots \\ m_{N_g p}\tilde{\mathbf{v}}_{N_g} \end{pmatrix} = \mathbf{b}_{jp}^T \mathbf{M} \mathbf{b}_{ip} \tilde{s}_{ip}$$

for $\tilde{s}_{ip}$. Notably, this describes the PIC and RPIC transfers when the $N_r = 3$ and $N_r = 6$ respectively. Furthermore, it generalizes the result to the affine case. Remarkably, it can be shown that the matrix $\mathbf{b}_{jp}^T \mathbf{M} \mathbf{b}_{ip}$ is both diagonal and constant in time for quadratic and cubic B-splines, i.e. does not depend on the configuration of the particles relative to the grid. This not only means that these solves can be done efficiently, but it also shows that for APIC, $\tilde{s}_{ip}$ are the PIC modes $i = 1, 2, 3$ and the RPIC modes for $i = 4, 5, 6$ and that the remaining APIC modes are determined independently since the components are decoupled in the solve. Lastly, the transfer conserves linear and angular momentum by the same argument as for RPIC since the right hand side terms

$$\mathbf{b}_{jp}^T \begin{pmatrix} m_{1p}\tilde{\mathbf{v}}_1 \\ m_{2p}\tilde{\mathbf{v}}_2 \\ \vdots \\ m_{N_g p}\tilde{\mathbf{v}}_{N_g} \end{pmatrix} = \mathbf{b}_{jp}^T \mathbf{M} \mathbf{b}_{ip} \tilde{s}_{ip}$$

are the linear momentum components for $j = 1, 2, 3$ and the angular momentum components (computed about the particle) for $j = 4, 5, 6$.

5.6.3. Coefficient computations

To construct the transfer, we need to compute the basis coefficients $\tilde{s}_{ip}$. This requires building the matrix $\mathbf{b}_{jp}^T \mathbf{M} \mathbf{b}_{ip}$ and inverting it. Notably, building this matrix and its inverse requires very little computation. In the case of cubic and quadratic B-spline interpolation, the matrix is actually constant and diagonal. For cubic B-splines, $\mathbf{b}_{jp}^T \mathbf{M} \mathbf{b}_{ip}$ is diagonal with entries

$$m_p \times \{1, 1, 1, \frac{2}{3}\Delta x^2, \frac{2}{3}\Delta x^2, \frac{2}{3}\Delta x^2, \frac{2}{3}\Delta x^2, \frac{2}{3}\Delta x^2, \frac{2}{3}\Delta x^2, \frac{1}{3}\Delta x^2, \frac{1}{3}\Delta x^2, \frac{1}{3}\Delta x^2\}$$

For quadratic B-splines, $\mathbf{b}_{jp}^T \mathbf{M} \mathbf{b}_{ip}$ is diagonal with entries

$$m_p \times \{1, 1, 1, \frac{1}{2}\Delta x^2, \frac{1}{2}\Delta x^2, \frac{1}{2}\Delta x^2, \frac{1}{2}\Delta x^2, \frac{1}{2}\Delta x^2, \frac{1}{2}\Delta x^2, \frac{1}{4}\Delta x^2, \frac{1}{4}\Delta x^2, \frac{1}{4}\Delta x^2\}.$$

For the cubic and quadratic cases, this transfer is equivalent to those derived in previous sections, albeit without the intuition needed to prove a number of the useful properties.

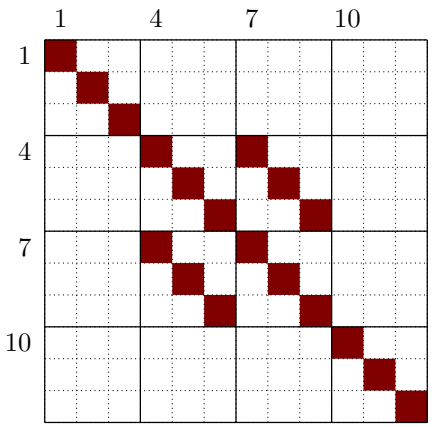

Figure 2: The matrix structure of $\mathbf{b}_{jp}^T \mathbf{M} \mathbf{b}_{ip}$ in the multilinear case. See section 5.6.3 for more detailed information on all entries.It is symmetric as with quadratic/cubic B-spline interpolations, but it is not diagonal. The inversion of this matrix has the same sparsity structure.

For multilinear interpolation function, it is a symmetric matrix (but not diagonal).

$$\mathbf{b}_{jp}^T \mathbf{M} \mathbf{b}_{ip} = m_p \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_0 & \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\Delta x \mathbf{Z} - \mathbf{Z}^2 \end{pmatrix}$$

where

$$\mathbf{A}_0 = -\text{diag} \begin{pmatrix} z_{p2}^2 + z_{p3}^2 + \Delta x(z_{p2} + z_{p3}) \\ z_{p1}^2 + z_{p3}^2 + \Delta x(z_{p1} + z_{p3}) \\ z_{p1}^2 + z_{p2}^2 + \Delta x(z_{p1} + z_{p2}) \end{pmatrix} \qquad \mathbf{A}_1 = -\text{diag} \begin{pmatrix} (\Delta x + z_{p2} + z_{p3})(z_{p2} - z_{p3}) \\ (\Delta x + z_{p1} + z_{p3})(z_{p3} - z_{p1}) \\ (\Delta x + z_{p1} + z_{p2})(z_{p1} - z_{p2}) \end{pmatrix},$$

$\mathbf{O}_p$ is the the bottom left corner location of the cell that particle p affects, $\mathbf{z}_p = \mathbf{O}_p - \mathbf{x}_p$ and $\mathbf{Z} = \text{diag}(\mathbf{z})$. Figure 2 shows the structure of this matrix. Its inverse has the same structure.

5.7. Degrades to backward Euler case

The transfers are the same as in [22], except for the update rule for $\mathbf{B}_p^{n+1}$, which we show below simplifies into the transfer from [22] with $\lambda = 0$. Using λ recovers the the backward Euler grid update rule $\tilde{\mathbf{x}}_i^{n+1} =$

$\mathbf{x}_i^n + \Delta t \tilde{\mathbf{v}}_i^{n+1}$. Then,

$$
\begin{aligned}
\tilde{\mathbf{x}}_i^{n+1} &= \mathbf{x}_i^n + \Delta t \tilde{\mathbf{v}}_i^{n+1} \\
\sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1} &= \sum_i w_{ip}^n (\mathbf{x}_i^n + \Delta t \tilde{\mathbf{v}}_i^{n+1}) \\
\mathbf{x}_p^{n+1} &= \mathbf{x}_p^n + \Delta t \mathbf{v}_p^{n+1} \\
\mathbf{B}_p^{n+1} &= \frac{1}{2} \sum_i w_{ip}^n \big(\tilde{\mathbf{v}}_i^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n + \tilde{\mathbf{x}}_i^{n+1} - \mathbf{x}_p^{n+1})^T + (\mathbf{x}_i^n - \mathbf{x}_p^n - \tilde{\mathbf{x}}_i^{n+1} + \mathbf{x}_p^{n+1}) (\tilde{\mathbf{v}}_i^{n+1})^T \big) \\
&= \frac{1}{2} \sum_i w_{ip}^n \big(\tilde{\mathbf{v}}_i^{n+1} (2\mathbf{x}_i^n - 2\mathbf{x}_p^n + \Delta t \tilde{\mathbf{v}}_i^{n+1} - \Delta t \mathbf{v}_p^{n+1})^T + \Delta t (\mathbf{v}_p^{n+1} - \tilde{\mathbf{v}}_i^{n+1}) (\tilde{\mathbf{v}}_i^{n+1})^T \big) \\
&= \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n)^T + \frac{\Delta t}{2} \sum_i w_{ip}^n \big(-\tilde{\mathbf{v}}_i^{n+1} (\mathbf{v}_p^{n+1})^T + \mathbf{v}_p^{n+1} (\tilde{\mathbf{v}}_i^{n+1})^T \big) \\
&= \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n)^T + \frac{\Delta t}{2} \big(-\mathbf{v}_p^{n+1} (\mathbf{v}_p^{n+1})^T + \mathbf{v}_p^{n+1} (\mathbf{v}_p^{n+1})^T \big) \\
&= \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n)^T
\end{aligned}
$$

This was the original APIC transfer.

Another departure from [22] is the update rule for $\mathbf{x}_p^{n+1}$. We note, however, that these are also equivalent in the $\lambda = 0$ case.

$$
\begin{aligned}
\mathbf{x}_p^{n+1} &= \sum_i w_{ip}^n \tilde{\mathbf{x}}_i^{n+1} \\
&= \sum_i w_{ip}^n (\mathbf{x}_i^n + \Delta t \lambda \mathbf{v}_i^n + \Delta t (1-\lambda) \tilde{\mathbf{v}}_i^{n+1}) \\
&= \mathbf{x}_p^n + \Delta t \lambda \sum_i w_{ip}^n \mathbf{v}_i^n + \Delta t (1-\lambda) \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} \\
&= \mathbf{x}_p^n + \Delta t \lambda \sum_i w_{ip}^n \mathbf{v}_i^n + \Delta t (1-\lambda) \mathbf{v}_p^{n+1}
\end{aligned}
$$

Note that the second term is not $\mathbf{v}_p^n$. This term vanishes in the special case $\lambda = 0$, so that the transfer in [22] could be done using the particle velocity. We see that the proposed method represents a generalization of the original APIC scheme.

5.8. Note on transfer construction

The most important difference between the method described above and the original APIC method from [22] is (38). This transfer was constructed by first assuming that the transfer should take the form

$$
\mathbf{B}_p^{n+1} = \sum_i w_{ip}^n \big(\tilde{\mathbf{v}}_i^{n+1} (a\mathbf{x}_i^n + b\mathbf{x}_p^n + c\tilde{\mathbf{x}}_i^{n+1} + d\mathbf{x}_p^{n+1})^T + (e\mathbf{x}_i^n + f\mathbf{x}_p^n + g\tilde{\mathbf{x}}_i^{n+1} + h\mathbf{x}_p^{n+1}) (\tilde{\mathbf{v}}_i^{n+1})^T \big).
$$

Terms involving $\mathbf{v}_i^n$ could also be considered; such terms would add a FLIP-like character to the transfer. Terms similar to $(\tilde{\mathbf{v}}_i^{n+1})^T \mathbf{x}_i^n \mathbf{I}$ are also technically possible; we do not include them since we did not find

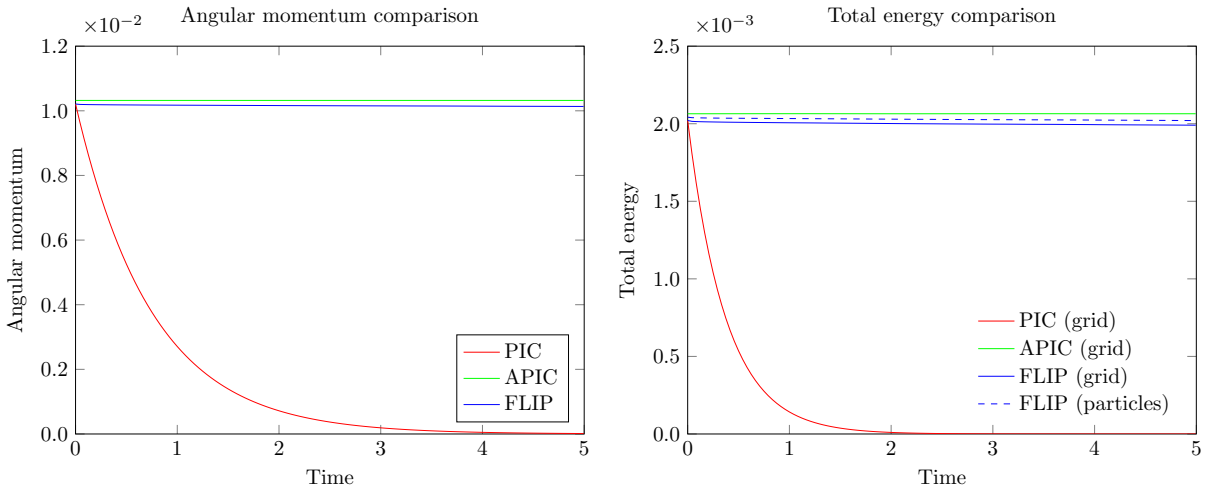

Figure 3: Angular momentum and total energy evolution over time for a rotating elastic disk with initial angular momentum (Section 6.1). APIC has a slightly higher initial angular momentum due to the additional $\mathbf{B}_p$ matrix that holds some extra contribution that is lacking for PIC or FLIP.

them to contribute meaningfully to the transfer. We restrict ourselves here to the form above, which leaves us to choose the eight coefficients. We require (1) angular momentum conservation (See Section 5.3), (2) affine round trip (See Section 5.5), and (3) one particle stability (See Section 5.4). These three constraints uniquely determine all eight coefficients in the general case. In the special case $\lambda = 0$, the eight terms are not linearly independent; this allows additional freedom to eliminate terms, resulting in the simpler transfer from [22].

5.9. Stability concerns for multilinear interpolation

With linear interpolation weights, $\mathbf{D}_p^n$ is not invertible. The particle to grid transfer from

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

can in the multilinear interpolation case be re-written as

$$m_i^n \mathbf{v}_i^n = \sum_p (w_{ip}^n m_p \mathbf{v}_p^n + m_p \mathbf{B}_p^n \nabla w_{ip}^n)$$

using $w_{ip}^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n) = \nabla w_{ip}^n$. To see why this can cause problems, consider the case with one particle. Let $\mathbf{v}_p^n = \mathbf{0}$ and $\mathbf{B}_p^n = \mathbf{I}$. Then

$$m_i^n \mathbf{v}_i^n = m_p \nabla w_{ip}^n$$
$$\mathbf{v}_i^n = \frac{\nabla w_{ip}^n}{w_{ip}^n}$$

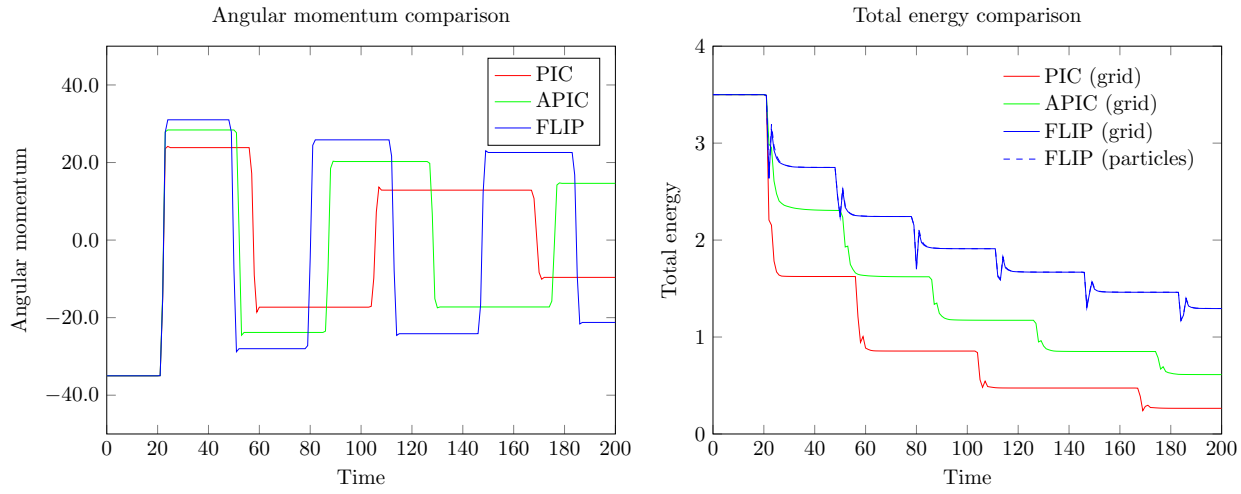

Figure 4: Angular momentum and total energy evolution over time for an elastic cylinder rebounding between two walls (Section 6.2).

Consider a grid cell at $[0, \Delta x] \times [0, \Delta x]$ with grid degrees of freedom at $\mathbf{x}_{(i,j)} = (i\Delta x, j\Delta x)$. If the particle p is at $(\epsilon\Delta x, \frac{1}{2}\Delta x)$, then $w^n_{(1,1)p} = \frac{\epsilon}{2}$ and $\nabla w^n_{(1,1)p} = \langle \frac{1}{2\Delta x}, \frac{\epsilon}{\Delta x} \rangle$. But then, $\mathbf{v}^n_{(1,1)} = \langle \frac{1}{\epsilon\Delta x}, \frac{2}{\Delta x} \rangle$, which is unbounded. This in turn results in a kinetic energy contribution of $\frac{1}{2} m^n_{(1,1)} \|\mathbf{v}^n_{(1,1)}\|^2 = \frac{1}{4\Delta x} m_p(\epsilon^{-1} + 4\epsilon)$. Since ϵ can be arbitrarily small, the energy of the grid node can be arbitrarily large. This unbounded growth in energy causes instability and makes a multilinear interpolation kernel unsuitable for this APIC formulation.

In order to use APIC using multilinear interpolation function without being unstable, we can lag the affine matrix with the transfers being

$$m^n_i \mathbf{v}^n_i = \sum_p w^n_{ip} m_p (\mathbf{v}^n_p + \mathbf{C}^n_p (\mathbf{x}^n_i - \mathbf{x}^n_p)),$$
$$\mathbf{C}^{n+1}_p = \sum_i w^n_{ip} \tilde{\mathbf{v}}_i ((\mathbf{D}^n_p)^{-1} (\mathbf{x}^n_i - \mathbf{x}^n_p))^T.$$

For multilinear interpolation, it further simplifies to $\mathbf{C}^{n+1}_p = \sum_i \tilde{\mathbf{v}}_i (\nabla w^n_{ip})^T$. Note that this formulation does not suffer from the same energy increasing problem as long as $\mathbf{C}$ is bounded. The difference is effectively that the $\mathbf{B}$ formulation inverts $\mathbf{D}^n_p$ at the end of the time step rather than doing so at the beginning of the next time step. In the quadratic and cubic cases, the $\mathbf{C}$ formulation and the $\mathbf{B}$ formulation are equivalent, since $\mathbf{D}^n_p$ is a constant scalar multiple of the identity and thus $\mathbf{D}^n_p = \mathbf{D}^{n+1}_p$. For multilinear interpolation, we always use the lagged version for stability.

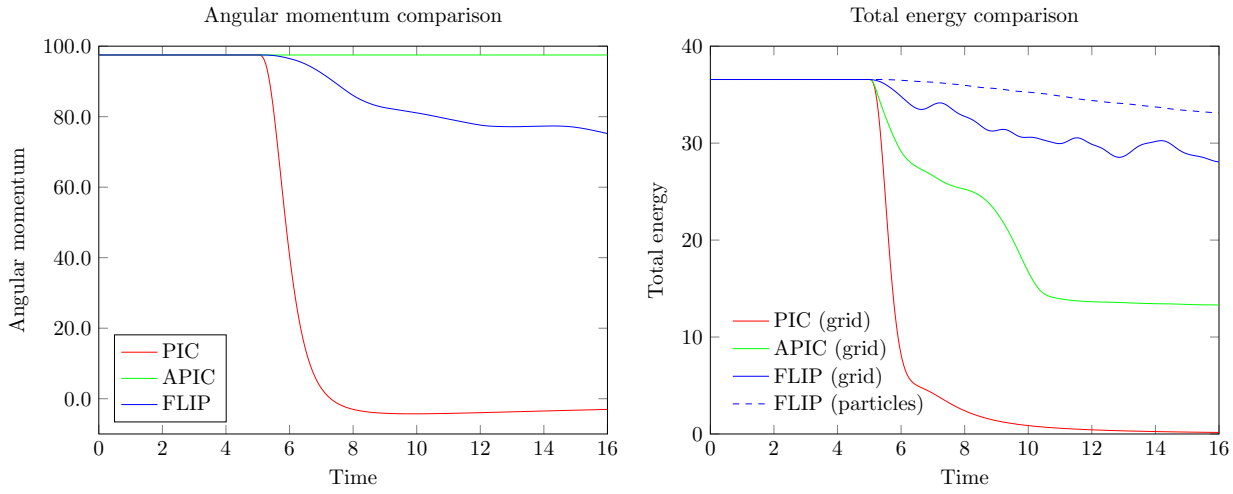

Figure 5: Angular momentum and total energy evolution over time for a skew impact of two elastic cylinders (Section 6.3).

6. Numerical simulations

6.1. Rotating elastic cylinder

We begin our tests by running a simple rotation test. We use a $[0,1] \times [0,1]$ domain with 32×32 resolution. We initialize a circle with radius 0.3 centered at $(0.5, 0.5)$, seeded with four particles per cell. The circle begins rotating with angular velocity 0.4 about its center. We use an initial density $\rho = 2$ and a Neo-Hookean constitutive model with $E = 1000$ and $\nu = 0.3$. See Figure 3. Note that the initial energy and angular momentum is not the same for the different schemes. The energy and angular momentum for APIC is slightly higher (by around 1%) since the initial state for this scheme includes $\mathbf{B}_p^n$, which is nonzero for a spinning object. This additional state contributes to energy and angular momentum, making it slightly higher. Energy is printed for FLIP as measured on particles and on the grid; the grid version is consistently slightly less since energy is filtered during the transfer to the grid.

6.2. Rebound of an elastic cylinder

We run the same example as in section 4.1 of [7].

The grid spacing is $h = 0.5$. Slip boundary conditions are applied at $x = 0$ and $x = 15$. The cylinder is initially centered at $(2.5, 2.5)$ and has radius 1.5. MPM particles are sampled with alignment to the grid with spacing 0.25 (so 4 particles per cell for a full cell). Material density is 4. The constitutive model is Neo-Hookean with Young's Modulus 85.5 and Poisson's ratio 0.425. The initial velocity of the cylinder is $(0.5, 0)$. See Figure 4.

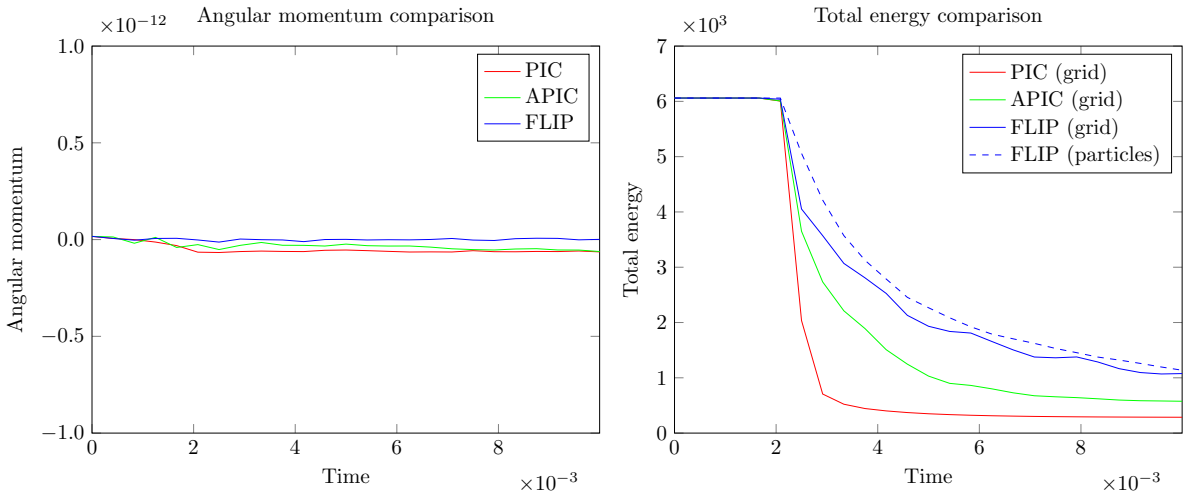

Figure 6: Angular momentum and total energy evolution over time for two colliding hollow cylinders (Section 6.4).

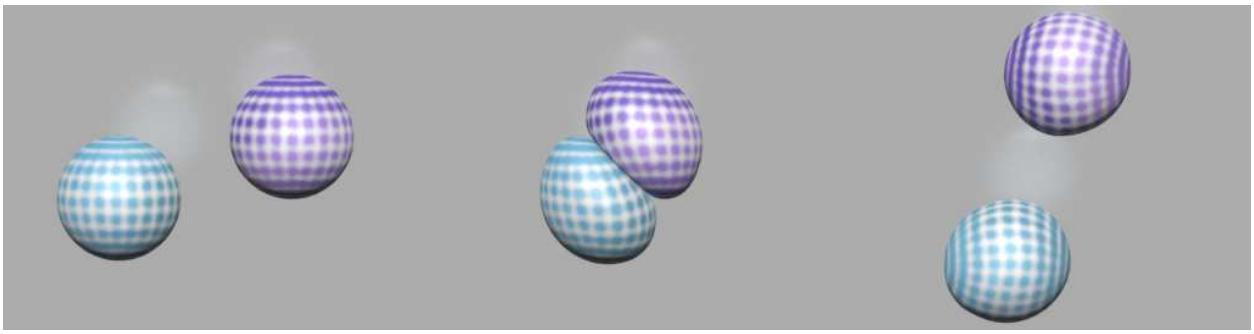

Figure 7: Skew impact of two colliding spheres with initial translational velocities in 3D (Section 6.5). The figure shows frame 70/145/253 with framerate $24Hz$.

6.3. Skew impact of two elastic cylinders

We run the same example as in section 4.2 of [7].

The grid spacing is $h = 1$. The first cylinder is initially centered at $(3, 3)$ with velocity $(0.75, 0)$. The second cylinder is initially centered at $(16, 5)$ with velocity $(-0.75, 0)$. Each cylinder has radius 2. MPM particles are sampled with alignment to the grid with spacing 0.5 (so 4 particles per cell for a full cell). Material density is 5. The constitutive model is Neo-Hookean with Young's Modulus 31.685 and Poisson's ratio 0.44022.

6.4. Elastic cylinder collision

We extend the previous example to two colliding hollow cylinders.

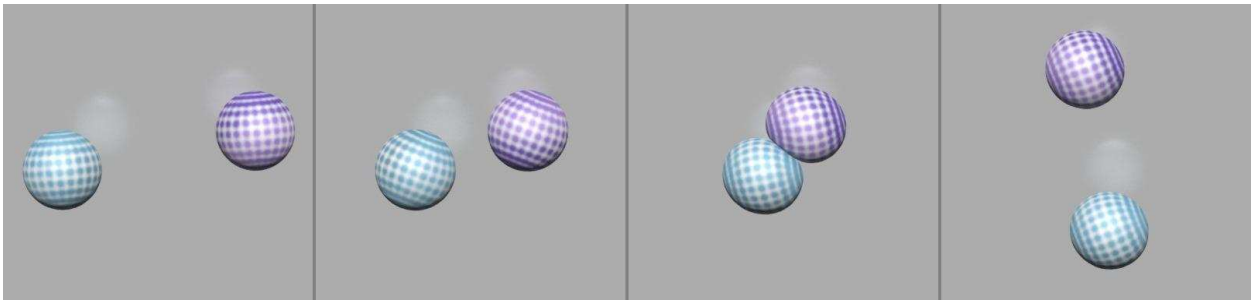

Figure 8: Skew impact of two colliding spheres with initial translational velocities and angular velocities in 3D (Section 6.5). The figure shows frame 1/34/63/92 with framerate $12Hz$.

The grid spacing is $h = 0.01$. The first ring is initially centered at $(0.1, 0.24)$ with velocity $(50, 0)$. The second ring is initially centered at $(0.4, 0.24)$ with velocity $(-50, 0)$. Each ring has outer radius 0.04 and inner radius 0.03. MPM particles are sampled with alignment to the grid with spacing 1/300. Material density is 1010. The constitutive model is Neo-Hookean with Young's Modulus 7.3e7 and Poisson's ratio 0.4.

6.5. Elastic sphere collision (3D)

We extend the skew impact of spheres to 3D. The grid spacing is $h = 30/256$. The first sphere is initially centered at $(10, 13, 15)$ with velocity $(0.75, 0, 0)$. The second sphere is initially centered at $(20, 15, 15)$ with velocity $(-0.75, 0, 0)$. Each sphere has radius 2. MPM particles are sampled with 4 particles per cell for a total particle count of $333,213$. Material density is 5. The constitutive model is Neo-Hookean with Young's Modulus 31.685 and Poisson's ratio 0.44022. Figure 7 shows the visualized objects at time 2.92, 6.04 and 10.54.

We further extend the previous test by initializing each sphere with an angular velocity of $(0, 0, 1)$ (i.e., the spheres initially rotate counterclockwise) and scaling the Young's modulus by 8. Figure 8 shows the visualized objects at time 0.08, 2.83, 5.25 and 7.67.

6.6. Torus dropping

We drop 25 tori (with 8592 particles each) into a box with width 0.4×0.4 and height 0.3. Each torus has inner radius 0.03 and outer radius 0.06 and is sampled at height 1.0 with random initial rotation around the ground normal. The material density is 5. Young's modulus is 150 and Poisson's ratio is 0.3. Figure 9 shows the particles and the reconstructed surfaces at time 8.50.

6.7. Ringing

The noise caused by FLIP transfers can be kept at tolerable levels in most cases, and our other numerical tests bear this out. Indeed, the usefulness of FLIP transfers depends on this. While investigating this noise,

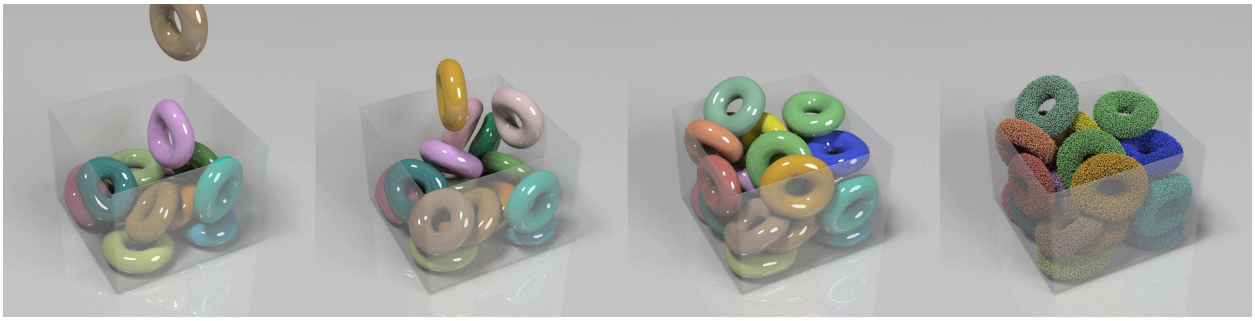

Figure 9: The proposed method is tested on a complicated scenario with 25 tori droppped into a container (Section 6.6). The simulation runs with framerate $24Hz$.

we observed that the FLIP transfer null modes can be intentionally excited in various ways, such as by carefully vibrating an object. We have also come across circumstances where FLIP happens to perform unusually badly. We document here one such test under which we observed this. The test, as shown in Figure 1, was originally reported in [22].

The simulation used the domain is $[0\,m, 1\,m]\times[0\,m, 1\,m]$, with a 22×22 node-grid, resulting in $\Delta x = \frac{1}{21}\,m$. Fixed objects inserted as objects. The ground is at $y = 0.35\,m$. The left wall is at $x = 0.1\,m$. Gravity is $2\,ms^{-2}$ downward.

The Δt is chosen in two steps. The desired time step is compute as $dt^* = \max(\Delta t_{\min}, \min(\Delta t_{\max}, \nu\frac{\Delta x}{\max(v_{\max}, v_\tau)}))$, where we have used $\Delta t_{\min} = 10^{-6}\,s$, $\Delta t_{\max} = 0.005\,s$, $\nu = 0.1$, and $v_\tau = 0.01\,ms^{-1}$. The max speed is computed as $v_{\max} = \max_i \|\mathbf{v}_i^n\|$. The second step is to ensure that we are able to output data at regular intervals of $\frac{1}{24}\,s$. Let Δt_f be the time until the next such time boundary. If $\Delta t_f < 1.001\Delta t^*$, then we take $\Delta t = \Delta t_f$ to avoid crossing the boundary. Otherwise, if $\Delta t_f < 2\Delta t^*$, we take $\Delta t = \frac{\Delta t_f}{2}$ to avoid sliver time steps. Otherwise, the time boundary is far away and we use $\Delta t = \Delta t^*$.

The projectile is initialized in the domain $[0.15\,m, 0.25\,m] \times [0.45\,m, 0.55\,m]$ with velocity $\langle 3\,ms^{-1}, 0\rangle$. The target on the right is initially stationary in the domain $[0.6\,m, 0.8\,m] \times [0.35\,m, 0.65\,m]$. The regular triangulations are visible in the figure. The top and bottom row of particles in the target are enforced as stationary using a zero-length spring penalty force computed as $\mathbf{f} = -c(\mathbf{x}_p^{n+1} - \mathbf{x}_p^0)$, where we have used $c = 10^5\,kg\,s^{-2}$. $\mathbf{x}_p^0$ is the initial position of the particle. The area associated with each particle is one-third of the area of all of its adjacent triangles. The mass of each particle is set so that its density is $4kg\,m^{-2}$. The constitutive model is fixed corotated [36] with stiffness $E = 10\,kg\,s^{-2}$ and Poisson's ratio $\nu = 0.3$ for both objects. The deformation gradient is computed from the current and initial particle locations using the Lagrangian mesh as in a fully Lagrangian scheme as in Jiang et al [22]. We note that this is equivalent to using the convected particle domain interpolation (CPDI) method [27] as noted in [26].

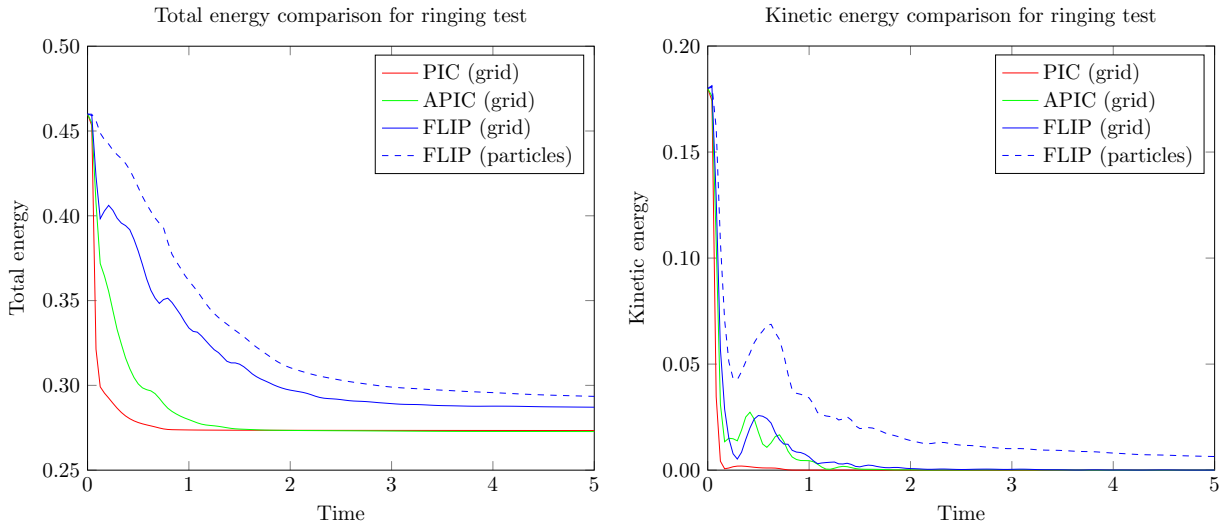

Figure 10: Total energy and kinetic energy evolution over time for the test in Section 6.7 (Figure 1).

The time integration differs from the one in this manuscript in that backward Euler is used rather than midpoint rule for the grid update and the deformation gradient is computed from a Lagrangian mesh (CPDI) rather than tracked as is typically done with MPM. Cubic B-splines are used for the interpolation kernel. The time points shown in Figure 1 are $t = \frac{2}{24}\ s$, $t = \frac{5}{24}\ s$, and $t = \frac{40}{24}\ s$. The target has almost stopped moving by the last frame in each case, so the final resting configuration for the target in each case closely resembles the last frame. The projectile continues to move for a while longer. In particular, the very dynamic-looking target in the last frame for the FLIP sim is in fact what the final resting configuration looks like for this sim. The expected result is, of course, for the target to look like the original, but sagging slightly under gravity. Both PIC and APIC eventually come to rest in this configuration.

6.8. Round trip transfer error

In this test we measure the errors caused by just transfers for both PIC and APIC. To do this, we initialize a periodic $[0, 2\pi] \times [0, 2\pi]$ domain with particles using Poisson disk particle seeding [37] with minimum particle separation of $\frac{\Delta x}{3}$. We initialize the grid velocity field to $\langle 1.1 \sin x, 0.9 \cos y \rangle$. We then do a transfer from grid to particles followed by a transfer from particles back to grid using PIC or APIC transfers with $\Delta t = 0$. This allows us to quantify the transfer errors in isolation from other error sources. With ideal transfers, the resulting grid velocity field would match the initial one. This round-trip transfer error represents the limiting behavior of a time step as $\Delta t \to 0$, since all other sources of error will vanish in this limit. We compute the error between the grid velocities before and after transfers in the L^2 norm. The results are shown in Figure 11.

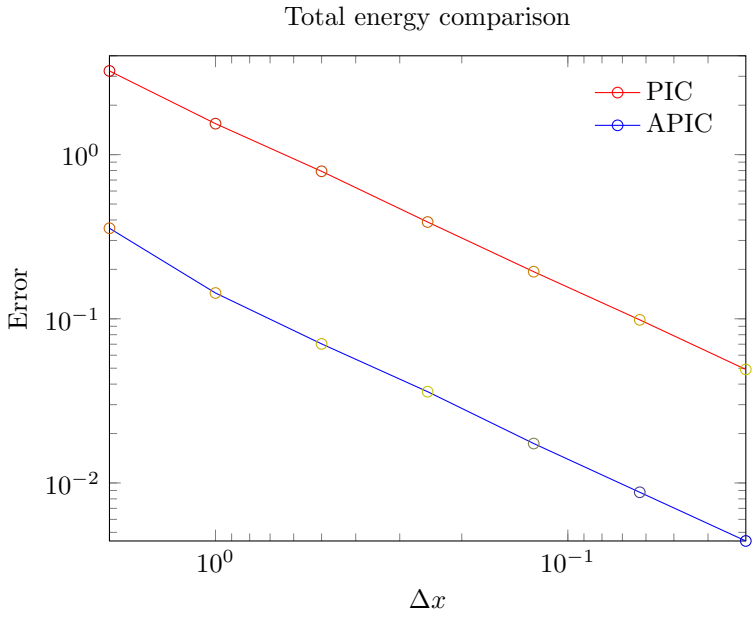

Figure 11: Comparison of transfer error across transfers from grid to particle and back to grid with $\Delta t = 0$. Both PIC and APIC transfer errors converge with first order accuracy, but APIC errors are an order of magnitude lower.

7. Discussion

The scheme described in this paper was constructed to demonstrate the angular momentum conservation properties of the proposed APIC transfers. The remaining elements of the discretization were chosen to be simple and defensible while supporting this goal. The transfers are relatively insensitive to other aspects of the discretization. We note that by using a PIC strategy, some short-wave length information that cannot be resolved by the grid will be dissipated, and this is a limitation of the approach. Note that in Figure 1 high frequency deformation is seen in the FLIP example, wheres both APIC and PIC damp this out. Thus while APIC does greatly reduce the dissipation of the original PIC, it does still suffer to some extent.

7.1. Update rule for $\mathbf{F}_p^{n+1}$

In (40), we used the simple first order update rule for $\mathbf{F}_p^{n+1}$. We have experimented with taking an extra Taylor series term.

$$\mathbf{F}_p^{n+1} = \mathbf{N}\mathbf{F}_p^n \tag{109}$$

$$\mathbf{N} = \mathbf{I} + \mathbf{H} + \frac{1}{2}\mathbf{H}^2 \tag{110}$$

$$\mathbf{H} = \sum_i (\tilde{\mathbf{x}}_i^{n+1} - \mathbf{x}_i^n)(\nabla w_{ip}^n)^T. \tag{111}$$

Adding the extra term complicates the force and its derivatives somewhat, so we have not used it here.

We have found uses for this update rule in other contexts. In particular, this rule improves stability for larger time steps by ensuring that $\det(\mathbf{F}_p^{n+1}) \geq 0$, which is important for many constitutive models. This property is also useful when using line searches to solve the nonlinear systems that arise with implicit time integration schemes, since line searches may explore configurations that result in $\det(\mathbf{F}_p^{n+1}) < 0$. This update does not ensure invertibility, though this is a degenerate value and does not occur frequently in practice.

The invariant $\det(\mathbf{F}_p^{n+1}) \geq 0$ is obtained by ensuring that $\det(\mathbf{N}) \geq 0$. Let $\mathbf{H} = \mathbf{P}\hat{\mathbf{H}}\mathbf{P}^{-1}$ be the eigenvalue decomposition of $\mathbf{H}$. Then, $\mathbf{N} = \mathbf{P}\hat{\mathbf{N}}\mathbf{P}^{-1}$ is the eigenvalue decomposition of $\mathbf{N}$. Since $\det(\mathbf{N}) = \det(\mathbf{P})\det(\hat{\mathbf{N}})\det(\mathbf{P}^{-1}) = \det(\hat{\mathbf{N}})$, $\det(\mathbf{N})$ is just the product of diagonal entries of $\hat{\mathbf{N}}$, which are of the form $1 + \lambda + \frac{\lambda^2}{2}$. Here, λ may occur as real numbers or complex conjugate pairs. Real entries contribute a factor of $1 + \lambda + \frac{\lambda^2}{2} = (1 + \frac{\lambda}{2})^2 + \frac{\lambda^2}{4} \geq 0$ to the product. Complex conjugate pairs contribute a factor of $(1 + \lambda + \frac{\lambda^2}{2})(1 + \overline{\lambda} + \frac{\overline{\lambda}^2}{2}) = \left|1 + \lambda + \frac{\lambda^2}{2}\right|^2 \geq 0$. Since all contributions are nonnegative, the determinant is also nonnegative.

The full matrix exponential goes a step further and ensures $\det(\mathbf{F}_p^{n+1}) > 0$, a property that no fixed number of Taylor terms can guarantee. The full matrix exponential is very difficult to differentiate, however, so we have never used it in an implicit scheme.

Acknowledgements

We thank Daniel Ram and Theodore Gast for their insightful suggestions. The authors were partially supported by NSF CCF-1422795, ONR (N000141110719, N000141210834), DOD (W81XWH-15-1-0147), Intel STC-Visual Computing Grant (20112360) as well as a gift from Disney Research.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

[14] J. Monaghan, On the problem of penetration in particle methods, J Comp Phys 82 (1) (1989) 1–15.

[15] J. Nairn, Numerical simulation of orthogonal cutting using the material point method, Eng Frac Mech 149 (2015) 262–275.

[16] C. Hammerquist, J. Nairn, A new method for material point method particle updates that reduces noise and enhances stability, Comp Meth App Mech Eng (pre-print).

[17] C. Gritton, M. Berzins, Improving accuracy in the mpm method using a null space filter, Comput Part Mech 4 (1) (2017) 131–142.

[18] C. Gritton, M. Berzins, R. Kirby, Improving accuracy in particle methods using null spaces and filters, Int Cent Num Meth Eng, 2015, pp. 202–213.

[19] J. Brackbill, G. Lapenta, Particle-in-cell magnetohydrodynamics, in: 16th Int Conf on the Numer Sim of Plasmas, 1998.

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

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

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

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

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

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

[26] V. Nguyen, C. Nguyen, T. Rabczuk, S. Natarajan, On a family of convected particle domain interpolations in the material point method, FEM Anal Des 126 (2017) 50 – 64.

[27] A. Sadeghirad, R. Brannon, J. Burghardt, A convected particle domain interpolation technique to extend applicability of the material point method for problems involving massive deformations, Int J Num Meth Eng 86 (12) (2011) 1435–1456.

[28] O. Gonzalez, Exact energy and momentum conserving algorithms for general models in nonlinear elasticity, Comp Meth in App Mech Eng 190 (13) (2000) 1763–1783.

[29] T. Laursen, X. Meng, A new solution procedure for application of energy-conserving algorithms to general constitutive models in nonlinear elastodynamics, Comp Meth in App Mech Eng 190 (46) (2001) 6309–6322.

[30] J. Simo, N. Tarnow, The discrete energy-momentum method. conserving algorithms for nonlinear elastodynamics, Zeitschrift für angewandte Mathematik und Physik ZAMP 43 (5) (1992) 757–792.

[31] J. C. Simo, N. Tarnow, K. Wong, Exact energy-momentum conserving algorithms and symplectic schemes for nonlinear dynamics, Comp Meth in App Mech Eng 100 (1) (1992) 63–116.

[32] C. Kane, Variational integrators and the newmark algorithm for conservative and dissipative mechanical systems, Ph.D. thesis, caltech (1999).

[33] A. Lew, J. Marsden, M. Ortiz, M. West, Variational time integrators, Int J Numer Meth Eng 60 (1) (2004) 153–212.

[34] T. F. Gast, C. Schroeder, Optimization integrator for large time steps, in: Proc ACM SIGGRAPH/Eurograph Symp Comp Anim, 2014.

[35] T. Gast, C. Schroeder, A. Stomakhin, C. Jiang, J. Teran, Optimization integrator for large time steps, IEEE Trans Vis Comp Graph 21 (10) (2015) 1103–1115.

[36] A. Stomakhin, R. Howes, C. Schroeder, J. Teran, Energetically consistent invertible elasticity, in: Proc Symp Comp Anim, 2012, pp. 25–32.

[37] R. Bridson, Fast poisson disk sampling in arbitrary dimensions, in: ACM SIGGRAPH 2007 Sketches, SIGGRAPH '07, 2007.