A New Method for Material Point Method Particle Updates that Reduces Noise and Enhances Stability

Chad C. Hammerquist[a], John A. Nairn[a,*]

[a] *Wood Science and Engineering, Oregon State University, Corvallis, OR 97330, USA*

Abstract

Current material point method (MPM) particle updates use a PIC approach, a FLIP approach, or a linear combination of PIC and FLIP. A PIC update filters velocity in each time step, which causes unwanted numerical diffusion, while FLIP eliminates that diffusion, but may retain too much noise. This paper develops a new particle update termed XPIC(m) (for eXtended PIC of order m) because it generalizes PIC updates. XPIC(1) is identical to current PIC methods, but higher orders of XPIC(m) address the over filtering and numerical diffusion of PIC, while still filtering out noise caused by the nontrivial null space of the extrapolation matrix used in MPM. As $m \to \infty$, XPIC(m) converges to a modified FLIP update with orthogonal removal of null space noise. The frequency response and filtering properties of XPIC(m) are investigated and several numerical examples demonstrate its advantages over other update methods.

Keywords: Material Point Method, Null Space, Computational Mechanics

1. Introduction

The Material Point Method (MPM) is a solid mechanics numerical tool that is well suited for solving problems involving complex geometries, large deformations, history-dependent materials, and contact. The method is a hybrid Eulerian-Lagrangian formulation where a modeled object is discretized into a set of material points or particles. Information needed to solve the equations of motions is extrapolated back and forth between material points and a background grid. MPM has been able to simulate a diverse array of complex problems. A partial list includes modeling complex biological structures [1] (including resolving cellular structure of wood [2–4]), granular materials [5], land slides and avalanches [6], explosive welding [7], cutting simulations [8, 9], sea ice dynamics [10], snow dynamics [11] and fracture simulations [12, 13].

Each dynamic MPM time step interpolates particle information to the grid, solves equations of motion on the grid, and then updates particle properties including stress, strain, plastic or damage history variables, velocity, and position. When updating velocity and position, two methods (or a mixture of the two) are currently available — **P**article **I**n **C**ell (or PIC) and **FL**uid **I**mplicit **P**article (or FLIP). PIC was developed by Harlow [14]. Although stable, it filters velocities that can lead to overdamping of kinetic energy in dynamic problems. FLIP was developed by Brackbill *et al.* [15, 16] to reduce dissipation by eliminating velocity filtering. It can, however, introduce noise that eventually reduces stability. PIC and FLIP particle methods, which were derived for fluids, were extended to solid mechanics by MPM [17]. The original MPM used FLIP methods and they remain the standard MPM particle update method.

Attempts to reduce FLIP noise include using a convex combination of FLIP and PIC [9, 11] and using a smoothing operator for spatial gradients [18]. While these approaches do reduce noise, they still suffer from unwelcome damping. Here, we propose a new method referred to as e**X**tended **PIC** or XPIC(m) of order $m = 1$ to ∞. XPIC(1) is identical to current PIC methods, which over filters velocities and may dissipate kinetic energy. As $m \to \infty$, XPIC(m) asymptotically approaches a modified FLIP update that removes all noise associated with the null space of the extrapolation matrix. This limit provides an optimal update method for MPM with ideal frequency properties, *i.e*, removal of null space noise without causing numerical diffusion or reducing accuracy

*Corresponding author: `john.nairn@oregonstate.edu`, Tel: +1-541-737-4265, Fax: +1-541-737-3385

Section 2 rederives FLIP and PIC velocity updates using inverse problem methodology. We then show that the new XPIC(m) method can be derived as a natural extension of that framework simply by revising the constraint in the inverse problem. The frequency response of XPIC(m) updates are examined along with convergence of XPIC(m) to orthogonal null space removal. Finally, Section 3 gives example problems that demonstrate some XPIC(m) properties and show advantages of XPIC(m) over prior methods.

2. Numerical Methods

2.1. MPM Extrapolations

MPM is based on a dual representation of a modeled object — a Lagrangian view (particles) and an Eulerian view (grid). The particles carry conserving quantities such as mass, momentum and (possibly) history-dependent material properties. Momenta and gradient information are computed on the grid and used to solve the momentum equation. In each time step, updated grid information (or Eulerian frame) $\mathcal{G}$ is linearly mapped to the particle space (or Lagrangian frame) $\mathcal{P}$:

$$\mathbf{S} : \mathcal{G} \to \mathcal{P} \tag{1}$$

where $\mathbf{S}$ is a matrix of interpolation coefficients. To create this array, the grid space is expanded in an approximate basis of "grid shape functions," denoted here as $N_i(\boldsymbol{x})$ for node i. The particle space is expanded in an approximate basis of "particle shape functions," that are based on a particle's position and domain and denoted here as $\chi_p(\boldsymbol{x})$ for particle p (it is typically 1 within the particle domain and zero elsewhere). The final MPM shape functions are created by convolution of these two bases [19].

$$S_{pi}(\boldsymbol{x}) = \frac{\int \chi_p(\boldsymbol{x}) N_i(\boldsymbol{x}) d\boldsymbol{x}}{\int \chi_p(\boldsymbol{x}) d\boldsymbol{x}} \tag{2}$$

These MPM shape functions define the elements S_{pi} of the $\mathbf{S}$ matrix corresponding to particle p and node i. These shape functions have partition of unity on the grid:

$$\sum_i S_{pi} = 1 \qquad \forall p. \tag{3}$$

The simplest mapping is a nearest grid point interpolation (NGP), where particles map all their information to their nearest grid node. In one dimension of a regular grid with cell size Δx, NGP interpolation corresponds to grid shape functions that are "box-car" functions (or $N_i(x) = \mathrm{rect}((x - x_i)/\Delta x)$ for node at x_i), and particle shape functions that are Dirac delta functions (or $\chi_p(x) = \delta(x - x_p)$ for particle at x_p). The FLIP and PIC precursors to MPM were developed around these shape functions [14–16]. The original development of MPM replaced "box-car" functions with overlapping tent functions (or $N_i(x) = \mathrm{tri}((x - x_i)/\Delta x)$) and retained Dirac delta functions ( ) on the particles [17]. The resulting MPM shape functions are in $\mathcal{C}^0$, but their discontinuous derivatives cause problems when particles cross cell boundaries. The generalized interpolation material point (GIMP) shape functions [19] were developed, in part, to improve cell-crossing issues. GIMP's mapping convolves overlapping tent functions ( ) on the grid with "box-car" functions (or $\chi_p = \mathrm{rect}((x - x_p)/l_p)$ where l_p is current length of the particle domain) on the particles. The resulting MPM shape functions are very similar to quadratic b-splines and are in $\mathcal{C}^1$. Other possible shape functions include CPDI [20], higher order b-splines [21], and many undeveloped ones that could be derived by GIMP methods [19].

The $\mathbf{S}$ matrix maps from the grid to the particles, but MPM also needs to map from the particles to the grid. When velocity is mapped from the particles to the grid, it is in the form of momentum. This extrapolation results in a mass-weighted mapping of particle velocities ($\boldsymbol{V}$) to grid velocities ($\boldsymbol{v}$) of:

$$\mathbf{S}^+ : \mathcal{P} \to \mathcal{G} \implies \boldsymbol{v} = \mathbf{S}^+ \boldsymbol{V} \tag{4}$$

where all MPM implementations use [17]:

$$\mathbf{S}^+ = \mathbf{D}\mathbf{S}^T\mathbf{M}, \quad \mathbf{D}^{n\times n} = \mathrm{diag}\left(\frac{1}{\mathbf{S}^T\boldsymbol{M}}\right), \quad \text{and} \quad \mathbf{M}^{N\times N} = \mathrm{diag}\left(\boldsymbol{M}\right) \qquad \left(i.e.,\ S^+_{ip} = \frac{M_p S_{pi}}{\sum M_p S_{pi}}\right) \tag{5}$$

where $\boldsymbol{M}$ is a vector of particle masses (M_p) and n and N are the number of nodes and particles, respectively. Throughout this paper we use lower case letters for grid quantities and upper case letters for particle quantities. Much MPM literature is based on the corresponding extrapolation of particle momentum ($\boldsymbol{P}$) to the grid using $\boldsymbol{p} = \mathbf{S}^T\mathbf{P}$ and elements S^T_{ip} are often denoted as S_{ip}; we changed this prior notation to the more consistent $S^T_{ip} = S_{pi}$.

In each MPM time step, $\mathbf{S}^+$ maps particle velocities to the grid and $\mathbf{S}$ maps them back. The mapping and its reverse come naturally from the variational framework of MPM [19] and was originally derived by a weighted, least-squares analysis of extrapolation followed by using a lumped mass matrix to derive the above elements of $\mathbf{S}^+$ [17]. A better approach would be to find $\mathbf{S}^+$ by inverting $\mathbf{S}$, but that approach has two problems. First, the number of particles is usually greater than the number of grid nodes in play, which means the $\mathbf{S}$ matrix is neither square nor invertible. Second, dealing with non-square $\mathbf{S}$ by finding a generalized Moore-Penrose inverse [22] on each time step would be prohibitively expensive. In contrast, the current reverse mapping in Eq. (5) is very efficient; in fact, it is almost free.

If the standard MPM shape functions are replaced by the simpler NGP shape function, and all particles have the same mass, we have observed that $\mathbf{S}^+$ in Eq. (5) is the exact Moore-Penrose inverse of $\mathbf{S}$ (see theorem 1 in the Appendix). Furthermore, an NGP analysis for velocity updates can be posed as a suite of inverse problems with exact solutions. By changing the constraint in the problem, various analyses lead to FLIP, PIC, or the new XPIC(m) update methods. For non-NGP interpolations, or when particles have different masses, the inverse solutions are no longer exact (or $\mathbf{S}^+$ is no longer the exact generalized inverse of $\mathbf{S}$), but the derived update methods can be treated as approximate solutions.

2.2. FLIP and PIC Derivations by Inverse Methodology

Although literature derivation of FLIP [15, 16] and PIC [14] updates were done differently, we show how to derive them using inverse methodology. Starting with a vector of updated velocities on the grid:

$$\boldsymbol{v}^{(k+)} = \boldsymbol{v}^{(k)} + \boldsymbol{a}^{(k)}\Delta t \tag{6}$$

where $\boldsymbol{v}^{(k)}$ and $\boldsymbol{a}^{(k)}$ are velocity and acceleration vectors on the grid in time step k, the MPM update needs to find particle velocities by inverting the particle-to-grid extrapolation:

$$\mathbf{S}^+\boldsymbol{V}^{(k+1)} = \boldsymbol{v}^{(k+)} \tag{7}$$

In general, there are more particles than active grid points, so Eq. (7) has an infinite number of solutions. To get a unique solution some constraint must be added. One choice is to constrain the new particle velocity to be close to the previous velocity, which gives this inverse problem:

$$\boldsymbol{V}^{(k+1)} = \underset{\boldsymbol{V}}{\text{argmin}} \ \ \|\boldsymbol{V} - \boldsymbol{V}^{(k)}\|_2 \qquad \text{such that:} \qquad \mathbf{S}^+\boldsymbol{V} = \boldsymbol{v}^{(k+)} \tag{8}$$

It follows from Theorem 2 in the Appendix that Eq. (8) has the solution:

$$\boldsymbol{V}^{(k+1)} = \mathbf{S}\boldsymbol{v}^{(k+)} + (\mathbf{I} - \mathbf{S}\mathbf{S}^+)\boldsymbol{V}^{(k)} \tag{9}$$

Rewriting Eq. (9) gives:

$$\boldsymbol{V}^{(k+1)} = \mathbf{S}(\boldsymbol{a}^{(k)}\Delta t + \mathbf{S}^+\boldsymbol{V}^{(k)}) + (\mathbf{I} - \mathbf{S}\mathbf{S}^+)\boldsymbol{V}^{(k)} = \boldsymbol{V}^{(k)} + \mathbf{S}\boldsymbol{a}^{(k)}\Delta t \tag{10}$$

which is the usual FLIP update [9, 11] (*i.e.*, increment particle velocity by acceleration extrapolated from the grid).

Another constraint choice, which will give a new style update, is to constrain magnitude of the updated velocity vector to be as small as possible:

$$\boldsymbol{V}^{(k+1)} = \underset{\boldsymbol{V}}{\text{argmin}} \ \ \|\boldsymbol{V}\|_2 \qquad \text{such that:} \qquad \mathbf{S}^+\boldsymbol{V} = \boldsymbol{v}^{(k+)} \tag{11}$$

which from Theorem 2 in the Appendix has the solution:

$$\boldsymbol{V}^{(k+1)} = \mathbf{S}\boldsymbol{v}^{(k+)} \tag{12}$$

This update is the usual PIC update [9, 11] (*i.e.*, replace particle velocity with velocity extrapolated from the grid). This formulation helps explain PIC dissipation — it is trying to drive the velocities towards zero in order to minimize its constraint. It achieves this minimization by diffusing the velocities and thereby dissipating kinetic energy.

Each MPM time step needs to update both particle velocity and position. Those updates can be written in a generalized form that accommodates FLIP and PIC updates as:

$$\boldsymbol{V}^{(k+1)} = \boldsymbol{V}^{(k)} + \int_0^{\Delta t} \mathbb{A}^{(k)}\,dt = \boldsymbol{V}^{(k)} + \mathbb{A}^{(k)}\Delta t \tag{13}$$

$$\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)} + \int_0^{\Delta t} \left(\mathbf{S}\boldsymbol{v}^{(k)} + \mathbb{A}^{(k)}t\right) dt = \boldsymbol{X}^{(k)} + \mathbf{S}\boldsymbol{v}^{(k+)}\Delta t + \left(\frac{1}{2}\mathbb{A}^{(k)} - \mathbf{S}\boldsymbol{a}^{(k)}\right)(\Delta t)^2 \tag{14}$$

where $\mathbb{A}^{(k)}$ is an *effective* acceleration for the chosen update method. The integrals were found by midpoint rule with $\mathbb{A}^{(k)}$ and $\boldsymbol{v}^{(k)}$ constant over the time step. FLIP or PIC style updates use these effective accelerations:

$$\mathbb{A}^{(k)} = \mathbf{S}\boldsymbol{a}^{(k)} \qquad \text{(FLIP)} \tag{15}$$

$$\mathbb{A}^{(k)} = \mathbf{S}\boldsymbol{a}^{(k)} - \frac{(\mathbf{I} - \mathbf{S}\mathbf{S}^{+})\boldsymbol{V}^{(k)}}{\Delta t} \qquad \text{(PIC)} \tag{16}$$

The PIC effective acceleration is the acceleration needed to replace particle velocity with velocity extrapolated from the grid. Obviously, a linear combination of FLIP and PIC [9, 11] uses:

$$\mathbb{A}^{(k)} = \mathbf{S}\boldsymbol{a}^{(k)} - \frac{f_{PIC}(\mathbf{I} - \mathbf{S}\mathbf{S}^{+})\boldsymbol{V}^{(k)}}{\Delta t} \tag{17}$$

where f_{PIC} is fraction PIC in the update. Many MPM codes simplify to a first order position update — $\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)} + \mathbf{S}\boldsymbol{v}^{(k+)}\Delta t$. Although this approach works to first order for pure FLIP updates, it is unacceptable for a generalized update including both FLIP and PIC. When using some PIC, the *apparently* second order $(\Delta t)^2$ term in Eq. (14) expands to include a first order term [9], which implies that a generalized position update has to be second order.

2.3. XPIC(m) Derivation

Alternatives to FLIP and PIC updates can be derived by changing the constraint imposed on the inverse problem. The goals for a new update method are to reduce both velocity diffusion caused by PIC and noise caused by FLIP. A potential constraint to achieve these goals is to minimize the difference between new velocity $\boldsymbol{V}^{(k+1)}$ and a smoothed previous velocity $\boldsymbol{V}_{sm}$. The justification is that minimizing a velocity *difference* reduces velocity diffusion while differencing with a *smoothed* velocity may reduce noise. The inverse problem becomes:

$$\boldsymbol{V}^{(k+1)} = \underset{\boldsymbol{V}}{\text{argmin}} \ \ \|\boldsymbol{V} - \boldsymbol{V}_{sm}\|_2 \qquad \text{such that:} \qquad \mathbf{S}^{+}\boldsymbol{V} = \boldsymbol{v}^{(k+)} \tag{18}$$

which for NGP shape functions has the exact solution:

$$\boldsymbol{V}^{(k+1)} = \mathbf{S}\boldsymbol{v}^{(k+)} + (\mathbf{I} - \mathbf{S}\mathbf{S}^{+})\boldsymbol{V}_{sm} \tag{19}$$

An efficient (and effective) method to derive a smoothed velocity is to extrapolate initial grid velocity back to the particles or $\boldsymbol{V}_{sm} = \mathbf{S}\boldsymbol{v}^{(k)} = \mathbf{S}\mathbf{S}^{+}\boldsymbol{V}^{(k)}$. Using this smoothing, the update in Eq. (19) becomes:

$$\boldsymbol{V}^{(k+1)} = \mathbf{S}\boldsymbol{v}^{(k)} + (\mathbf{I} - \mathbf{S}\mathbf{S}^{+})\mathbf{S}\mathbf{S}^{+}\boldsymbol{V}^{(k)} \tag{20}$$

This update leads to a new effective acceleration for particle updates:

$$\mathbb{A}^{(k)} = \mathbf{S}\boldsymbol{a}^{(k)} - \frac{(\mathbf{I} - \mathbf{S}\mathbf{S}^{+})^2\boldsymbol{V}^{(k)}}{\Delta t} \qquad \text{XPIC(2)} \tag{21}$$

The only difference between XPIC(2) and PIC is that the $(\mathbf{I}-\mathbf{SS}^{+})$ term is now squared (hence the "2" in XPIC(2)). We are led to propose a generalized XPIC(m) method with an effective acceleration of:

$$\mathbb{A}^{(k)} = \mathbf{S}\boldsymbol{a}^{(k)} - \frac{(\mathbf{I}-\mathbf{SS}^{+})^{m}\boldsymbol{V}^{(k)}}{\Delta t} \qquad \text{XPIC}(m) \tag{22}$$

Comparing to Eq. (19), XPIC(m) corresponds to using the smoothed velocity:

$$\boldsymbol{V}_{sm} = \big((\mathbf{I} - (\mathbf{I}-\mathbf{SS}^{+})^{m-1}\big)\boldsymbol{V}^{(k)} \tag{23}$$

The physical significance of this result is addressed below.

The above NGP derivations for FLIP, PIC, and XPIC(m) are exact inverse problem solutions (for special case of all particles having the same mass). When NGP solutions are replaced by MPM shape functions (or for particles with different masses), the updates are no longer exact solutions to the inverse problems, but can be interpreted as approximate solutions under the specified constraints. Insights from those constraints reveal that PIC filters out too much velocity to meet its constraint to minimize velocity vector magnitude while FLIP eliminates that filtering, but may retain too much noise. Our new XPIC(m) method introduces minimization to a constraint based on difference with a smoothed velocity that may both minimize dissipation and reduce noise.

2.4. XPIC(m) Properties

To quantify XPIC(m) performance, we considered its spatial filtering properties. Updated particle velocities can be written as filtered particle velocities from the previous step plus extrapolated grid acceleration:

$$\boldsymbol{V}^{(k+1)} = \mathbf{P}\boldsymbol{V}^{(k)} + \mathbf{S}\boldsymbol{a}^{(k)}\Delta t \tag{24}$$

where $\mathbf{P}$ is a projection matrix that defines the particle velocity filtering. The projection $\mathbf{P}$ is determined by the update method:

$$\mathbf{P} = \mathbf{I} \qquad \text{FLIP} \tag{25}$$

$$\mathbf{P} = \mathbf{SS}^{+} \qquad \text{PIC } (= \text{XPIC}(1)) \tag{26}$$

$$\mathbf{P} = \mathbf{I} - (\mathbf{I}-\mathbf{SS}^{+})^{m} \qquad \text{XPIC}(m) \tag{27}$$

It is evident that PIC and XPIC(m) updates filter velocities while FLIP does no filtering. As done in Refs. [14–16], consider a simple 1D example with N particles of equal mass, evenly spaced by Δx, and classic MPM linear shape functions. The velocity of particle $p = 0$ to $N-1$ can be represented in spatial frequency space by:

$$V_p = \frac{1}{N}\left(v_0 + \sum_{k=1}^{N-1} v_k e^{2\pi i k p/N}\right) \qquad \text{where} \qquad v_k = \sum_{p=0}^{N-1} V_p e^{-2\pi i k p/N} \tag{28}$$

are coefficients from discrete Fourier transform of the particle velocities. When PIC or XPIC(m) is used in MPM, the velocities are convolved in each step with the above filtering matrix $\mathbf{P}$. The filtered velocity of particle p becomes:

$$(\mathbf{P}\boldsymbol{V})_p = \frac{1}{N}\left(v_0 + \sum_{k=1}^{N-1} P_k v_k e^{2\pi i k p/N}\right) \tag{29}$$

Here, the $P_k \in [0,1]$ coefficients represent damping coefficient at each spatial frequency. Due to shape function properties, the constant velocities are undamped. The Fourier transform of the linear shape function mapping of this problem to the grid is given as:

$$P_k = \text{sinc}^2(k\Delta x) \tag{30}$$

where $\text{sinc}(x) = \sin(x\pi)/(x\pi)$. This transform also holds for mapping back to the particles. Therefore the Fourier transform of the $\mathbf{P}$ matrix for PIC with linear shape functions is given as:

$$P_k = \text{sinc}^4(k\Delta x) \tag{31}$$

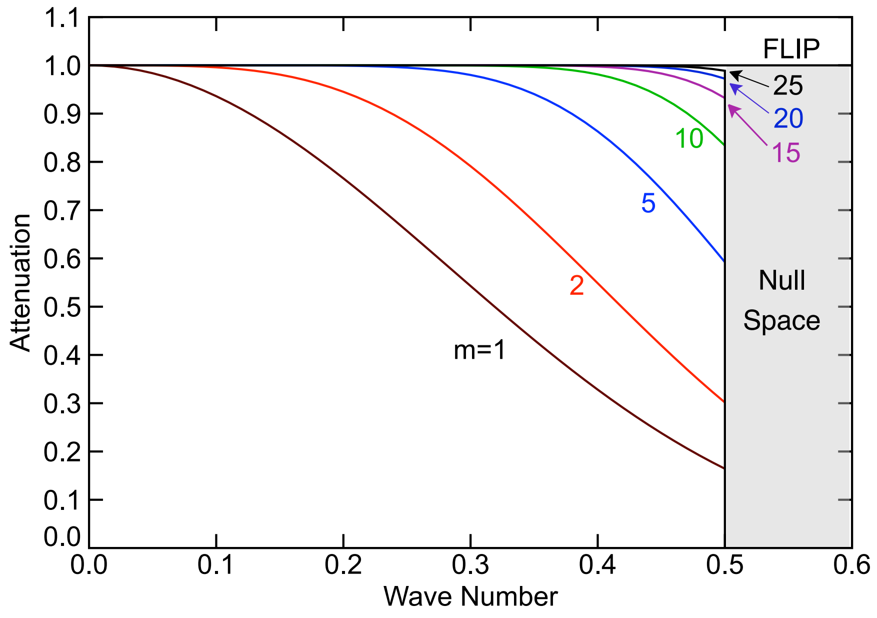


Figure 1: Frequency Response of FLIP, PIC (or XPIC(1)) and several orders of XPIC(m) in one dimension with evenly spaced particles of equal mass, and the classic MPM shape functions. The shaded area is the null space that is removed by PIC and XPIC(m),

Following this logic, the Fourier transform for XPIC(m) filtering matrix $\mathbf{P}$ is given as:

$$P_k = 1 - \big(1 - \operatorname{sinc}^4(k\Delta x)\big)^m \tag{32}$$

The frequency responses for FLIP, PIC, and XPIC(m) are plotted in Fig. 1. Because FLIP does no filtering, its response is a horizontal line for all frequencies. Given the discretization of the filtering, both PIC and XPIC(m) completely remove all null space components, which is plotted as frequency response dropping to zero for wave number greater than 0.5 ($i.e$, wavelength equal to the size of two grid cells), which corresponds to the Nyquist sampling frequency on the grid. The responses for PIC and XPIC(m) differ for wave number less than 0.5. For PIC, all the frequencies, except the constant velocity, are damped. This frequency response explains the large diffusion caused by PIC. In contrast, the XPIC(m) frequency response has a larger region with little or no damping. Even XPIC(2), for example, has very little attenuation for any frequencies with a wave number less than 0.1 ($i.e$, wavelength greater than 10 grid cells). As m grows, the frequency response for XPIC(m) converges to a brick-wall filter at the null space of $\mathbf{S}^+$.

If $m \to \infty$ (and all particles have the same mass), XPIC(m) converges to exact orthogonal null space removal, regardless of shape function type, problem dimension, or spatial distribution of the particles. In other words, Theorem 3 in the Appendix shows that:

$$\lim_{m\to\infty} (\mathbf{I} - \mathbf{S}\mathbf{S}^+)^m = \boldsymbol{\Omega} \tag{33}$$

where $\boldsymbol{\Omega}$ is an orthogonal projection onto the null space of $\mathbf{S}^+$. The null space $\mathcal{N}(\mathbf{S}^+)$ of matrix $\mathbf{S}^+$ is the linear subspace of all vectors that $\mathbf{S}^+$ maps to zero:

$$\mathbf{S}^+ \boldsymbol{z} = 0 \qquad \forall \boldsymbol{z} \in \mathcal{N}(\mathbf{S}^+). \tag{34}$$

Conceptually, $\mathcal{N}(\mathbf{S}^+)$ represents all velocity modes on the particles that are not mapped to the grid. Substituting into Eq. (27), as $m \to \infty$, the XPIC(m) filtering matrix becomes

$$\mathbf{P} = \mathbf{I} - \boldsymbol{\Omega} \tag{35}$$

which is an orthogonal projection onto the orthogonal complement of the null space of $\mathbf{S}^+$. Any velocity modes (and only those modes) invisible to the grid are removed by this filtering. Consequently the grid

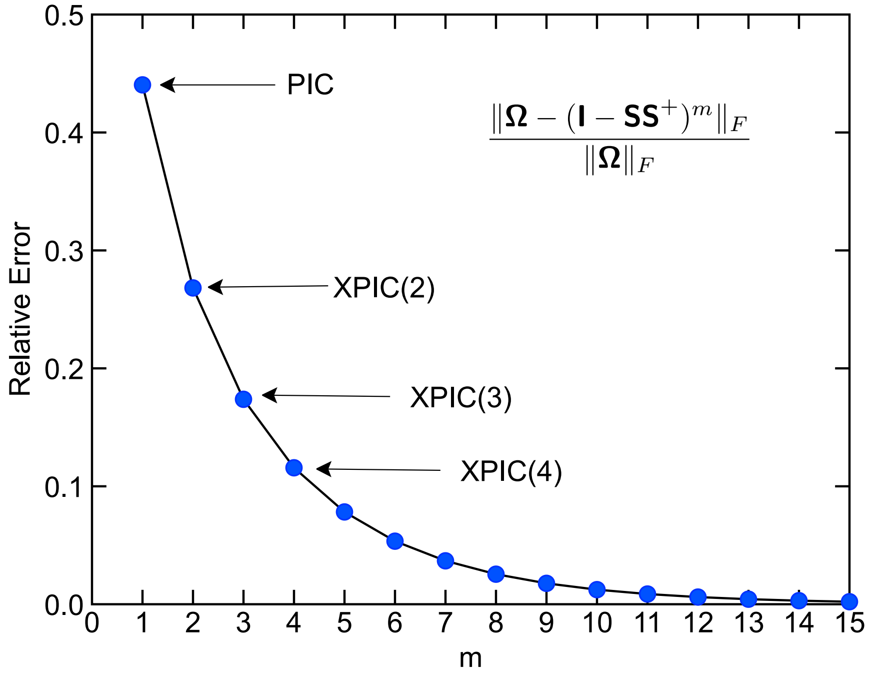


Figure 2: Convergence of $(\mathbf{I} - \mathbf{S}\mathbf{S}^+)^m$ to $\mathbf{\Omega}$ for 1D example with 50 particles. The relative error is the equation shown on the plot,

space $\mathcal{G}$ and the particle space $\mathcal{P}$ are on the same page, working with the same information after the least possible amount of damping. The physical significance of Eq. (23) now is that XPIC(m) minimizes difference between updated velocity and previous velocity with null space modes removed.

For finite m in calculations, XPIC(m) for increasing m represents successive approximations to orthogonal null space removal. To test the convergence, we built a simple 1D example with 50 equally-spaced particles using linear shape functions. For this simple example, we could use singular value decomposition (SVD) to exactly calculate the null space projection operator, $\mathbf{\Omega}$. Figure 2 plots normalized magnitude of the difference between $(\mathbf{I} - \mathbf{S}\mathbf{S}^+)^m$ and the exact $\mathbf{\Omega}$ as a function of the XPIC(m) order. The results show geometric convergence to the null space projection (*i.e.*, $\log(||\mathbf{\Omega} - (\mathbf{I} - \mathbf{S}\mathbf{S}^+)^m||_F)$ converges linearly in m).

Convergence to $\mathbf{\Omega}$ is significant because previously an orthogonal null space removal would require a matrix decomposition in each time step, which has computational cost of $O(N^3)$ [18, 23]. Due to sparsity of $\mathbf{S}$, the XPIC(m) method scales linearly in the number of particles (see section 2.5). If some particles have different mass, Eq. (33) is no longer exact, but instead converges to an oblique projection onto the null space of $\mathbf{S}^+$. Our numerical experiments, however, suggest nearly identical effects. Furthermore, a common approach to MPM when modeling particles of different materials having different densities (and hence different masses) is to use multiple velocity fields — one velocity field for each material type [24]. In this so-called multimaterial mode MPM, each velocity field updates separately and normally uses constant-mass particles. The XPIC(m) method should therefore approach the appropriate null space removal. The velocity fields interact by contact [24] or interface laws [25] that act as boundary conditions to those fields.

2.5. Implementation Details

The details for efficiently (as possible) implementing XPIC(m) are given here. The particle effective acceleration for XPIC(m) is:

$$\mathbb{A}^{(k)}\Delta t = \mathbf{S}\boldsymbol{a}^{(k)}\Delta t - (\mathbf{I} - \mathbf{S}\mathbf{S}^+)^m\boldsymbol{V}^{(k)} = \mathbf{S}\boldsymbol{a}^{(k)}\Delta t - \boldsymbol{V}^{(k)} + m\mathbf{S}\boldsymbol{v}^{(k)} - m\mathbf{S}\boldsymbol{v}^* \tag{36}$$

where

$$\boldsymbol{v}^* = \frac{1}{m}\sum_{r=2}^{m}\binom{m}{r}(-1)^r(\mathbf{S}^+\mathbf{S})^{r-1}\boldsymbol{v}^{(k)} = \sum_{r=2}^{m}(-1)^r\boldsymbol{v}_r^* \tag{37}$$

Table 1: To calculate $\boldsymbol{v}^*$ allocate space for three velocities on each node ($\boldsymbol{v}^*$, $\boldsymbol{v}_-$, and $\boldsymbol{v}_+$) and use the algorithm presented in this table. For the inner i and j loops, ϑ_p is the set of nodes $\{k \ : \ S_{pk} \neq 0\}$ or nodes for particle p with non-zero shape function — this set will have a small number of members and its size will be independent of problem size.

```
let $\boldsymbol{v}^* \leftarrow 0$ and $\boldsymbol{v}_- \leftarrow \boldsymbol{v}^{(k+)} - \boldsymbol{a}^{(k)}\Delta t$
for each $r$ in $2$ to $m$ do
    let $\boldsymbol{v}_+ \leftarrow 0$
    for each $p$ in $N$ do
        for each $i$ in $\vartheta_p$ do
            for each $j$ in $\vartheta_p$ do
                let $(\boldsymbol{v}_+)_i \leftarrow (\boldsymbol{v}_+)_i + \frac{m-r+1}{r}\frac{M_p S_{pi} S_{pj}}{m_i^{(k)}}(\boldsymbol{v}_-)_j$
    let $\boldsymbol{v}^* \leftarrow \boldsymbol{v}^* + (-1)^r \boldsymbol{v}_+$ and $\boldsymbol{v}_- \leftarrow \boldsymbol{v}_+$
```

One could include grid or particle damping, such as described in Ref. [9], but those terms are omitted here to focus on XPIC(m) terms. The new $\boldsymbol{v}^*$ term, which is unique to XPIC(m) with $m > 1$, can be evaluated by recursion:

$$\boldsymbol{v}_r^* = \frac{1}{m}\binom{m}{r}(\mathbf{S}^+\mathbf{S})^{r-1}\boldsymbol{v}^{(k)} = \frac{1}{m}\binom{m}{r}\mathbf{S}^+\mathbf{S}(\mathbf{S}^+\mathbf{S})^{r-2}\boldsymbol{v}^{(k)} = \frac{m-r+1}{r}\mathbf{S}^+\mathbf{S}\boldsymbol{v}_{r-1}^* \tag{38}$$

starting with $\boldsymbol{v}_1^* = \boldsymbol{v}^{(k)}$.

In typical MPM codes, the particle update is done after $\boldsymbol{v}^{(k+)}$ has replaced $\boldsymbol{v}^{(k)}$. Using only updated grid velocities, the recursion relation starts with $\boldsymbol{v}_1^* = \boldsymbol{v}^{(k+)} - \boldsymbol{a}^{(k)}\Delta t$ and the effective acceleration becomes:

$$\mathbb{A}^{(k)}\Delta t = (1-m)\mathbf{S}\boldsymbol{a}^{(k)}\Delta t - \boldsymbol{V}^{(k)} + m\mathbf{S}\boldsymbol{v}^{(k+)} - m\mathbf{S}\boldsymbol{v}^* \tag{39}$$

Explicit particle updates for position and velocity are:

$$\boldsymbol{V}^{(k+1)} = \boldsymbol{V}^{(k)} + \left(\mathbf{S}\boldsymbol{a}^{(k)} - \boldsymbol{a}_{ex}^{(k)}\right)\Delta t \tag{40}$$

$$\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)} + \mathbf{S}\boldsymbol{v}^{(k+)}\Delta t - \left(\mathbf{S}\boldsymbol{a}^{(k)} + \boldsymbol{a}_{ex}^{(k)}\right)\frac{(\Delta t)^2}{2} \tag{41}$$

where $\boldsymbol{a}_{ex}^{(k)}$ appears as a "damping" term, but here is implementing XPIC(m):

$$\boldsymbol{a}_{ex}^{(k)}\Delta t = -m\mathbf{S}\big(\boldsymbol{v}^{(k+)} - \boldsymbol{a}^{(k)}\Delta t\big) + \boldsymbol{V}^{(k)} + m\mathbf{S}\boldsymbol{v}^* \tag{42}$$

Note again, that while many MPM codes stop the position update with the first term involving updated grid velocities, the implementation of XPIC(m) requires the *apparent* second order term because part of that term evaluates to first order.

For most MPM codes, the addition of an XPIC(m) option reduces to a new calculation of $\boldsymbol{v}^*$ just before the particle update; pseudocode for that calculation is given in Table 1. The increment in $\boldsymbol{v}_+$ corresponds to the evaluation of $\boldsymbol{v}_r^*$ by Eq. (38) on node i:

$$(\boldsymbol{v}_r^*)_i = \frac{m-r+1}{r}\sum_p \frac{M_p S_{pi}}{m_i^{(k)}}(\mathbf{S}\boldsymbol{v}_{r-1}^*)_p = \frac{m-r+1}{r}\sum_p\sum_j \frac{M_p S_{pi} S_{pj}}{m_i^{(k)}}(\boldsymbol{v}_{r-1}^*)_j \tag{43}$$

Due to sparsity of $\mathbf{S}$, the number of nodes needed for the i and j loops will be small and independent of the problem's size. The calculation therefore scales linearly with the XPIC(m) order times the number of particles (N) (linearly with mN) and is easily parallelized.

3. Results and Discussion

This section presents examples that show improvements of XPIC(m) methods over existing methods and demonstrate some XPIC(m) properties.

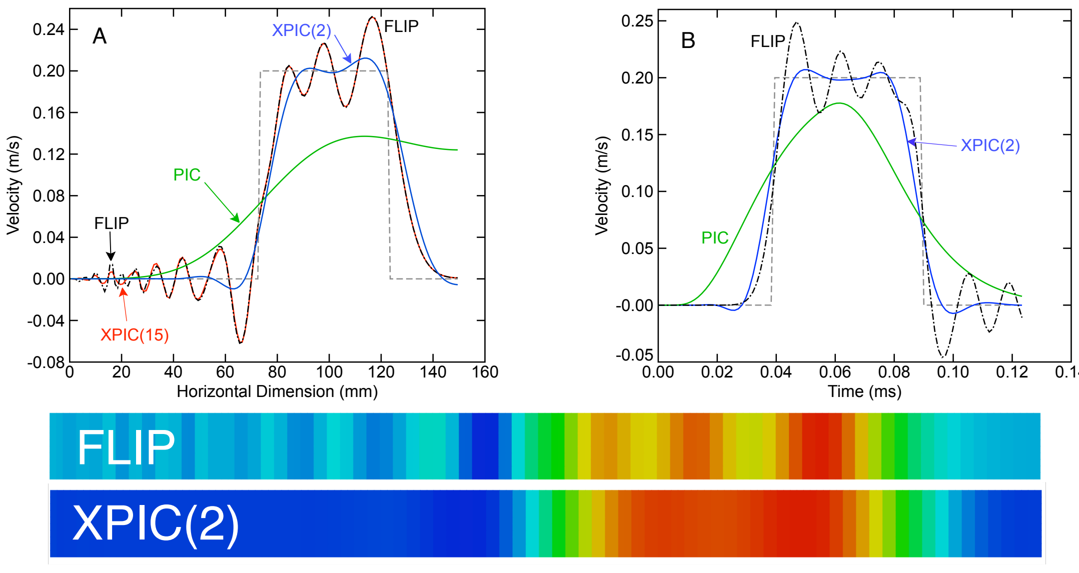


Figure 3: Simulation results for velocity square wave propagating through an elastic bar. A. Spatial solution at $t = (0.12323ms)$ for FLIP (black dash-dot line), PIC, XPIC(2) and XPIC(15) (red solid line, mostly on top of FLIP results). B. Temporal solution at $(x = 40mm)$ for FLIP (black dash-dot line), PIC, and XPIC(2). Analytical solution plotted in dashed gray line. The colored bars on the bottom plot axial stress in the particles for FLIP and XPIC(2) updates at $t = (0.12323ms)$.

3.1. Square velocity wave in an elastic bar

The first example is a velocity square wave propagating through an elastic bar in plane stress. The material was isotropic, linear elastic with modulus $E = 1$ GPa, $\nu = 0.33$ and $\rho = 1$ g/cm^3. The bar was given symmetry boundary conditions on both sides to avoid internal reflections. The background grid had 2×2 mm cells, the time step was 8.2154×10^{-5} ms, and uGIMP shape functions were used (note: uGIMP shape functions are GIMP shape functions where $\chi_p(\boldsymbol{x})$ is 1 in the translated but undeformed particle domain and 0 elsewhere [19]). This simulation (and all simulations in this paper), had four particle per cell. A 20 kHz square pulse (0.5 ms duration) velocity condition was applied to one end of the bar (at $x = 0$):

$$V_x = 0.2\,\mathrm{rect}(20(t + 0.25)) \tag{44}$$

for V_x in m/s and t in ms. The incident wave should propagate along the bar with analytical solution:

$$V_{x,t} = 0.2\,\mathrm{rect}(20(t + 0.25) - 0.001x) \tag{45}$$

MPM results for this problem and various velocity update methods are in Fig. 3. The two plots show spatial snapshot for $t = 0.12323$ ms (A) and temporal response at a fixed location ($x = 40$ mm) (B). The colored bars below the plots show axial stress by FLIP and XPIC(2) corresponding to the spatial snapshot plot. The FLIP solution has the expected oscillatory ringing for a finite discretization of a square wave. The PIC solution shows massive diffusion of the pulse, demonstrating PIC is poor choice for problems with high frequencies. The XPIC(2) solution has much less damping. It is able to remove the high frequency components of the FLIP solutions while still retaining a good reproduction of the expected square wave.

No method on a discretized grid can exactly represent a square wave pulse. For this simple problem, increasing m in XPIC(m) dampens fewer high frequency components and becomes similar to ringing by FLIP updates. The MPM results for XPIC(15), for example, are shown in the spatial plot in Fig. 3A. Although the XPIC(15) results are close to FLIP, they still remove some high frequency noise from the FLIP solution around $x = 20$ mm. In this region, XPIC(15) showed smooth ringing that gradually increased in amplitude, while with FLIP, the early ringing was noisier and variable in amplitude. In other words, XPIC(m) converges to FLIP with noise removal and not to FLIP. Differences between XPIC(15) and FLIP are small in the square wave example, but larger differences are seen in subsequent examples.

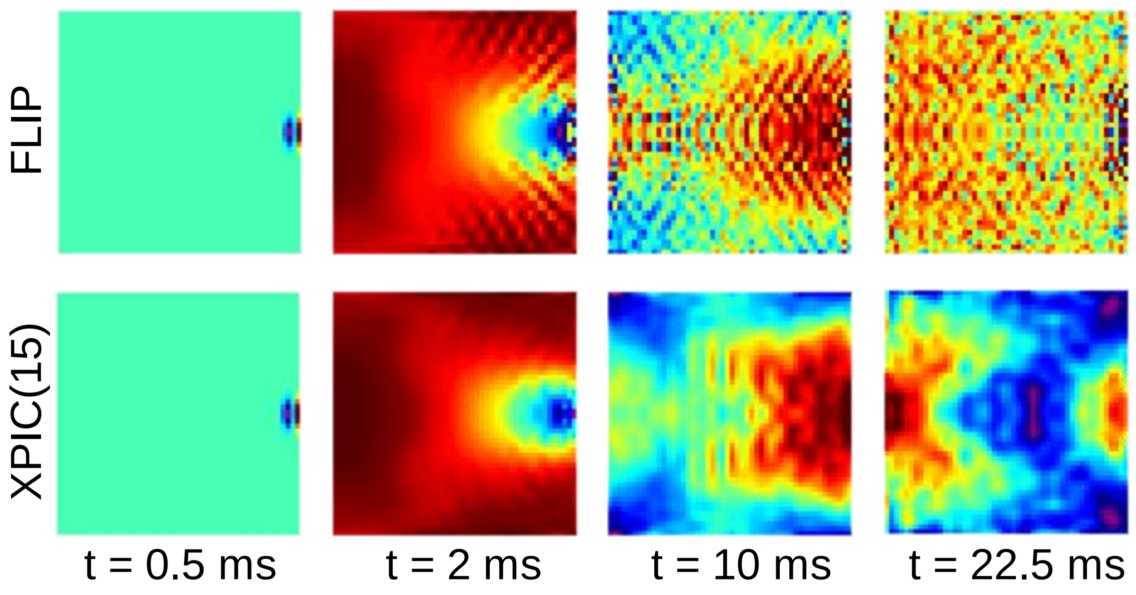


Figure 4: Snapshots of velocity magnitudes in an elastic square impacted at a point on the right edge at 0.5 ms. The top row is for FLIP updates and the bottom row is for XPIC(15) updates.

3.2. Velocity waves in an elastic square

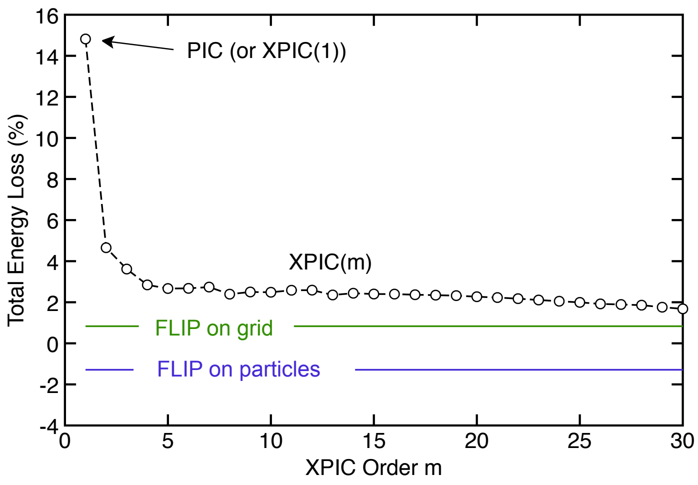
For a 2D wave propagation problem, we simulated side impact of a soft elastic square under plane stress conditions. The square was 50×50 mm and modeled with a hyperelastic neohookean material [26] with $E = 5$ MPa, $\nu = 0.33$, and $\rho = 1.5$ g/cm^3. The background grid had 1×1 mm cells, the time step was 0.01423 ms, and uGIMP shape functions were used. The square initially moved horizontally through the grid at $V_0 = 2.5$ m/s until it hit a boundary condition point with zero velocity just before $t = 0.5$ ms, which initiated stress waves and caused the square to bounce back.

Snapshots of velocity fields within the square are given in Fig. 4 for MPM results using FLIP or XPIC(15) updates. At early times, FLIP and XPIC(15) were similar, but even by $t = 2$ ms, FLIP developed noise that is absent in XPIC(15) results. At later times, the FLIP simulation devolved almost entirely into noise while XPIC(15) is still representing some complex wave patterns. The tensile wave speed for this material was 57.7 mm/s, which translates to 1.7 ms for wave to reflect back on fourth within the square. Thus the snapshots at $t = 10$ and $t = 22.5$ ms correspond to multiple stress wave reflections within the square.

3.3. Two Colliding Disks

To examine energy conservation, we considered a simulation where two circular disks collide. The two, 25 mm radius disks were modeled with a hyperelastic neohookean material [26] with $E = 1$ MPa, $\nu = 0.33$, and $\rho = 1$ g/cm^3. The background grid had 2×2 mm cells, the time step was 0.026 ms, and uGIMP shape functions were used. The disks were moving towards each other with initial speed of 0.316 m/s (or 1% of the material's wave speed). Contact between the disks was resolved explicitly using multimaterial MPM methods based on both contact pressure and relative displacement of the two disks [25].

For an elastic impact, total energy (kinetic energy plus strain energy) should be conserved. MPM is close to conservation, but is not perfect. The total energy lost in the simulations at $t = 174$ ms, which was after contact and bouncing apart, is plotted in Fig. 5 for FLIP and XPIC(m) as a function of m. For FLIP, the total energy measured on the particles ("FLIP on particles") actually increased about 1.3%. This increase may be caused by velocity noise that particularly affects kinetic energy calculations. The "FLIP on grid" line shows total energy loss where kinetic energy is evaluated on the grid during each time step instead of from particle velocity. Energy conservation by this calculation shows about 0.8% loss. For XPIC(m), the PIC method (*i.e.*, XPIC(1)) lost about 15% of the energy and provides an example of the PIC method causing too much damping. As m increased, the energy conservation improved significantly. Even XPIC(2)



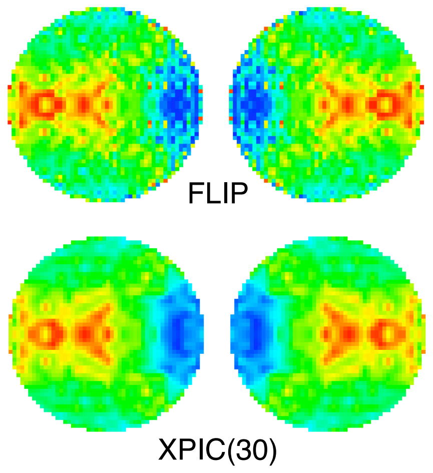


Figure 5: Left: Total energy (kinetic energy plus strain energy) loss *vs.* XPIC(m) order for colliding disks simulations. The energy loss for FLIP used kinetic energy calculated either on the particles or on the grid. Right: Kinetic energy (by colors) in the two disks by FLIP or XPIC(30) at $t = 34.786$ ms just after colliding and then separating.

was much better than PIC; it lost only 1/3 the energy lost by PIC. After about XPIC(8), the total energy lost was less than 2.5%.

In summary, XPIC(m) filters velocities, which will cause some energy dissipation, but it provides optimal filtering in that it removes mostly noise that can cause instability in FLIP simulations. When m is large enough, the total energy loss is small. The snapshots on the right compare FLIP to XPIC(30). The FLIP method develops noise that is mostly eliminated in the XPIC(30) simulation.

3.4. Performance considerations

A disadvantage of XPIC(m) is increased computational time with increasing m. Section 2.5 claims this cost scales linearly with m. To verify this scaling, the relative computational cost for the previous colliding disks problem is plotted in Fig. 6 as a function of m. The straight line fit shows that the additional cost of XPIC(m) — it is linear in m.

A possible approach to minimizing computational cost, would be to only periodically run an XPIC(m) update. For example, imagine running a FLIP simulation, but then every X time steps replace the FLIP update with an XPIC(25) update. The hope is that XPIC(25) is doing an excellent job of removing noise in the null space and perhaps the noise reduction can be done every X steps instead of every step. To quantify the null space noise, we defined a relative null space noise level R_k as a ratio of the quadratic mean of the noise removed in time step k to the initial velocity:

$$R_k = 100\% \times \frac{1}{||\boldsymbol{V}_0||}\sqrt{\sum_N \left[\left(\overline{\boldsymbol{V}}^{(k)}_{FLIP} - \overline{\boldsymbol{V}}^{(k)}_{XPIC(25)}\right)\cdot\left(\overline{\boldsymbol{V}}^{(k)}_{FLIP} - \overline{\boldsymbol{V}}^{(k)}_{XPIC(25)}\right)\right]} \tag{46}$$

where $\overline{\boldsymbol{V}}^{(k)}_{FLIP}$ and $\overline{\boldsymbol{V}}^{(k)}_{XPIC(25)}$ are the velocities that would occur in time step k if the previous update was done by FLIP or XPIC(25), respectively. This metric tracks magnitude of the velocity components that could be removed by an XPIC(25) update (a close approximation to null space components in that time step). This "noise" metric can be calculated using:

$$\overline{\boldsymbol{V}}^{(k)}_{FLIP} - \overline{\boldsymbol{V}}^{(k)}_{XPIC(25)} = (\mathbf{I} - \mathbf{SS}^+)^{25}\boldsymbol{V}^{(k-1)} = \boldsymbol{V}^{(k-1)} - 25\mathbf{S}(\boldsymbol{v}^{(k-1)} - \boldsymbol{v}^*) \tag{47}$$

To test noise reduction, we repeated the side impact of an elastic square (see section 3.2) which accumulated significant noise with FLIP. Figure 7 plots evolution of R_k *vs.* time for a FLIP simulation and four XPIC(25) simulations where the XPIC(25) noise reduction was applied every step ($X = 1$) or only every $X = 5$, 25, and 100 steps. The FLIP simulation's noise increased and eventually reached a high plateau. XPIC(25) with $X = 1$, noise is removed every time step and remains low (dashed line in Fig. 7). For

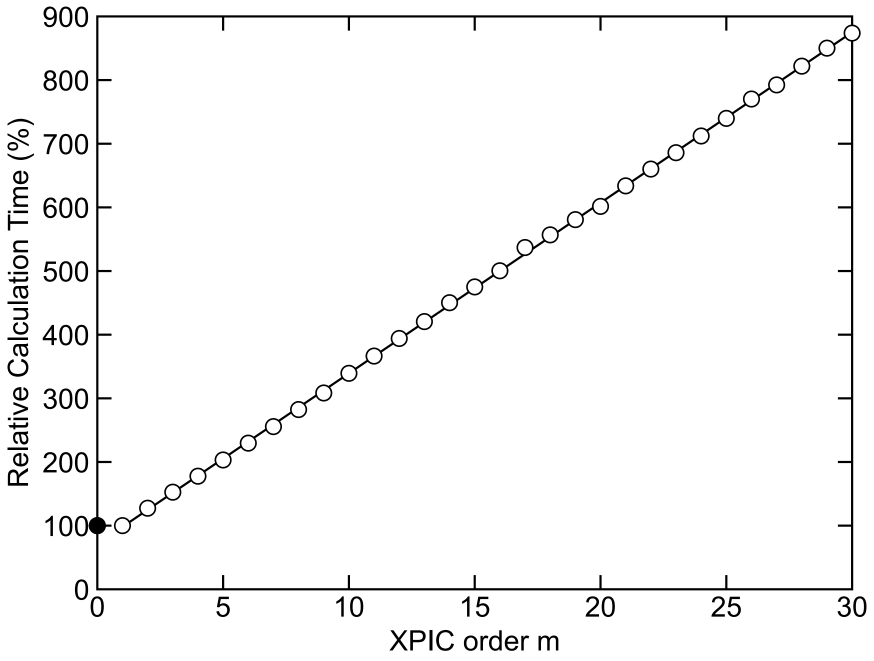


Figure 6: Computational time relative to FLIP as a function of order m in XPIC(m) calculations for the colliding disks simulations. The FLIP time is the solid dot plotted for $m = 0$.

XPIC(25) with $X > 1$, the simulations allowed some noise, but provided considerable noise reduction compared to FLIP at significantly lower computational cost compared to XPIC(25) with $X = 1$. The periodic drops in noise occurred each time the XPIC(25) update was invoked. The choice of X likely depends on the problem. Perhaps an adaptive method could start with an initial X and then increase it or decrease it depending on evolution of R_k. This approach will be the subject of future work.

3.5. Future research

Reference [18] addresses numerical instabilities caused by the nontrivial null space of $\mathbf{S}^+$, by considering spatial gradients in MPM instead of particle updates done here. To remove null space noise, they applied the PIC filtering matrix, $\mathbf{P} = \mathbf{S}\mathbf{S}^+$, to the gradients. This does help solve numerical instabilities, but at the cost of increased numerical diffusion and decreased accuracy. The XPIC(m) methodology presented here could also be applied to spatial gradients, which could limit the numerical diffusion that plagued Ref. [18]. We have not yet investigated this possibility.

4. Conclusions

We presented a new update method named XPIC(m) (for eXtended PIC) for explicit MPM codes. The XPIC(m) method, which generalizes the PIC update, addresses much of the numerical diffusion of PIC and removes numerical instabilities found in FLIP. We showed that for large m, XPIC(m) converges to an orthogonal removal of null space components. For lower values of m, XPIC(m) dampens some components not within the null space, but such behavior might provide a useful, and adjustable, frequency-dependent damping scheme. Several examples were presented demonstrating the improvement of XPIC(m) simulations compared to FLIP and PIC simulations.

Acknowledgements

This work was supported by the National Institute of Food and Agriculture (NIFA) award #2013-34638-21483, of the United States Department of Agriculture (USDA).

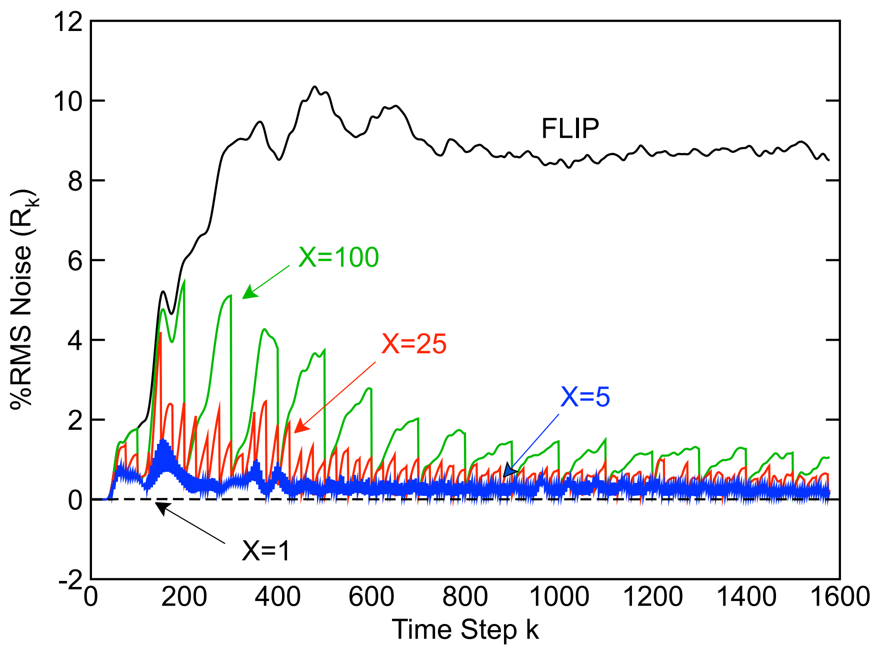


Figure 7: Evolution of noise, as measured by R_k, with time step for FLIP and XPIC(25) as well as for FLIP with XPIC(25) applied every $X = 5$ steps, 25 steps, or 100 steps.

References

[1] J. E. Guilkey, J. B. Hoying, J. A. Weiss, Computational modeling of multicellular constructs with the material point method, Journal of Biomechanics 39 (11) (2006) 2074–2086.

[2] J. A. Nairn, Numerical simulations of transverse compression and densification in wood, Wood and Fiber science 38 (4) (2007) 576–591.

[3] P. Perré, G. Almeida, M. Ayouz, X. Frank, New modelling approaches to predict wood properties from its cellular structure: image-based representation and meshless methods, Annals of Forest Science 73 (1) (2016) 147–162.

[4] J. A. Nairn, Material point method simulations of transverse fracture in wood with realistic morphologies, Holzforschung 61 (4) (2007) 375–381.

[5] S. Bardenhagen, J. Brackbill, D. Sulsky, The material-point method for granular materials, Computer Methods in Applied Mechanics and Engineering 187 (3) (2000) 529 – 541.

[6] C. M. Mast, P. Arduino, G. R. Miller, P. Mackenzie-Helnwein, Avalanche and landslide simulation using the material point method: flow dynamics and force interaction with structures, Computational Geosciences 18 (5) (2014) 817–830.

[7] Y. Wang, H. Beom, M. Sun, S. Lin, Numerical simulation of explosive welding using the material point method, International Journal of Impact Engineering 38 (1) (2011) 51 – 60.

[8] R. Ambati, X. Pan, H. Yuan, X. Zhang, Application of material point methods for cutting process simulations, Computational Materials Science 57 (2012) 102 – 110.

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

[10] D. Sulsky, H. Schreyer, K. Peterson, R. Kwok, M. Coon, Using the material-point method to model sea ice dynamics, Journal of Geophysical Research: Oceans 112 (C02S90) (2007) 1–18.

[11] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, A. Selle, A material point method for snow simulation, ACM Trans. Graph. 32 (4) (2013) 102:1–102:10.

[12] J. A. Nairn, Material point method calculations with explicit cracks, Computer Modeling in Engineering and Sciences 4 (6) (2003) 649–664.

[13] S. G. Bardenhagen, J. A. Nairn, H. Lu, Simulation of dynamic fracture with the material point method using a mixed J-integral and cohesive law approach, International Journal of Fracture 170 (1) (2011) 49–66.

[14] F. Harlow, The particle in cell computing method for fluid dynamics, Methods in Computational Physics 3 (1964) 319.

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

[16] J. Brackbill, D. Kothe, H. Ruppel, FLIP: A low-dissipation, particle-in-cell method for fluid flow, Computer Physics Communications 48 (1) (1988) 25 – 38.

[17] D. Sulsky, Z. Chen, H. Schreyer, A particle method for history-dependent materials, Computer Methods in Applied Mechanics and Engineering 118 (1) (1994) 179 – 196.

[18] C. Gritton, M. Berzins, R. M. Kirby, Improving accuracy in particle methods using null spaces and filters, in: Proceedings of the 4th International Conference on Particle-Based Methods - Fundamentals and Applications, PARTICLES 2015, International Center for Numerical Methods in Engineering, 2015, pp. 202–213.

[19] S. G. Bardenhagen, E. M. Kober, The generalized interpolation material point method, Computer Modeling in Engineering and Sciences 5 (6) (2004) 477–496.

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

[21] M. Steffen, P. C. Wallstedt, J. E. Guilkey, R. M. Kirby, M. Berzins, Examination and analysis of implementation choices within the material point method (MPM), Computer Modeling in Engineering and Sciences 31 (2) (2008) 107–127.

[22] A. Ben-Israel, T. N. Greville, Generalized Inverses, Theory and Application, Springer-Verlag, New York, 2003.

[23] L. N. Trefethen, D. Bau III, Numerical linear algebra, Vol. 50, Siam, 1997.

[24] S. G. Bardenhagen, J. E. Guilkey, K. M. Roessig, J. U. Brackbill, W. M. Witzel, J. C. Foster, An improved contact algorithm for the material point method and application to stress propagation in granular materials, Computer Modeling in Engineering and Sciences 2 (2001) 509–522.

[25] J. A. Nairn, Modeling of imperfect interfaces in the material point method using multimaterial methods, Computer Modeling in Engineering and Sciences 92 (3) (2013) 271–299.

[26] R. W. Ogden, Non-Linear Elastic Deformations, Dove Publications, Inc., Mineola, New York (page 222), 1984.

Appendix

We derive some exact results for nearest-neighbor grid point (NGP) shape functions applied to problems where all particles have the same mass.

Theorem 1. *If* $\mathbf{S}$ *is the matrix of NGP shape functions and all the particles have the same mass, then* $\mathbf{S}$ *is the Moore-Penrose inverse of* $\mathbf{S}^+$.

Proof. Consider $\mathbf{S}$, it is a non-negative matrix created from nearest neighbor interpolations. By definition, each particle maps to only one grid point, so each row has only one nonzero element and it is equal to one. Consequentially, the columns of $\mathbf{S}$ are orthogonal and $\mathbf{S}$ can be into partitioned by its nonzero columns. Each column has the Moore-Penrose generalized inverse (denoted with †):

$$S_i^\dagger = ||S_i||^{-2} S_i^T \tag{48}$$

Because all elements of S_i are either 1 or 0:

$$S_i^\dagger = (\sum_p S_i)^{-1} S_i^T \tag{49}$$

Because the columns are orthogonal (*i.e.*. $S_i S_j^T = 0$ for $i \neq j$), the Moore-Penrose inverse of $\mathbf{S}$, or $\mathbf{S}^\dagger$, is the matrix of columns $S_i^\dagger$. Thus for constant mass particles

$$\mathbf{S}^\dagger = \mathbf{D}\mathbf{S}^T = \mathbf{S}^+ \quad \text{where} \quad \mathbf{D} = \operatorname{diag}\left(\sum_p S_{pi}\right)^{-1} \tag{50}$$

By the properties of Moore-Penrose generalized inverses, $(\mathbf{S}^+)^\dagger = (\mathbf{D}\mathbf{S}^T)^\dagger = \mathbf{S}$. □

Theorem 2. *If A is a linear map such that $A : \mathbb{R}^n \to \mathbb{R}^m$ with $m < n$, then the inverse problem:*

$$\tilde{x} = \underset{x}{\operatorname{argmin}} \quad \|x - x_0\|_2 \qquad \text{such that :} \qquad \text{Ax = b} \tag{51}$$

has the solution

$$\tilde{x} = A^\dagger b + (I - A^\dagger A)x_0 \tag{52}$$

where $A^\dagger$ is the Moore-Penrose inverse of A.

Proof. Introduce Lagrange multipliers:

$$L(x, \lambda) = (x - x_0)^T (x - x_0) + \lambda^T (Ax - b) \tag{53}$$

Differentiating with respect to x and λ gives:

$$\nabla_x L = 2(x - x_0) + A^T \lambda \qquad \text{and} \qquad \nabla_\lambda L = Ax - b \tag{54}$$

Setting $\nabla_x L = \nabla_\lambda L = 0$ and solving for x gives:

$$x = A^T (AA^T)^{-1} b - A^T (AA^T)^{-1} A x_0 + x_0 \tag{55}$$

Because all elements of A are real, its Moore-Penrose inverse is $A^\dagger = A^T (AA^T)^{-1}$ and Eq. (55) simplifies to:

$$x = A^\dagger b + (I - A^\dagger A) x_0 \tag{56}$$

□

Theorem 3. *If* $\mathbf{S}^+$ *represents the mapping matrix from the particles to grid and* $\mathbf{S}$ *represents the mapping from grid to particles, and all particles have the same mass, then*

$$\lim_{m \to \infty} (\mathbf{I} - \mathbf{S}\mathbf{S}^+)^m = \mathbf{\Omega} \tag{57}$$

where Ω is an orthogonal projection onto the null space of $\mathbf{S}^+$.

Proof. The matrix $(\mathbf{I} - \mathbf{S}\mathbf{S}^{+})$ can be written as $(\mathbf{I} - \mathbf{S}\mathbf{D}\mathbf{S}^{T})$. Decompose $\mathbf{S}$ using weighted singular value decomposition: $\mathbf{S} = U\Sigma V^{T}$ where U is an $N \times N$ orthogonal matrix, Σ is an $N \times n$ matrix containing singular values of the matrix $\mathbf{S}$ on the main diagonal, and V is an $n \times n$ matrix that is orthonormal over its constraint, *i.e.* $V^{T}\mathbf{D}V = \mathbf{I}$. Partition the decomposition by its non-zero singular values:

$$\mathbf{S} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} V_1^{T} \\ V_2^{T} \end{bmatrix} \tag{58}$$

where Σ_r is a diagonal matrix with dimension of the rank of $\mathbf{S}$. Solve for Σ:

$$\begin{aligned} \Sigma \quad &= \quad \begin{bmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} U_1^{T} & U_2^{T} \end{bmatrix} \mathbf{S}\mathbf{D} \begin{bmatrix} V_1 & V_2 \end{bmatrix} \implies \begin{bmatrix} U_1^{T}\mathbf{S}\mathbf{D}V_1 & U_2^{T}\mathbf{S}\mathbf{D}V_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \end{bmatrix} \\ &\implies \quad U_2^{T}\mathbf{S}\mathbf{D}V_1 = \mathbf{0} \ \text{ and } \ U_2^{T}\mathbf{S}\mathbf{D}V_2 = \mathbf{0} \implies U_2^{T}\mathbf{S}\mathbf{D} = \mathbf{0} \\ &\implies \quad \mathbf{D}\mathbf{S}^{T}U_2 = \mathbf{S}^{+}U_2 = \mathbf{0} \end{aligned} \tag{59}$$

So the matrix U_2 forms an orthonormal basis for the null space of $\mathbf{S}^{+}$. Rewriting the matrix of interest:

$$(\mathbf{I} - \mathbf{S}\mathbf{S}^{+})^{m} = U_1(1 - \Sigma_r^{2})^{m}U_1^{T} + U_2U_2^{T} \tag{60}$$

The matrix $\mathbf{S}\mathbf{S}^{+}$ is a doubly stochastic matrix, so all its eigenvalues are $\in [0, 1]$, the diagonal elements of $\Sigma_r \in (0, 1]$, and it follows that:

$$\lim_{m\to\infty} (1 - \Sigma_r^{2})^{m} = 0 \tag{61}$$

Finally, $\lim_{m\to\infty} (\mathbf{I} - \mathbf{S}\mathbf{S}^{+})^{m} = U_2U_2^{T}$, which is an orthogonal projection onto the null space of $\mathbf{S}^{+}$. $\square$