A Moving Least Squares Material Point Method with Displacement Discontinuity and Two-Way Rigid Body Coupling

YUANMING HU[†], MIT CSAIL
YU FANG[†], Tsinghua University
ZIHENG GE[†], University of Science and Technology of China
ZIYIN QU, University of Pennsylvania
YIXIN ZHU[†], University of California, Los Angeles
ANDRE PRADHANA, University of Pennsylvania
CHENFANFU JIANG, University of Pennsylvania

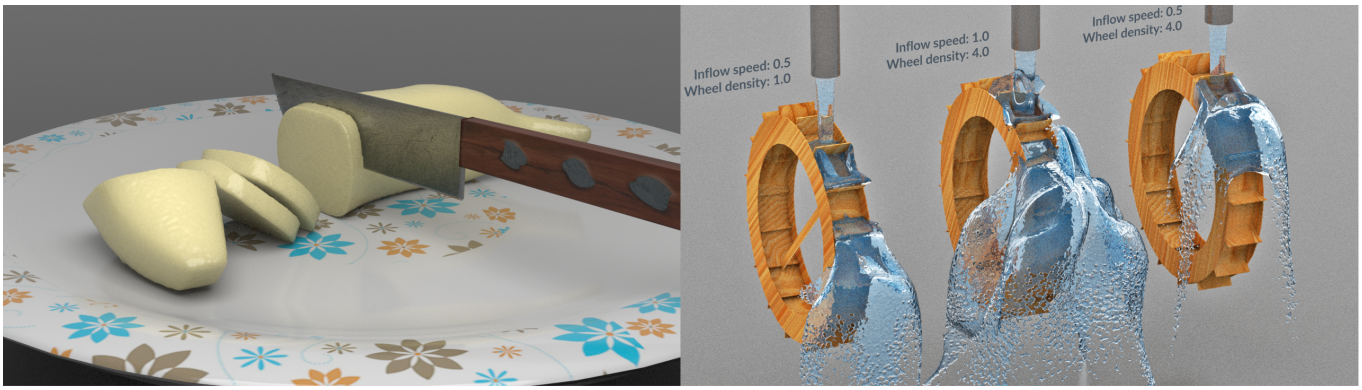

Fig. 1. Our method allows MPM to handle world space material cutting, complex thin boundaries and natural two-way rigid body coupling.

In this paper, we introduce the Moving Least Squares Material Point Method (MLS-MPM). MLS-MPM naturally leads to the formulation of Affine Particle-In-Cell (APIC) [Jiang et al. 2015] and Polynomial Particle-In-Cell [Fu et al. 2017] in a way that is consistent with a Galerkin-style weak form discretization of the governing equations. Additionally, it enables a new stress divergence discretization that effortlessly allows all MPM simulations to run two times faster than before. We also develop a Compatible Particle-In-Cell (CPIC) algorithm on top of MLS-MPM. Utilizing a colored distance field representation and a novel compatibility condition for particles and grid nodes, our framework enables the simulation of various new phenomena that are not previously supported by MPM, including material cutting, dynamic open boundaries, and two-way coupling with rigid bodies. MLS-MPM with CPIC is easy to implement and friendly to performance optimization.

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

Additional Key Words and Phrases: Material Point Method (MPM), moving least squares, cutting, discontinuity, distance field, rigid coupling

ACM Reference format:
Yuanming Hu[†], Yu Fang[†], Ziheng Ge[†], Ziyin Qu, Yixin Zhu[†], Andre Pradhana, and Chenfanfu Jiang. 2018. A Moving Least Squares Material Point Method with Displacement Discontinuity and Two-Way Rigid Body Coupling. *ACM Trans. Graph.* 37, 4, Article 1 (August 2018), 14 pages.
DOI: 10.1145/3197517.3201293

DOI: 10.1145/3197517.3201293

1 INTRODUCTION

Since the pioneering work of Terzopoulos et al. [1988], simulating topologically changing materials has been a popular research topic in graphics. Among various topics, fracture and cutting of deformable objects are most intensively explored. The goal of breaking mesh connectivity has led to techniques such as local remeshing [O'Brien et al. 2002; O'Brien and Hodgins 1999], the Virtual Node Algorithm (VNA) [Hegemann et al. 2013; Molino et al. 2005; Wang et al. 2014] and the eXtended Finite Element Method (XFEM) [Koschier et al. 2017]. Maintaining remeshing quality efficiently and robustly can be however very complicated. While VNA and XFEM reduce some difficulty, they impose additional challenges like floating point arithmetic in degenerate scenarios and self-collision on embedded surfaces.

Compared to mesh-based approaches, meshless animation of solid topology change was shown to be promising by Pauly et al. [2005]. More recently, the Material Point Method (MPM) [Sulsky et al. 1995] emerged as an effective choice for various materials and gained

† Y. Hu, Y. Fang, Z. Ge and Y. Zhu were visiting students at the University of Pennsylvania during this work.

popularity in VFX and in animations such as Disney's *Frozen* [Stomakhin et al. 2013]. Utilizing both meshless Lagrangian particles and a background Eulerian grid, MPM has become advantageous in simulating multi-physics phenomena, as shown in [Stomakhin et al. 2014] and [Tampubolon et al. 2017]. In contrast to FEM, MPM automatically supports arbitrarily extreme topologically changing dynamics including material split and merge. It also does not suffer from boundary difficulties such as the tensile instability in Smoothed Particle Hydrodynamics (SPH).

Despite its high efficacy in many situations, traditional MPM fails to model sharp separation of material points and cannot represent discontinuous velocities. We show a 2D cutting example in Fig. 2. Directly colliding with a thin level set (Fig. 2a) fails due to the under-resolution of the collision object. Plastic softening (Fig. 2b), which is common for modeling material failure, results in too many debris particles. Finally, particle deletion (Fig. 2c) requires removing a noticeable number of particles, which causes visual artifacts. The main issue is that particle kernels in MPM are *nonconforming* to boundaries. Each particle exchanges data with grid nodes in its entire kernel support across the boundary which naturally causes velocity field smoothing. This issue is more pronounced in MPM than FLIP-based fluids [Zhu and Bridson 2005] due to the wider kernel support of quadratic or cubic B-splines required for solids [Steffen et al. 2008].

1.1 Contributions

For resolving these fundamental issues, we develop a Compatible Particle-In-Cell (CPIC) algorithm that allows for material point discontinuity and infinitely thin boundaries based on relative locations between particles and grid nodes. Unlike node-visibility-based algorithms that are common in element-free Galerkin (EFG) crack simulations [Belytschko et al. 1994; Belytschko and Tabbara 1996] or the transparency method used by Pauly et al. [2005], our formulation does not require any expensive ray mesh intersection queries. Additionally, CPIC facilitates two-way rigid-MPM coupling in a straightforward fashion.

Our framework is based on a novel weak form discretization of MPM. We show that the low dissipation Affine PIC[Jiang et al. 2015, 2017b] and Polynomial PIC [Fu et al. 2017] methods can be derived from a Galerkin-style Moving Least Squares (MLS) discretization of the governing equations. We extend the idea and use MLS to further replace the shape functions in the stress divergence term. This leads to a new force computation scheme that does not require evaluating the gradients of nodal shape functions. Compared to traditional MPM, the resulting Moving Least Squares Material Point Method (MLS-MPM) provides almost identical visual results and enables an effortless 2× speed up with easier implementation.

2 RELATED WORK

2.1 Deformable Objects Fracture and Cutting

Fracture simulation was pioneered by Terzopoulos et al. [1988]. With FEM, the most simple and efficient approach for handling cutting and fracture is to split surfaces along element boundaries [Müller and Gross 2004]. A more accurate strategy splits individual elements, as pioneered by O'Brien et al. for brittle [1999] and ductile fracture [2002] via locally remeshing tetrahedral elements according to the embedded fracture surface. Their algorithm preserves the orientation of fracture surfaces during the remeshing process. Bao et al. [2007] presented a novel algorithm for efficient fracture of nearly rigid materials. Kaufmann et al. [2009] used Discontinuous Galerkin Finite Element Method (DGFEM) for handling discontinuities. Hegemann et al. [2013] minimizes the Griffith's energy for ductile fracture of embedded level sets. Pauly et al. [2005] presented a meshless framework for elastoplastic fracture, where explicit crack surfaces are initiated with stress criteria on particles. Chen et al. [2014] developed an efficient adaptive remeshing method based on gradient descent flow, which automatically refines fracture surfaces. Pfaff et al. [2014] also performed adaptive remeshing for fracturing thin sheets. Hahn and Wojtan [2015; 2016] used Boundary Element Method (BEM) and Lagrangian crackfronts to produce highly detailed fracture surfaces.

For material cutting, the Virtual Node Algorithm (VNA) by Molino et al. [2005] duplicates (instead of splitting) simulation elements that intersect the cutting geometry. The original VNA only allows one cut per face and does not handle degenerate cases. To overcome these shortcomings, Sifakis et al. [2007] improved it to allow arbitrarily generalized cutting surfaces at smaller scales than tetrahedron resolution. Wang et al. [2014] further developed a robust adaptive VNA with robust floating point arithmetic for degenerate intersections. Recently, Koschier et al. [2017] presented a new remeshing-free cutting algorithm with XFEM, which was shown to better preserve physical plausibility such as mass conservation and correctly maintained stiffness properties. We refer to the survey by Wu et al.[2015] for more detailed previous work on cutting.

2.2 Fluid Boundaries and Rigid-Fluid Coupling

While there exists a lot of previous work on resolving solid fluid interaction and complex boundaries, we will focus on reviewing the treatment of thin shell rigid boundaries, which is most relevant to our work. Carlson et al. [2004] presented the Rigid Fluid method where rigid bodies are resolved on the Eulerian grid through a rigidity projection. This approach works best when the rigid body is not extremely thin. The first work considering thin solid/fluid coupling is by Guendelman et al. [2005]. They used a robust ray casting algorithm to augment the velocity interpolation and kernel computation near surfaces. Later work further improved the stability [Robinson-Mosher et al. 2008; Shinar et al. 2008] and accuracy [Robinson-Mosher et al. 2009] near boundaries. Chentanez et al. [2006] combined fluid pressure projection and elasticity integration into simultaneous equations and enabled the usage of large time steps. To obtain higher accuracy of boundary handling, Klingner et al. [2006] proposed two-way rigid-fluid coupling based on conforming unstructured meshes and remeshing. Feldman et al. [2005] also adopted boundary conforming tetrahedral meshes for discretizing the domain. However these approaches have not been investigated for treating thin shell, dynamic, rigid bodies. Batty et al. [2007] proposed a variational pressure projection (at sub-grid resolution) to account for partial cell volume. Narain et al. [2010] adopted this formulation for coupling frictional stress of granular media with rigid bodies. Azevedo et al. [2016] extended the cut-cell approach to enable one way coupling between hybrid Lagrangian/Eulerian

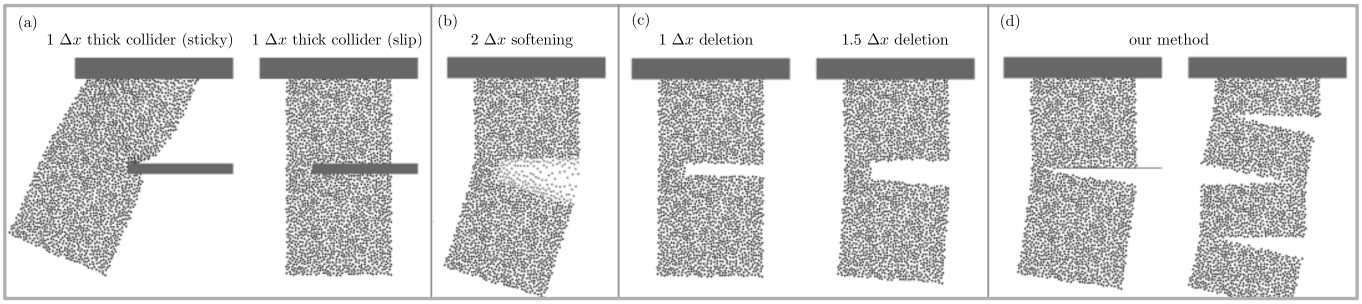

Fig. 2. (a) Traditional level set collision objects cannot cut the elastic object even if they only cover one layer of grid nodes; (b) Plastic softening allows material separation, but introduces a lot of visually unappealing damaged debris; (c) Particle deletion either does not work or causes too much volume loss; (d) Our method successfully handles both progressive and instant cutting.

fluids with arbitrarily thin solid boundary obstacles. They also addressed the treatment of thin gaps between multiple objects. Zarifi et al. [2017] developed positive-definite cut-cell method for strong coupling between elastic objects and incompressible fluids.

Particle-based methods are also popular in fluid simulation. Traditional Smoothed Particle Hydrodynamics (SPH) [Müller et al. 2003] provides very limited control over solid boundaries. Becker et al. [2009] achieved two-way coupling of compressible SPH and rigid bodies by sampling boundary particles on rigid bodies and using a predictor-corrector scheme to compute forces on particles. Akinci et al. [2012] also sampled particles on rigid boundaries, but proposed a more versatile method that handles pressure and friction directly with hydrodynamic forces. Their approach works well for thin shell rigid bodies. Koschier et al. [2017] recently proposed density maps for SPH boundaries. Using precomputed density maps, their approach eliminated the need for sampling rigid boundary particles. Macklin et al. [2013] proposed position-based fluids under the position-based dynamics (PBD) framework [Müller et al. 2007] where collisions against boundaries are formulated as non-penetration constraints. The approach is further extended in [Macklin et al. 2014] to allow solid-fluid coupling through density constraints.

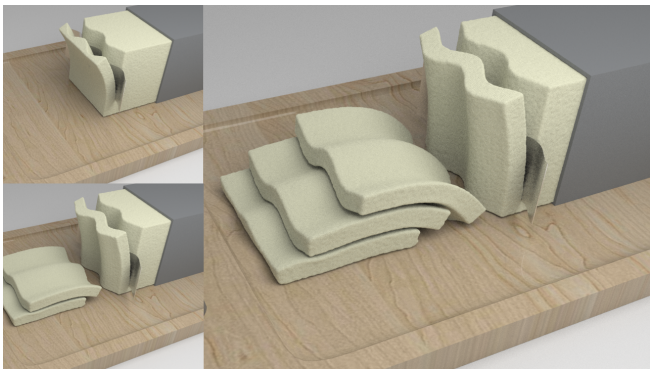

Fig. 3. Cutting cheese with a wavy knife shows appealing peeling behavior.

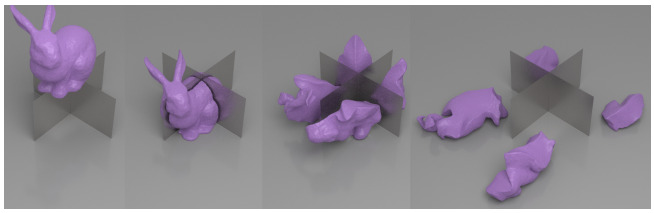

Fig. 4. An elastic bunny is split by two intersecting thin plates.

2.3 Material Point Method

MPM [Sulsky et al. 1995] is a hybrid Lagrangian/ Eulerian discretization scheme for solid mechanics. It is also recognized as a generalization of the FLIP [Brackbill and Ruppel 1986] method, which is widely used for liquid animation [Zhu and Bridson 2005]. More recently, MPM has been applied to various computer graphics applications including snow [Stomakhin et al. 2013], sand [Daviet and Bertails-Descoubes 2016; Klár et al. 2016], foam [Ram et al. 2015; Yue et al. 2015], cloth [Jiang et al. 2017a], and solid-fluid mixture [Stomakhin et al. 2014; Tampubolon et al. 2017]. Notably, Daviet et al. [2016] presented a semi-implicit frictional boundary condition for coupling MPM sand with rigid bodies. Due to the adoption of a single velocity field, it remains challenging to separate continuum materials in MPM. Wretborn et al. [2017] animated MPM crack propagation by gluing multiple grids together. It assumes pre-fracturing and purely elastic materials. In engineering literature, MPM material discontinuity is sometimes achieved by explicit front tracking [Nairn 2003], which performs multiple ray intersection tests per particle across an explicit mesh. Most other approaches achieve material separation through strain softening or material damaging [Banerjee et al. 2012]. This is similar to element deletion in FEM, and causes mesh dependent volume loss as well as undesirable debris. Gao et al. [2017] developed spatially adaptive MPM. They resolved thin features by locally refining the computational grid and particles. Moutsanidis et al. [2018] modeled strong discontinuities in MPM using a single velocity field. They locally modify interpolation functions near discontinuity.

2.4 Moving Least Squares

As a local fitting scheme, Moving Least Squares (MLS) has been widely adopted in computer graphics, including for image deformation [Kanamori et al. 2011; Schaefer et al. 2006], surface reconstruction [Lancaster and Salkauskas 1981; Levin 2004], particle-based simulation [Band et al. 2017; Müller et al. 2004], and data compression [Langlois et al. 2014]. We refer to Levin's book [Levin 1998] for a more detailed introduction to MLS.

MLS is most popular in the areas of deformation and surface reconstruction. The pioneering work of Schaefer et al. [2006] derived a closed-form expression for least-squares image deformation. Kanamori et al. [2011] extended this idea to reduce distortion in wide-angle images. Zhu et al. [2007] generalized it to 3D deformation problems. Sato et al. [2014] proposed a relevant method for deforming fluid flow fields based on physical laws. As for surface reconstruction, Levin et al. [2004] introduced a MLS projection procedure for constructing smooth surfaces from potentially noisy point cloud data. Fleishman et al. [2005] augmented this algorithm with robust statistical tools, yielding piecewise smooth surfaces.

MLS has also been applied to particle-based simulations, especially for interpolating continuous functions on sampled particles. Müller et al. [2004] provided a local MLS approximation to the gradient of the displacement field for evaluating stress, strain and other mechanical values. Pauly et al. [2005] extended their work to modeling fracture surfaces. Martin et al. [2010] used Generalized Moving Least Squares (GMLS) for discretizing the displacement field in elastica. More recently, Band et al. [2017] embedded MLS into SPH boundary handling. Their method allows particles to slip along boundaries without any distortion. Plus, MLS is also powerful in data compression. For example Langlois et al. [2014] presented an eigenmode compression of modal sound based on non-linear optimization of MLS.

MLS is a core idea behind meshless methods such as the element-free Galerkin (EFG) method [Belytschko et al. 1994; Huerta et al. 2004] and the Reproducing Kernel Particle Method (RKPM) [Liu et al. 1995]. In §3 we review the main idea of MLS function reconstruction, and derive APIC, PolyPIC and MLS-MPM from this point of view.

3 MOVING LEAST SQUARES MPM

In this section we derive MLS-MPM as a new spatial discretization that unifies APIC, PolyPIC and force computation consistently with the weak form of the momentum equation. Interestingly, our derivation shows that MPM, although seemingly quite different from purely Lagrangian meshless methods, can be treated as a modified element-free Galerkin (EFG) method [Belytschko et al. 1994], where the background Eulerian grid merely acts as a helper structure for accelerating MLS interpolation from particle neighbor regions.

We use subscript i to denote grid node quantities and p to denote particle quantities. We provide a list of important notations used in this section in Table 1.

3.1 Discrete MLS in element-free Galerkin

We start with reviewing MLS in purely meshless methods such as element-free Galerkin (EFG) [Belytschko et al. 1994]; see [Huerta et al. 2004] for more details.

Variable	Type	Meaning
u	any	any continuous function approximated with MLS
$\mathbf{x}_i$	vector	the location of sample/node i
$\mathbf{x}_p$	vector	the location of particle p
$\mathbf{z}$, $\mathbf{x}$	vector	an arbitrary continuous location
$\mathbf{P}(\mathbf{x})$	vector	the polynomial basis
$\mathbf{c}(\mathbf{x})$	vector	all basis coefficients
$\mathbf{M}(\mathbf{x})$	matrix	the moment matrix
$\mathbf{M}_p$	matrix	$\mathbf{M}(\mathbf{x}_p)$
$\xi_i(\mathbf{x})$	scalar	weighting function centered at $\mathbf{x}_i$
$\Phi_i(\mathbf{x})$	scalar	MLS shape function centered at $\mathbf{x}_i$
$N_i(\mathbf{x})$	scalar	B-spline basis function centered at $\mathbf{x}_i$
$\rho(\mathbf{x}, t)$	vector	the continuous density field
$\mathbf{v}(\mathbf{x})$	vector	the continuous velocity field
m_i	scalar	mass of node i
$\mathbf{v}_i$	vector	velocity of node i
$\mathbf{v}_i^n$	vector	velocity of node i at time n over domain Ω^{t^n}
$\hat{\mathbf{v}}_i^n$	vector	velocity of node i at time $n+1$ over domain Ω^{t^n}
m_p	scalar	mass of particle p
$\mathbf{v}_p$	vector	velocity of particle p
$\mathbf{C}_p$	matrix	affine matrix of particle p
$\mathbf{q}$	vector	test function in the weak form
$q_{\alpha,\beta}$	scalar	derivative of q_α wrt. x_β
$\boldsymbol{\sigma}$	matrix	Cauchy stress
$\mathbf{F}_p$	matrix	the deformation gradient on particle p
$\mathbf{f}_i$	vector	the force on grid node i

Table 1. Important notations used in the MLS-MPM derivation (§3).

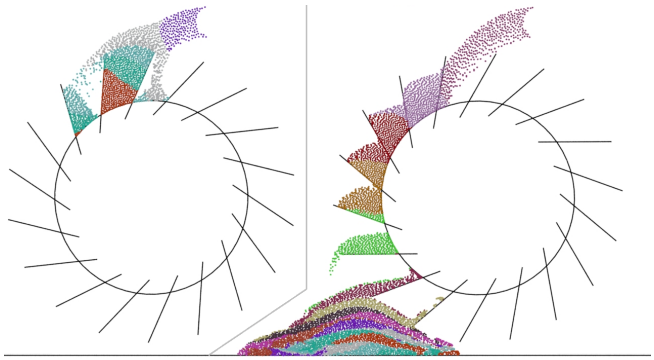

Fig. 5. Two dimensional sand inflow is two-way coupled with a wheel. The wheel is made of intersecting thin boundaries, allowing clean separation for materials on different sides and corners.

Suppose one is given a set of samples at some locations $\mathbf{x}_i$ for a continuous function $u_i = u(\mathbf{x}_i)$, the idea behind MLS [Lancaster

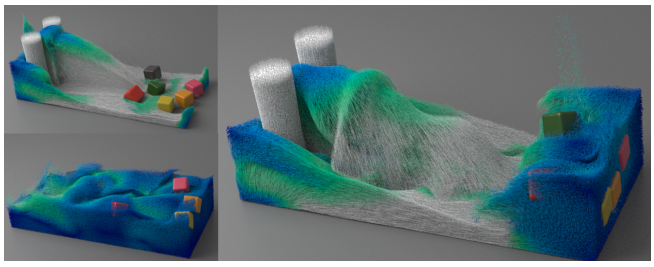

Fig. 6. Our method enables two-way coupled simulation of splashing water and rigid blocks with different densities.

and Salkauskas 1981] is that for a fixed $\mathbf{x}$, one can approximate u at any location $\mathbf{z}$ in the continuous space near $\mathbf{x}$ using a polynomial least-squares fit of u from the samples in this local region with $u(\mathbf{z}) = \mathbf{P}^T(\mathbf{z})\mathbf{c}(\mathbf{x})$, where $\mathbf{P}(\mathbf{z}) = [p_0(\mathbf{z}), p_1(\mathbf{z}), \ldots, p_l(\mathbf{z})]^T$ forms an l dimensional subspace of polynomials of degree m, and $\mathbf{c}(\mathbf{x}) = [c_0(\mathbf{x}), c_1(\mathbf{x}), \ldots, c_l(\mathbf{x})]^T$ are the basis coefficients. In practice, to avoid numerical instabilities caused by large entries in the moment matrix (see $\mathbf{M}$ below), the polynomial basis can be re-centered around a fixed point $\mathbf{x}$ by replacing $\mathbf{P}^T(\mathbf{z})$ with $\mathbf{P}^T(\mathbf{z}-\mathbf{x})$ (see e.g. [Liu et al. 1995]), leading to

$$u(\mathbf{z}) = \mathbf{P}^T(\mathbf{z}-\mathbf{x})\mathbf{c}(\mathbf{x}). \tag{1}$$

In EFG, $\mathbf{c}(\mathbf{x})$ is evaluated using weighted least squares that minimizes the functional $J_{\mathbf{x}}(\mathbf{c}) = \sum_{i\in B_{\mathbf{x}}} \xi_i(\mathbf{x}) \left(\mathbf{P}^T(\mathbf{x}_i-\mathbf{x})\mathbf{c}(\mathbf{x}) - u_i\right)^2$, where $\xi_i(\mathbf{x})$ is a localized weighting function centered at $\mathbf{x}_i$, and $B_{\mathbf{x}}$ denotes the set of indices satisfying $\xi_i(\mathbf{x}) \neq 0$. The solution is

$$\mathbf{c}(\mathbf{x}) = \mathbf{M}^{-1}(\mathbf{x})\mathbf{b}(\mathbf{x}), \tag{2}$$

where $\mathbf{b}(\mathbf{x}) = \sum_{i\in B_{\mathbf{x}}} \xi_i(\mathbf{x})\mathbf{P}(\mathbf{x}_i-\mathbf{x})u_i$ and $\mathbf{M}(\mathbf{x}) = \sum_{i\in B_{\mathbf{x}}} \xi_i(\mathbf{x})\mathbf{P}(\mathbf{x}_i-\mathbf{x})\mathbf{P}^T(\mathbf{x}_i-\mathbf{x})$. Note that when $\mathbf{P}$ only contains linear polynomials, a more intuitive form of $\mathbf{c}(\mathbf{x})$ is given in §3.1.1.

Substituting Eq. 2 back into Eq. 1 gives

$$u(\mathbf{z}) = \sum_{i\in B_{\mathbf{x}}} \xi_i(\mathbf{x})\mathbf{P}^T(\mathbf{z}-\mathbf{x})\mathbf{M}^{-1}(\mathbf{x})\mathbf{P}(\mathbf{x}_i-\mathbf{x})u_i, \tag{3}$$

which can also be expressed as $u(\mathbf{z}) = \sum_{i\in B_{\mathbf{x}}} \Phi_i(\mathbf{z})u_i$, where $\Phi_i(\mathbf{z}) = \xi_i(\mathbf{x})\mathbf{P}^T(\mathbf{z}-\mathbf{x})\mathbf{M}^{-1}(\mathbf{x})\mathbf{P}(\mathbf{x}_i-\mathbf{x})$ can be defined as the nodal shape function of $\mathbf{x}_i$ in EFG. Interestingly this is exactly the shape function used in the Reproducing Kernel Particle Method (RKPM) [Liu et al. 1995].

The polynomial subspace is usually composed of monomials up to degree m. In 2D this corresponds to $\mathbf{P}(\mathbf{x}) = [1]^T$ for the constant basis, $\mathbf{P}(\mathbf{x}) = [1, x, y]^T$ for linear basis, and $\mathbf{P}(\mathbf{x}) = [1, x, y, xy, x^2, y^2]$ for quadratic basis. Since the constant function is always a basis, we automatically have partition of unity, i.e. $\sum_i \Phi_i(\mathbf{x}) = 1$. With a complete degree-m polynomial basis ($l = m$), MLS is m-order-consistent† and reproduces all polynomials in $\mathbf{P}$ [Huerta et al. 2004]. Additionally, if $\xi_i(\mathbf{x})$ is of class C^k, then $\Phi_i(\mathbf{x})$ is of $C^{\min(k,m)}$.

3.1.1 The case of a linear polynomial basis. In the simple case of a complete linear polynomial basis ($m = l = 1$) as done by Müller et al.[2004] for meshless solids, MLS is a 1st-order-consistent interpolation scheme for scattered data and derivatives. With $\mathbf{P}(\mathbf{x}_i - \mathbf{x}) = [1, (\mathbf{x}_i - \mathbf{x})^T]^T$, Eq. 2 gives the reconstructed function value and its gradient estimation at $\mathbf{x}$:

$$\begin{bmatrix} \hat{u} \\ \nabla\hat{u} \end{bmatrix} = \mathbf{M}^{-1}(\mathbf{x})\mathbf{Q}^T\Xi(\mathbf{x}) \begin{bmatrix} u_1 \\ \vdots \\ u_N \end{bmatrix}, \tag{4}$$

where N is the total number of sample data points, $\Xi(\mathbf{x})$ is the diagonal weighting matrix with $\Xi_{ii} = \xi_i(\mathbf{x})$, and $\mathbf{Q}(\mathbf{x}) = [\mathbf{P}(\mathbf{x}_1 - \mathbf{x}), \ldots, \mathbf{P}(\mathbf{x}_N - \mathbf{x})]^T$, $\mathbf{M}(\mathbf{x}) = \mathbf{Q}^T\Xi\mathbf{Q}$.

† If the approximation reproduces exactly a basis of the polynomials of degree less than or equal to m, then the approximation is said to have m-order consistency [Huerta et al. 2004].

3.2 Equivalence of APIC/PolyPIC and MLS on velocities

MPM discretizes the governing equations using interpolation functions on the grid as shape functions and particles as quadrature points. Each particle (subscripted with p) has mass m_p, position $\mathbf{x}_p$, velocity $\mathbf{v}_p$, deformation gradient $\mathbf{F}_p$ and other parameters related to its material constitutive model. A grid (subscripted with i) acts as a scratch pad and stores mass m_i and velocity $\mathbf{v}_i$. In each time step, particles transfer mass and velocity to the grid. Grid velocity is then integrated over time and transferred back to particles.

APIC [Jiang et al. 2015] and PolyPIC [Fu et al. 2017] are equivalent to applying MLS to velocity $\mathbf{v}(\mathbf{x})$ with B-splines as the weighting function $\xi_i(\mathbf{x})$. We give a detailed discussion in the supplementary document [Hu et al. 2018]. Unlike EFG or RKPM where reconstruction and data sample locations are colocated, MPM uses Cartesian lattice nodes for data samples without particle neighbor search.

3.3 MLS-MPM as a special EFG discretization

In this section we derive the MLS-MPM discretization from the continuous weak form of the governing equations. Implicit summation convention on indices is assumed.

3.3.1 Governing equations. We start with the Eulerian governing equations:

$$\frac{D\rho}{Dt} + \rho\nabla\cdot\mathbf{v} = 0 \quad \text{(conservation of mass)}, \tag{5}$$

$$\rho\frac{D\mathbf{v}}{Dt} = \nabla\cdot\boldsymbol{\sigma} + \rho\mathbf{g} \quad \text{(conservation of momentum)}, \tag{6}$$

where ρ is mass density, $\mathbf{v}$ is velocity, $\mathbf{g} = (0, -9.8, 0)^T$ is gravity, $\boldsymbol{\sigma}$ is Cauchy stress, and $\frac{D\phi}{Dt} = \frac{\partial\phi}{\partial t} + \mathbf{v}\cdot\nabla\phi$ denotes the material derivative of any function $\phi(\mathbf{x}, t)$.

3.3.2 Weak form. As in standard Finite Element Methods [Hughes 2012], the weak formulation of the governing PDE involves multiplying the differential equation by a test function, integrating by parts, and applying boundary conditions.

Denoting the material domain at time t^n with Ω^{t^n}, an *updated Lagrangian* time discretization of the weak form of Eq. 6 following [Jiang et al. 2016] leads to (we drop $\mathbf{g}$ here for simplicity)

$$\begin{aligned} &\frac{1}{\Delta t}\int_{\Omega^{t^n}} \rho(\mathbf{x}, t^n)\left(\hat{v}_\alpha^{n+1}(\mathbf{x}) - v_\alpha^n(\mathbf{x})\right) q_\alpha(\mathbf{x}, t^n)d\mathbf{x} \\ &= \int_{\partial\Omega^{t^n}} q_\alpha(\mathbf{x}, t^n)\mathcal{T}_\alpha(\mathbf{x}, t^n)ds - \int_{\Omega^{t^n}} q_{\alpha,\beta}(\mathbf{x}, t^n)\sigma_{\alpha\beta}(\mathbf{x}, t^n)d\mathbf{x}, \end{aligned} \tag{7}$$

where $\mathbf{q}(\cdot, t) : \Omega^{t^n} \to \mathbb{R}^d$ is an arbitrary vector-valued test function that vanishes at the Dirichlet boundary $\partial\Omega_D$, $d = 2$ or 3 is the problem dimension, $\mathcal{T}(\mathbf{x}, t)$ is the traction field along the boundary. Here we have used $\mathbf{v}^n$ to denote the current Eulerian velocity at time n. $\hat{\mathbf{v}}^{n+1}$ denotes the updated velocity field after forces are applied. Both $\mathbf{v}^n$ and $\hat{\mathbf{v}}^{n+1}$ are defined for $\mathbf{x} \in \Omega^{t^n}$ as $\Omega^{t^n} \to \mathbb{R}^d$. Notice that we choose the notation $\hat{\mathbf{v}}^{n+1}$ instead of $\mathbf{v}^{n+1}$, since $\mathbf{v}^{n+1}$ is only defined on the domain of the next time step $\Omega^{t^{n+1}}$. The weak form is also expressed using index notations, where $v_\alpha^n, q_\alpha, \mathcal{T}_\alpha$ are α components of $\mathbf{v}^n, \mathbf{q}, \mathcal{T}$ and $q_{\alpha,\beta} = \frac{\partial q_\alpha}{\partial x_\beta}$. Implicit summation on

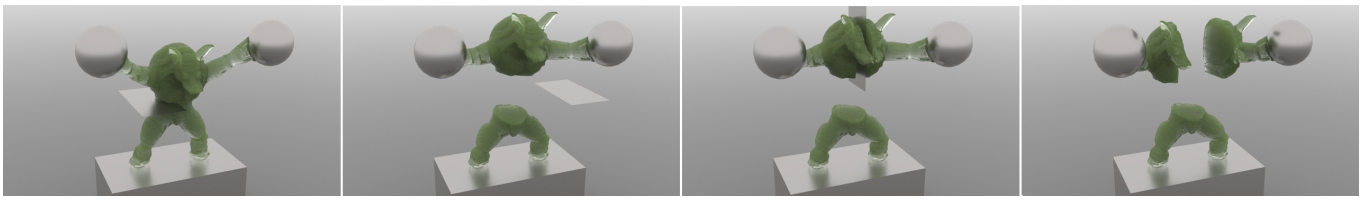

Fig. 7. We dissect an initially stretched elastic armadillo with two progressive cuts.

$\alpha, \beta = 1, \ldots, d$ is also assumed, where d is the problem dimension (2 or 3).

Typically in FEM, both the unknown variable and the test function can be approximated by functions in a finite-dimensional function space as linear combinations of some basis shape functions. The main difference between traditional MPM and MLS-MPM is the choice of this function space. Traditional MPM uses B-spline basis functions while MLS-MPM uses MLS shape functions ($\Phi_i(\mathbf{x})$ in §3.1). This choice is the key contribution of MLS-MPM and will be shown to provide important advantages.

3.3.3 Traditional MPM discretization. MPM discretizes all spatial terms using B-spline grid basis functions $N_i(\mathbf{x})$ as (with implicit summation): $q_\alpha(\mathbf{x}, t^n) = N_i(\mathbf{x}) q_{i\alpha}^n$, $v_\alpha^n(\mathbf{x}) = N_j(\mathbf{x}) v_{j\alpha}^n$, and $\hat{v}_\alpha^{n+1}(\mathbf{x}, t^n) = N_j(\mathbf{x}) \hat{v}_{j\alpha}^{n+1}$. This further induces the lumped mass formulation $m_i^n = \sum_p N_i(\mathbf{x}_p^n) m_p$ (see [Jiang et al. 2016] for a detailed derivation).

3.3.4 MLS-MPM momentum term. In this section we show how MLS-MPM discretizes the left hand side of the weak form Eq. 7. We first divide the continuum domain with particle partitions $\Omega_p^{t^n}$ as

$$\int_{\Omega^{t^n}} \rho(\mathbf{x}, t^n) v_\alpha^n(\mathbf{x}) q_\alpha(\mathbf{x}, t^n) d\mathbf{x} = \sum_p \int_{\Omega_p^{t^n}} \rho(\mathbf{x}, t^n) v_\alpha^n(\mathbf{x}) q_\alpha(\mathbf{x}, t^n) d\mathbf{x}.$$

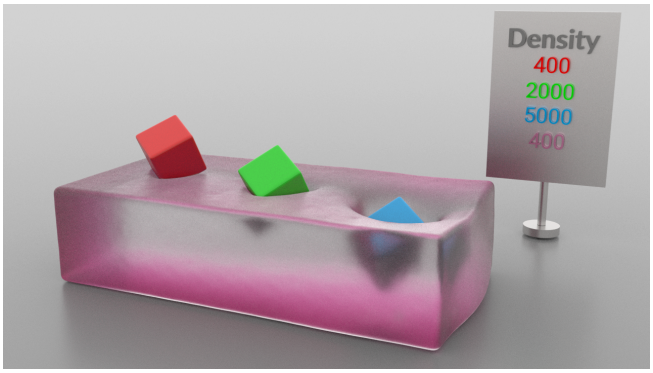

Fig. 8. An elastoplastic von-Mises Jello block is two-way coupled with rigid blocks with different density ratios.

In each integral over $\Omega_p^{t^n}$ since $\mathbf{x}$ is near $\mathbf{x}_p^n$, we can approximate the continuous equations with nodal data samples

$$v_\alpha^n(\mathbf{x}) = \sum_j \Phi_j(\mathbf{x}) v_{j\alpha}^n \tag{8}$$

and

$$q_\alpha(\mathbf{x}, t^n) = \sum_i \Phi_i(\mathbf{x}) q_{i\alpha}^n, \tag{9}$$

where we used the MLS shape function (§3.1)

$$\Phi_i(\mathbf{x}) = \xi_i(\mathbf{x}_p^n) \mathbf{P}^T(\mathbf{x} - \mathbf{x}_p^n) \mathbf{M}^{-1}(\mathbf{x}_p^n) \mathbf{P}(\mathbf{x}_i - \mathbf{x}_p^n). \tag{10}$$

Therefore

$$\sum_p \int_{\Omega_p^{t^n}} \rho(\mathbf{x}, t^n) v_\alpha^n(\mathbf{x}) q_\alpha(\mathbf{x}, t^n) d\mathbf{x} = \sum_{p,i,j} q_{i\alpha}^n v_{j\alpha}^n m_{ij}, \tag{11}$$

where $m_{ij} = \int_{\Omega_p^{t^n}} \rho(\mathbf{x}, t^n) \Phi_i(\mathbf{x}) \Phi_j(\mathbf{x}) d\mathbf{x}$ is the mass matrix. Mass lumping further approximates it with a diagonal matrix by summing each row. The diagonal entry is

$$m_i^n = \sum_p \int_{\Omega_p^{t^n}} \rho(\mathbf{x}, t^n) \Phi_i(\mathbf{x}) d\mathbf{x} \approx \sum_p m_p \Phi_i(\mathbf{x}_p^n) = \sum_p m_p N_i(\mathbf{x}_p^n),$$

which is consistent with traditional MPM. See [Jiang et al. 2016] for further derivations which are applicable to MLS-MPM as well.

The grid velocity is evolved from $\mathbf{v}_i^n$ to $\hat{\mathbf{v}}_i^{n+1}$. Intuitively, we can approximate time t^{n+1} velocities around time t^n particle locations $\mathbf{x}_p^n$ using MLS expression $\hat{v}_\alpha^{n+1}(\mathbf{x}) = \mathbf{P}^T(\mathbf{x} - \mathbf{x}_p^n) \mathbf{c}_{\hat{v}_\alpha^{n+1}}(\mathbf{x}_p^n)$, where the subscript in $\mathbf{c}$ denotes the reconstructed physical quantity. As explained in §3.2, evaluating $\mathbf{c}_{\hat{v}_\alpha^{n+1}}$ corresponds to the grid-to-particle transfer in APIC/PolyPIC.

3.3.5 MLS-MPM stress term. The key contribution of MLS-MPM is on the stress term, i.e., the right hand side of Eq. 7 without the boundary traction term. Note that the boundary integral evaluates to 0 assuming a zero Neumann boundary condition (no prescribed traction at the boundary).

Choose the test function. Similarly to the momentum term, we can express the stress integral through the summation over individual particle domains:

$$-\int_{\Omega^{t^n}} q_{\alpha,\beta}(\mathbf{x}, t^n) \sigma_{\alpha\beta}(\mathbf{x}, t^n) d\mathbf{x} = -\sum_p \int_{\Omega_p^{t^n}} q_{\alpha,\beta}(\mathbf{x}, t^n) \sigma_{\alpha\beta}(\mathbf{x}, t^n) d\mathbf{x}. \tag{12}$$

Recall in Eq. 9, we have chosen to express the test function $\mathbf{q}(\mathbf{x}, t)$ from a finite-dimensional function space (the discretized test space).

This allows us to apply standard Finite Element procedures [Hughes 2012] to convert Eq. 12 into a system of equations by letting $\mathbf{q}$ be, in turn, each of the basis functions in the test space. The resulting system contains $N_g d$ equations for all the degrees of freedom, where N_g is the total number of grid nodes and d is the problem dimension. For the degree of freedom corresponding to any grid node $\hat{i} \in \{1, \ldots, N_g\}$ and component $\hat{\alpha} \in \{1, \ldots, d\}$, we can enforce such a choice of $\mathbf{q}$ by setting

$$q^n_{i\alpha} = \delta_{i\hat{i}}\delta_{\alpha\hat{\alpha}} = \begin{cases} 1 & \text{if } \alpha = \hat{\alpha} \text{ and } i = \hat{i} \\ 0 & \text{otherwise} \end{cases}$$

in Eq. 9. Combining this with the MLS shape function from Eq. 10 leads to

$$q_\alpha(\mathbf{x}, t^n) = \mathbf{P}^T(\mathbf{x} - \mathbf{x}^n_p)\mathbf{M}^{-1}(\mathbf{x}^n_p)\xi_{\hat{i}}(\mathbf{x}^n_p)\mathbf{P}(\mathbf{x}_{\hat{i}} - \mathbf{x}^n_p)\delta_{\alpha\hat{\alpha}} \tag{13}$$

for any $\mathbf{x} \in \Omega^{t^n}_p$. Such test functions will be enumerated over all $\hat{i}$ and $\hat{\alpha}$ to get the resulting force components $f_{\hat{i}\hat{\alpha}}$ associated with all degrees of freedom on the grid.

Discretizing the force. To reach the discrete force, Eq. 12 requires the derivative of $\mathbf{q}$. Differentiating Eq. 13 gives

$$q_{\alpha,\beta}(\mathbf{x}, t^n) = \frac{\partial \mathbf{P}^T(\mathbf{x} - \mathbf{x}^n_p)}{\partial x_\beta}\mathbf{M}^{-1}(\mathbf{x}^n_p)\xi_{\hat{i}}(\mathbf{x}^n_p)\mathbf{P}(\mathbf{x}_{\hat{i}} - \mathbf{x}^n_p)\delta_{\alpha\hat{\alpha}}. \tag{14}$$

To simplify the derivation, we adopt the linear polynomial space $\mathbf{P}^T(\mathbf{x} - \mathbf{x}^n_p) = [1, x_1 - x^n_{p1}, x_2 - x^n_{p2}, x_3 - x^n_{p3}]$. Note that it is possible to generalize this choice to a higher order polynomial space, and we leave such an extension to future work. We also choose $\xi_{\hat{i}} = N_{\hat{i}}$ to be quadratic/cubic B-splines (so that $\mathbf{M}^{-1}$ is a constant). Under these assumptions, Eq. 14 becomes

$$q_{\alpha,\beta}(\mathbf{x}, t^n) = M_p^{-1}N_i(\mathbf{x}^n_p)(x_{\hat{i}\beta} - x_{p\beta})\delta_{\alpha\hat{\alpha}}, \tag{15}$$

where $M_p = \frac{1}{4}\Delta x^2$ for quadratic $N_i(\mathbf{x})$ and $\frac{1}{3}\Delta x^2$ for cubic $N_i(\mathbf{x})$.

Substituting it back into Eq. 12 reveals the $\hat{\alpha}$ component force computation on grid node $\hat{i}$:

$$\begin{aligned} f_{\hat{i}\hat{\alpha}} &= -\sum_p \int_{\Omega^{t^n}_p} q_{\alpha,\beta}(\mathbf{x}, t^n)\sigma_{\alpha\beta}(\mathbf{x}, t^n)d\mathbf{x} \\ &\approx -\sum_p V^n_p M_p^{-1}\sigma^n_{p\hat{\alpha}\beta}N_{\hat{i}}(\mathbf{x}^n_p)(x^n_{\hat{i}\beta} - x^n_{p\beta}), \end{aligned} \tag{16}$$

where V^n_p is the current volume of particle p at time n. Here the approximation comes from adopting a one-point quadrature rule to replace $\boldsymbol{\sigma}(\mathbf{x}, t^n)$ in $\Omega^{t^n}_p$ with $\boldsymbol{\sigma}^n_p$.

Note that in contrast to our result, traditional MPM uses $f_{\hat{i}\hat{\alpha}} = -\sum_p V^n_p\sigma^n_{p\hat{\alpha}\beta}N_{\hat{i},\beta}(\mathbf{x}^n_p)$ which requires explicitly differentiating the interpolation function $N_{\hat{i}}(\mathbf{x})$.

3.4 Deformation gradient and force

Deformation gradient $\mathbf{F} = \frac{\partial \mathcal{Z}}{\partial \mathbf{X}}$ is usually used to characterize finite deformation in elastoplasticity, where $\mathbf{X}$ denotes the material space, and $\mathcal{Z}(\mathbf{X}, t)$ is the deformation map. In MPM, $\mathbf{F}$ is evolved on each material particle with $\frac{\partial}{\partial t}\mathbf{F}(\mathbf{X}, t) = \frac{\partial \mathbf{v}}{\partial \mathbf{x}}(\mathcal{Z}(\mathbf{X}, t), t)\mathbf{F}(\mathbf{X}, t)$, where Eulerian velocity gradient $\frac{\partial \mathbf{v}}{\partial \mathbf{x}}$ is discretized on the grid. Based on the updated Lagrangian view, particle-wise $\mathbf{F}_p$ is updated as $\mathbf{F}^{n+1}_p = \left(\mathbf{I} + \Delta t\frac{\partial \hat{\mathbf{v}}^{n+1}}{\partial \mathbf{x}}(\mathbf{x}^n_p)\right)\mathbf{F}^n_p$, where traditional MPM uses $\frac{\partial \hat{\mathbf{v}}^{n+1}}{\partial \mathbf{x}}(\mathbf{x}^n_p) = \sum_i \hat{\mathbf{v}}^{n+1}_i\nabla N_i(\mathbf{x}^n_p)^T$. In the MLS view of $\mathbf{v}(\mathbf{x})$ we can differentiate Eq. 8. For linear polynomials this leads to

$$\frac{\partial \hat{\mathbf{v}}^{n+1}}{\partial \mathbf{x}} = \mathbf{C}^{n+1}_p \quad \text{and} \quad \mathbf{F}^{n+1}_p = \left(\mathbf{I} + \Delta t\mathbf{C}^{n+1}_p\right)\mathbf{F}^n_p, \tag{17}$$

where $\mathbf{C}^{n+1}_p$ is exactly the affine velocity matrix from APIC. Accordingly if we assume hyperelasticity with total potential energy $E = \sum_p V^0_p\Psi_p(\mathbf{F}_p)$ where V^0_p is particle initial volume and Ψ_p is the energy density function, it can be shown that

$$\mathbf{f}_i = -\frac{\partial E}{\partial \mathbf{x}_i} = -\sum_p N_i(\mathbf{x}^n_p)V^0_p M_p^{-1}\frac{\partial \Psi}{\partial \mathbf{F}}(\mathbf{F}^n_p)\mathbf{F}^{n\,T}_p(\mathbf{x}^n_i - \mathbf{x}^n_p), \tag{18}$$

which is consistent with the weak form result from Eq. 16 by noticing $\boldsymbol{\sigma} = \frac{1}{\det(F)}\frac{\partial \Psi}{\partial \mathbf{F}}\mathbf{F}^T$ and $\det(\mathbf{F})V^0_p = V^n_p$.

In contrast to traditional MPM, the MLS-MPM deformation gradient update and force computation directly re-use quantities from APIC and do not require any evaluation of the interpolation function gradient throughout the algorithm. This greatly simplifies the implementation of MPM and substantially decreases the computational cost (see §6 for more details).

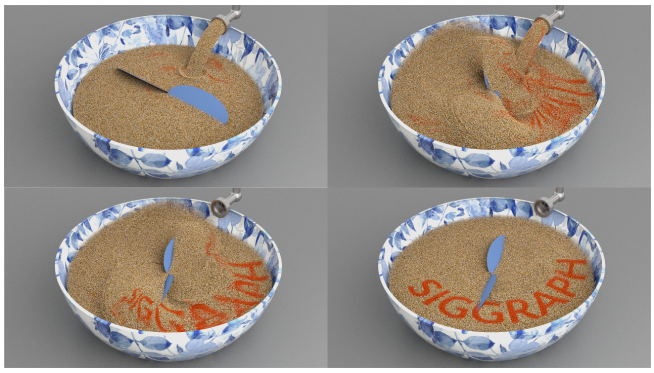

Fig. 9. We stir a bowl of dry sand with two thin plates. Materials from opposite sides experience independent dynamics even with the use of a single shared background grid.

3.5 Implicit Integration

As in [Stomakhin et al. 2013], implicit time stepping is naturally supported by MLS-MPM. Implicit MPM with Newton's method [Gast et al. 2015] requires computing the action of the energy Hessian on an arbitrary grid increment $\delta\mathbf{u}$. We show in the supplementary document [Hu et al. 2018] that

$$-\delta\mathbf{f}_i = \sum_p V^0_p\mathbf{A}_p\mathbf{F}^{n\,T}_p M_p^{-1}N_i(\mathbf{x}^n_p)(\mathbf{x}^n_i - \mathbf{x}^n_p), \tag{19}$$

where $\mathbf{A}_p = \frac{\partial^2 \Psi}{\partial \mathbf{F}\partial \mathbf{F}} : \sum_j M_p^{-1}N_j(\mathbf{x}^n_p)\delta\mathbf{u}_j(\mathbf{x}^n_j - \mathbf{x}^n_p)^T\mathbf{F}^n_p$. In practice it corresponds to a grid-to-particle gather step (for computing $\mathbf{A}_p$) and a particle-to-grid scatter step (for accumulating $\delta\mathbf{f}_i$).

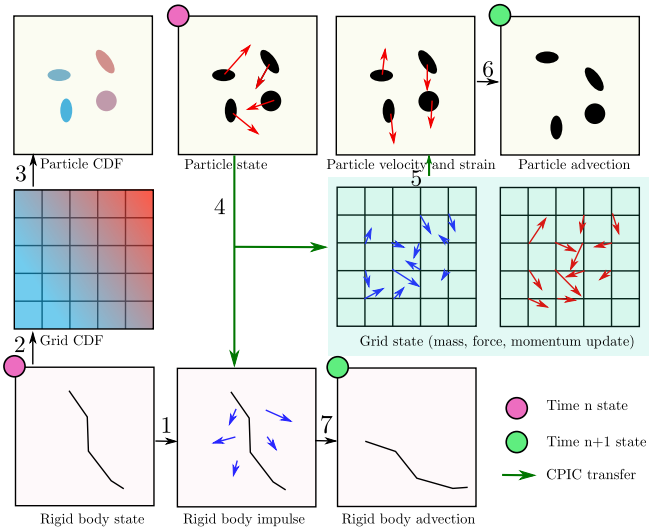

Fig. 10. Algorithm overview from time t^n to t^{n+1} for MLS-MPM with CPIC. Steps: (1) Rigid-rigid collision and rigid body articulation update rigid body velocities (§5.1); (2) Splat rigid body to grid CDF (§5.2); (3) Reconstruct particle CDF from grid CDF (§5.3); (4) CPIC particle-to-grid transfer and rigid body impulses (§5.4); (5) CPIC grid-to-particle transfer (§5.5); (6) MPM particle advection (§5.5); (7) Rigid body advection (§5.6).

4 FROM MPM TO MLS-MPM

Before introducing additional considerations for material discontinuity, here we summarize the essential steps in MLS-MPM, since it can be independently used for modifying any existing MPM framework.

(1) **Particles to grid.** Use APIC [Jiang et al. 2015] or PolyPIC [Fu et al. 2017] to transfer mass and momentum from the particles to the grid.
(2) **Update grid momentum.** Use either symplectic Euler (with force given by Eq. 18) or backward Euler (with force differential given by Eq. 19) to update grid momentum.
(3) **Grid to particles.** Use APIC or PolyPIC to transfer velocities and affine/polynomial coefficients from the grid to the particles.
(4) **Particle deformation gradient.** Update particle deformation gradient using the MLS approximation to the velocity gradient (Eq. 17).
(5) **Update particle plasticity.** Project particle deformation gradient for plasticity (if there is any).
(6) **Particle advection.** Particle positions are updated with their new velocities.

The only differences between MLS-MPM and traditional MPM are the force expression in step (2) and the **F** update in step (4). In fact, step (4) in MLS-MPM is simpler than MPM due to the reuse of $\mathbf{C}_p^{n+1}$ constructed in step (3). Step (2) in MLS-MPM is also easier to implement than MPM, and allows an easy-to-achieve performance gain as discussed in §6.

5 METHOD: MLS-MPM WITH CPIC

Here we detail the steps from time t^n to t^{n+1} for MLS-MPM, enhanced with a Compatible Particle-In-Cell (CPIC) algorithm for material discontinuity and rigid-body coupling (see Fig. 10 for a diagram of the logic steps). Note that we use the term "rigid body" to denote either a dynamic rigid body or a rigid collision boundary with scripted kinematics motion. We use p, q for MPM particle indices, r, s for rigid body indices, and i, j for grid node indices.

5.1 Rigid-rigid collision

This step includes rigid-rigid collision detection/resolution and rigid body articulation. It is independent from our MPM algorithm, and any external rigid body dynamics package can be used. We will skip the details and denote the updated velocity and angular velocity of rigid body r as $\mathbf{v}_r^* \leftarrow \mathbf{v}_r^n$ and $\omega_r^* \leftarrow \omega_r^n$. Note that $\mathbf{v}_r^*$ and ω_r^* are still intermediate rigid body velocities. They will be further updated to $\mathbf{v}_r^{n+1}$ and ω_r^{n+1} at the end of time step n (see §5.6).

5.2 Splat grid-wise colored distance field (CDF)

Traditional signed distance functions (SDFs) are convenient for performing inside/outside queries and normal estimations. As such, SDFs are widely used as the implicit surface representation for volumetric collision geometries in both FEM [Irving et al. 2004] and MPM [Stomakhin et al. 2013]. Traditional SDF level sets such as *OpenVDB* [Museth 2013] can be easily constructed from closed surfaces. Lossaso et al. [2006] developed an algorithm for treating the interface of multiple level sets.

To represent intersecting open boundaries, we extend discrete SDFs to Colored Distance Fields (CDFs) with unsigned distance $d(\mathbf{x}_i)$ and *color* information. The color at each point encodes both the set of nearby surfaces and which side $\mathbf{x}_i$ locates at. As a result CDFs can discretely represent multiple regions using a single Cartesian lattice. Note that for sub-grid accuracy and more non-trivial topology such as non-manifold bifurcation, it is a better choice to construct the distance field using the algorithm by Xu and Barbič [2014], or use the non-manifold level set proposed by Mitchell et al. [2015] which stores SDF on a hexahedral mesh.

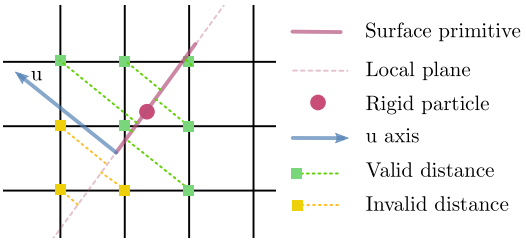

Fig. 11. Splatting the unsigned distance field from a rigid particle on a segment to 9 grid nodes. The u axis is the normal to the plane defined by the primitive. The value of u_{i,r_η} for each grid node thus represents the signed point-plane distance between grid node i and the plane that rigid particle r_η lies on. Note that such a distance is only considered to be valid (or existing) if the projection onto the plane actually lies *inside* the primitive geometry.

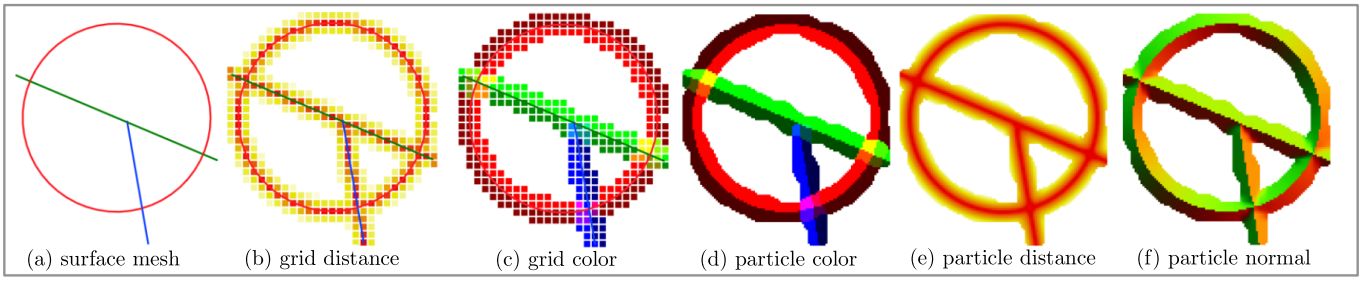

Fig. 12. (a) Three intersecting thin rigid boundaries; (b) Grid unsigned distance field; (c) Grid colors (relationship to boundaries); (d) Maintained particle color; (e)(f) Particle distances to the boundary and the normals reconstructed with MLS.

5.2.1 Grid unsigned distance. In this section we describe our fast algorithm for constructing a narrow-band unsigned distance field on the grid from rigid body surfaces.

Rigid particles. For each oriented rigid surface r, we adaptively sample auxiliary rigid particles on the mesh. We index these points as $r_1, r_2, \ldots, r_R$, where R is the total number of rigid particles on surface r. We also use $\mathcal{E}(r_\eta)$ to denote the primitive (segment in 2D and triangle in 3D) index for rigid particle r_η.

Valid distance. For computing a narrow-band distance field, we allow each rigid particle r_η to influence the closest grid node and the surrounding $3\times3\times3$ grid nodes in 3D. We can quickly project these 27 grid nodes onto the plane defined by $\mathcal{E}(r_\eta)$ and calculate the signed point-plane distance u_{i,r_η} determined by grid node $\mathbf{x}_i$ and rigid particle r_η. We illustrate this operation in Fig. 11. For efficiency, the distance is only considered *valid* and stored if the projection point lies *inside* the primitive (i.e. when point-plane distance equals point-primitive distance).

Distance rasterization. The minimum unsigned distance from $\mathbf{x}_i$ to the boundary is then

$$d_i = \min_{r,\eta} |u_{i,r_\eta}|.$$

During the process of computing all point-plane distances, we also keep track of the closest rigid body to $\mathbf{x}_i$ using index $r^*(\mathbf{x}_i)$. This index will be used in §5.4 for determining which rigid body we apply impulses on from grid velocities. Since each rigid body contains many rigid particles, we also track the rigid particle index $r_{\eta^*}(\mathbf{x}_i)$ that results in the smallest point-plane distance for node i and rigid body r. This index will be used in §5.2.2 for uniquely deciding the relative side relationship between them.

Trade-off. While there will be missing values at certain corners (which will be robustly handled as discussed in §5.3.2), this splatting process from rigid particles to grid nodes provides a very fast construction of a narrow band unsigned distance field with only point-plane projection computations. This process requires much less computation compared to the exact distance evaluation between points and meshes.

Rigid particle seeding. Note that our algorithm is not sensitive to the distribution of the rigid particles, as long as on each triangle there is at least one particle, and the whole triangle is covered by the kernel range of the particles. Specifically, we uniformly seed a lattice of particles on each triangle so that the minimum particle distance is smaller than grid spacing Δx.

5.2.2 Grid color field. The color of each grid node contains its affinity (closeness) to each rigid surface and a tag labeling the side it is on. Affinity A_{ir} for surface r and grid node i is

$$A_{ir} = \begin{cases} 1, & \exists\eta \text{ with valid } u_{i,r_\eta}, \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

Note that the validity of the signed distance u_{i,r_η} between grid node i and rigid particle r_η is defined in §5.2.1. The tag T_{ir} is determined by the signed distance of the closest rigid particle $r_{\eta^*}(\mathbf{x}_i)$, i.e., $T_{ir} = \text{sign}(u_{i,r_{\eta^*}(\mathbf{x}_i)})$.

5.2.3 Efficient CDF storage. Ideally one would like to have one CDF for each rigid body, but this is expensive in both computation and storage. Therefore, we store only one unsigned distance (32-bit float) and an extra 32-bit encoding of A_{ir}, T_{ir} (2 bits for each rigid body). This allows us to compress the CDF into 64 bits per grid node.

5.3 Reconstruct particle-wise colored distance field

Once we have the grid CDF (d_i, A_{ir} and T_{ir}), they can be reconstructed at other locations near the rigid surfaces. In our case we look at MPM particle locations $\mathbf{x}_p$. In Fig. 12 we show the reconstruction result for three intersecting rigid boundaries in 2D.

5.3.1 Particle color field. The color information can be reconstructed relatively easily. Specifically, a particle's affinities to rigid surfaces A_{pr} are directly inherited from grid affinity A_{ir}, where grid node i is any node within particle p's MPM support kernel. For particle tag T_{pr}, we take a distance weighted average

$$T_{pr} = \text{sign}\left(\sum_i N_i(\mathbf{x}_p) d_i T_{ir}\right), \tag{21}$$

where incorporating d_i in the weight reduces the influence of grid nodes that are near the rigid body whose tags can be ambiguous due to floating point error.

5.3.2 Particles distance and normal. As a particle approaches the surface boundary, it is not guaranteed to have a complete set of grid CDFs in its entire kernel support. Our fast distance splatting algorithm (§5.2.1) also tends to miss a small number of nodes near mesh corners. Therefore we cannot directly interpolate d_i to the particles. Instead, we use the robust MLS reconstruction technique

described in §3.1. According to the tag information on the particle and its associated grid nodes, we locally convert the grid unsigned distances to signed distances. Then we adopt Eq. 4 to reconstruct particle distance d_p and its gradient ∇d_p, where particle normal n_p is given by $\mathbf{n}_p = \nabla d_p / |\nabla d_p|$.

5.3.3 Particle color persistence and penalty force. Particle p's color A_{pr} and T_{pr} associated with rigid surface r will persist after being obtained, until all nodes in p's kernel lose affinities with r. This is important since a particle may slightly penetrate a surface due to numerical advection error, in which case we should not flip the tag. When this happens (see Fig. 13), we will then get a negative d_p with a correct normal $\mathbf{n}_p$ along which we could fix the penetration. Specifically, we detect negative d_p occurrences and keep track of a weak penalty force on these particles as

$$\mathbf{f}_p^{P,n} = -k_h d_p \mathbf{n}_p, \tag{22}$$

where k_h is the penalty stiffness parameter.

5.3.4 Particle grid compatibility. The reconstructed color information immediately allows us to partition all grid nodes within a particle's kernel. We use S_i to denote the set of surfaces that has non-zero A_{ir} and S_p for that of particle p. A grid node i and a particle p are *compatible* if and only if for all surfaces shared by the particle and the grid node, all tags are the same ($T_{ir} = T_{pr}, \forall r \in S_i \cap S_p$).

5.4 CPIC particle-to-grid transfer

We use i^{p+} to denote nodes that are compatible with particle p, and i^{p-} for the incompatible nodes. Similarly, p^{i+} are the particles that are compatible with grid node i, and p^{i-} are the incompatible particles.

We will assume the usage of APIC, although the extension to PolyPIC is straightforward. Near rigid surfaces, particles only transfer to compatible grid nodes:

$$m_i^n = \sum_{q \in \{p^{i+}\}} N_i(\mathbf{x}_q^n) m_q, \tag{23}$$

$$(m\mathbf{v})_i^n = \sum_{q \in \{p^{i+}\}} N_i(\mathbf{x}_q^n) m_q \left(\mathbf{v}_q^n + \mathbf{C}_q^n(\mathbf{x}_i - \mathbf{x}_q^n)\right). \tag{24}$$

Velocity projection operator. For each rigid body, we can compute its world-space velocity at position $\mathbf{x}$ as $\mathbf{V}_r^n(\mathbf{x}) = \mathbf{v}_r^n + \omega_r^n \times (\mathbf{x} - \mathbf{x}_r^n)$.

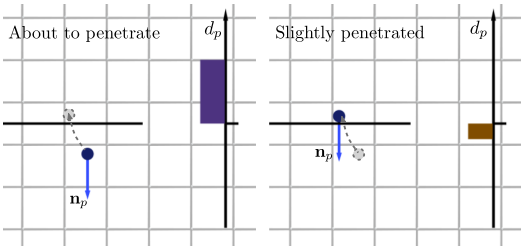

Fig. 13. Here we show the motion of a particle slightly penetrating a boundary due to numerical advection error. In this case, our method robustly maintains a persistent particle color (A_{pr}, T_{pr}) and normal $\mathbf{n}_p$. The reconstructed distance becomes negative when penetration happens. This allows us to apply a weak penalty force as explained in §5.3.3.

Fig. 14. A vertical Drucker-Prager plastic sand flow is two-way coupled with four rigid paddles, revealing intricate dynamics and flow patterns.

We also define a boundary projection operator for projecting an input velocity $\mathbf{v}$ given normal $\mathbf{n}$ and boundary condition B (sticky, slip, or separate):

$$\mathbf{Proj}(\mathbf{v}, \mathbf{n}, B, \mu) = \begin{cases} \vec{0}, & B \text{ is sticky}, \\ \mathbf{v}_t, & B \text{ is slip}, \\ \zeta \hat{\mathbf{v}}_t, & B \text{ is separate and } \mathbf{v} \cdot \mathbf{n} \le 0, \\ \mathbf{v}, & B \text{ is separate and } \mathbf{v} \cdot \mathbf{n} > 0, \end{cases} \tag{25}$$

where $\zeta = \max(0, |\mathbf{v}_t| + \mu \mathbf{v} \cdot \mathbf{n})$, $\mathbf{v}_t = \mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n}$, $\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{|\mathbf{v}_t|}$. Here $\mu \ge 0$ is the dynamic friction coefficient.

Velocity projection and impulses on rigid bodies. Given a particle p and rigid body r, the velocity contribution to the incompatible grid node i is projected to $\mathbf{Proj}_r(\mathbf{v}_p^n, \mathbf{n}_p, \mathbf{x}_i) = \mathbf{V}_r^n(\mathbf{x}_i) + \mathbf{Proj}(\mathbf{v}_p^n - \mathbf{V}_r^n(\mathbf{x}_i), \mathbf{n}_p, B_r, \mu_r)$, where B_r and μ_r are the boundary type and friction coefficient of rigid body r. In CPIC, each incompatible grid node $j \in i^{p-}$ results in an impulse $m_p(\mathbf{v}_p^n - \mathbf{Proj}_{r^*}(\mathbf{v}_p^n, \mathbf{n}_p, \mathbf{x}_j))N_j(\mathbf{x}_p^n)$ applied to the closest rigid body $r^*(\mathbf{x}_j)$ (tracked in §5.2.1) at $\mathbf{x}_j$.

MLS-MPM grid momentum update. The grid momentum is updated as (assuming symplectic Euler)

$$(m\hat{\mathbf{v}})_i^{n+1} = (m\mathbf{v})_i^n + \Delta t \left(m_i^n \mathbf{g} + \mathbf{f}_i^n\right), \tag{26}$$

where $\mathbf{g}$ is gravity and $\mathbf{f}_i^n$ is the MLS-MPM hyperelastic force given by Eq. 18. Note that we use notation $(m\hat{\mathbf{v}})_i^{n+1}$ instead of $(m\mathbf{v})_i^{n+1}$ since the later refers to the grid momentum in time t^{n+1} transferred from the particles. Implicit discretization of the hyperelastic force can be achieved similarly to [Stomakhin et al. 2013]. Then we update the velocities by dividing the momentum by mass: $\hat{\mathbf{v}}_i^{n+1} = (m\hat{\mathbf{v}})_i^{n+1} / m_i^n$.

5.5 CPIC grid-to-particle transfer

Normally for level set-based collision objects, boundary conditions are applied at grid nodes inside the level set. In our case for each particle, the velocities on incompatible grid nodes are however non-associated with the particle due to the enforcement of discontinuity. We take a ghost velocity approach, where we assume for any node $j \in i^{p-}$, its velocity is simply $\mathbf{v}_j = \mathbf{v}_p^n$ through a constant extrapolation from particle p. Thus the CPIC transfer from grid to particle

which gathers contributions from both incompatible and compatible nodes is given by

$$\mathbf{v}_p^{n+1} = \sum_{j \in i^{p-}} N_j(\mathbf{x}_p^n)\tilde{\mathbf{v}}_p + \sum_{j \in i^{p+}} N_j(\mathbf{x}_p^n)\hat{\mathbf{v}}_j^{n+1}, \tag{27}$$

$$\mathbf{C}_p^{n+1} = D_p^{-1}\left(\sum_{j \in i^{p-}} N_j(\mathbf{x}_p^n)\tilde{\mathbf{v}}_p \mathbf{z}_{jp}^{n\,T} + \sum_{j \in i^{p+}} N_j(\mathbf{x}_p^n)\hat{\mathbf{v}}_j^{n+1}\mathbf{z}_{jp}^{n\,T}\right), \tag{28}$$

where $\mathbf{z}_{jp}^n = \mathbf{x}_j - \mathbf{x}_p^n$. Here we include ghost velocities on incompatible nodes to prevent potential singularity of $\mathbf{D}_p$.

The collided particle velocity is given by $\tilde{\mathbf{v}}_p = \mathbf{Proj}_r(\mathbf{v}_p^n, \mathbf{n}_p^n, \mathbf{x}_j) + \Delta t c \mathbf{n}_p$, where r denotes the closest rigid boundary to the particle and c is non-zero when there is need to push the particle away from the boundary. Particle advection is then given by $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{v}_p^{n+1}$. With MLS-MPM, particle deformation gradient is updated as $\mathbf{F}_p^{n+1} = (\mathbf{I} + \Delta t \mathbf{C}_p^{n+1})\mathbf{F}_p^n$. Then we apply the penalty impulse $\Delta t \mathbf{f}_p^{p,n}$ (see Eq. 22) to the particle and the reverse impulse to the rigid body for conservation of momentum.

5.6 Rigid body advection

Dynamic rigid body velocities can be updated from the impulses computed in §5.4: $\mathbf{v}_r^{n+1} \leftarrow \mathbf{v}_r^*$ and $\omega_r^{n+1} \leftarrow \omega_r^*$, where $\mathbf{v}_r^*$ and ω_r^* are evolved from $\mathbf{v}_r^n$ and ω_r^n as described in §5.1. Then the rigid bodies are advected in a standard way.

6 A HIGH PERFORMANCE IMPLEMENTATION

Efficiency is a key concern in MPM since simulating a large number of particles can be time-consuming. The transfer operation from particle to grid (P2G) and that from grid to particle (G2P) are the bottlenecks for traditional MPM, which usually takes more than 85% of time based on our experience. In this section, we discuss our high-performance implementation of these two operations. Particularly, we decompose the performance gain into two parts: *performance engineering* and *algorithmic improvement*.

Performance engineering. We adopt low-level performance optimization techniques to accelerate the program with no algorithmic change. We use SPGrid [Setaluri et al. 2014] for background grid storage, and adopt techniques including *blocked transfer* and *eight-colored P2G for lock-free multi-threading*, as detailed in the supplementary document [Hu et al. 2018].

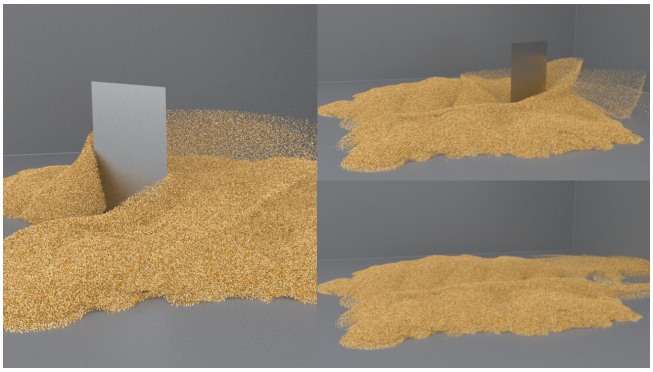

Fig. 15. We sweep a pile of sand with a kinematic thin shell object.

Timing (ms)	Reference	Ours (MPM)	Ours* (MPM)	Ours* (MLS-MPM)
P2G (1 thread)	4760 (1×)	5744 (0.83×)	2685 (1.77×)	1283 (3.71×)
P2G (4 threads)	1220 (1×)	1525 (0.80×)	688 (1.77×)	328 (3.72×)
G2P (1 thread)	8255 (1×)	7476 (1.10×)	1144 (7.21×)	589 (14.01×)
G2P (4 threads)	2070 (1×)	2011 (1.03×)	313 (6.61×)	163 (12.70×)

Table 2. Benchmarks of MPM transfer operations. Reliable reference implementation is from [Tampubolon et al. 2017]. Superscript * is with our performance optimization. All performance data are collected on an PC with an Intel Core i7-7700K CPU with four cores at 4.2GHz, and 4 × 8 GB DDR4 memory at 2400 MHz. Intel Turbo Boost is disabled for stable CPU frequency.

Algorithmic improvement. MLS-MPM halves the number of FLOPs needed for each particle. The unification of affine velocity field and deformation gradient eliminates the necessity for evaluating $\nabla N_i(\mathbf{x}_p^n)$, which speeds up both P2G and G2P. During P2G, MLS-MPM fuses the scattering of the affine momentum and particle force contribution into $N_i(\mathbf{x}_p^n)\mathbf{Q}_p(\mathbf{x}_i - \mathbf{x}_p^n)$, where

$$\mathbf{Q}_p = \Delta t V_p^0 M_p^{-1} \frac{\partial \Psi}{\partial \mathbf{F}}(\mathbf{F}_p^n)\mathbf{F}_p^{n\,T} + m_p \mathbf{C}_p,$$

so that only one matrix-vector multiplication is needed for the inner loop (27 iterations for 3D and quadratic B-spline); and during G2P, it avoids evaluating $\frac{\partial \mathbf{v}}{\partial \mathbf{x}}$ with $\nabla N_i(\mathbf{x})$.

6.1 Benchmark and discussion

We optimize the code for both traditional MPM and MLS-MPM. We also examine the generated assembly code to ensure that the compiler (gcc 5.4.1) correctly translates ideas mentioned above into machine instructions. Note that MLS-MPM is also easier to optimize thanks to its simplicity. To evaluate our implementation, we set up a benchmark with 8×10^6 uniformly distributed particles, and measure P2G and G2P time consumption for different implementations. More details on the benchmark is given in the supplementary document [Hu et al. 2018]. Results are summarized in Table 2, showing we achieve 3.71× and 14.01× higher performance for P2G and G2P respectively compared to a reliable implementation of the MPM solver in [Tampubolon et al. 2017] (with OpenVDB [Museth 2013]).

To further validate this significant improvement, we measure the floating point unit (FPU) utilization using Intel VTune on our optimized MLS-MPM and [Tampubolon et al. 2017]. Our P2G and G2P implementations lead to 1.93× and 7.38× FPU utilization compared with [Tampubolon et al. 2017]. Note that the performance improvement of our optimized traditional MPM over [Tampubolon et al. 2017] is proportional to the gain in FPU utilization.

Notably, the usage of MLS-MPM directly enhances MPM performance by 2× in the case of explicit time integration, regardless of whether low-level performance engineering is performed. Considering its ease of implementation, MLS-MPM can be easily imported to any existing MPM solvers with APIC/PolyPIC transfers.

Fig. 16. Our rigid-MPM coupling allows us to simulate terradynamics to predict legged robot's locomotion on granular media. We 3D print the model from [Li et al. 2013], equip it with motors, and demonstrate how our simulation results (top) match real world footage (bottom) on different motion patterns.

For implicit integration, efficient Krylov-solver-based implicit MPM usually adopts a matrix-free implementation to avoid reconstructing a sparse matrix in every time step. Each Krylov multiplication is essentially equivalent to a grid-to-particle-to-grid transfer cycle for velocity differentials. Transfers therefore remain the bottleneck. Similarly to the explicit force, our force differential also eliminates the necessity of evaluating kernel gradients and allows algorithmic performance gain. The gain is however less significant, because only in explicit time integration it benefits from unifying affine momentum and particle force contribution.

We developed our system based on *Taichi* [Hu 2018]. Our high-performance code will be open-sourced with the publication of this work. Please refer to our supplementary code and document [Hu et al. 2018] for more detailed discussion on implementation.

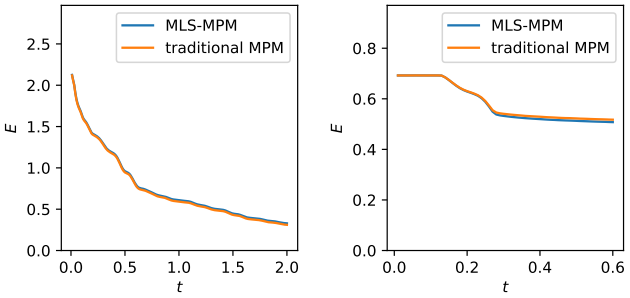

Fig. 17. Total energy evolution curves for MPM and MLS-MPM in the oscillating jello (left) and colliding balls (right) test cases. Numerical dissipation from the two methods are nearly identical.

7 RESULTS

Our experiments show that MLS-MPM produces visually comparable dynamics with traditional MPM. We also perform two standard hyperelasticity tests: *initially stretched oscillating cube* and *colliding balls*. The total energy evolution curves for MLS-MPM and MPM are plotted in Fig. 17 showing almost identical numerical dissipation.

We present various examples to demonstrate the efficacy of MLS-MPM with CPIC. Timing, statistics and material parameters are given in Table 3. We show world space cutting of elastic and elastoplastic objects including a progressively cut armadillo (Fig. 7), a dissected bunny (Fig. 4), a banana (Fig. 1) and a goat cheese block (Fig. 3). Similarly, our method handles thin boundary meshes, as demonstrated with sweeping (Fig. 15) and stirring (Fig. 9) granular materials. Two-way coupling with rigid bodies is naturally supported, and shown by dropping rigid blocks onto goo (Fig. 8), testing buoyancy in water (Fig. 6) and hitting paddles with sand (Fig. 14). A sand wheel (Fig. 5) and several water wheels (Fig. 1) simultaneously demonstrate the robust treatment of infinitely thin boundaries and two-way rigid body coupling.

Two-way coupled rigid-MPM simulation is also useful for robotics and terradynamics. In Fig. 16 we simulate and validate locomotion for robot navigating in granular media. We provide more details about the 3D printed robot in the supplementary document [Hu et al. 2018].

CPIC enables powerful new features for MPM at low cost since only a narrow band near the rigid boundaries needs CPIC. In the banana example (Fig. 1), each frame takes around 131.9s if the cutter is removed, while it takes 140.5s when cutting is enabled, with only 6% CPIC overhead. (For fair comparison in this experiment, CPIC and regular MPM transfers are optimized with equal efforts.)

8 LIMITATIONS AND FUTURE WORK

While CPIC tackles infinitely thin boundaries, it only resolves features at a scale of grid Δx. Thus we cannot handle sub-grid level

Example	sec/Frame	Δx	Δt	Particle #	Density	Young's Modulus	Bulk Modulus	Yield Stress	Friction Angle
(Fig. 7) Cutting armadillo	107.0	5.0×10^{-3}	5×10^{-6}	1.3M	400	1×10^5	-	-	-
(Fig. 4) Bunny split	16.7	3.3×10^{-3}	3×10^{-4}	2.0M	400	1.5×10^3	-	-	-
(Fig. 1) Water wheel	120.3	2.5×10^{-3}	5×10^{-5}	1.4M	1000	-	1×10^4	-	-
(Fig. 6) Buoyancy	122.9	3.1×10^{-3}	1×10^{-4}	5.3M	1000	-	1×10^4	-	-
(Fig. 3) Cheese	227.1	3.3×10^{-3}	2×10^{-5}	1.8M	400	4×10^5	-	10	-
(Fig. 8) Goo blocks	105.3	4.2×10^{-3}	1×10^{-4}	1.8M	400	5×10^4	-	10	-
(Fig. 1) Banana	140.5	4.2×10^{-3}	5×10^{-5}	1.2M	400	4×10^5	-	5	-
(Fig. 5) Sand wheel (2D)	0.3	2.5×10^{-3}	1×10^{-4}	4.1K	1000	3.5×10^5	-	-	45
(Fig. 15) Sand sweep	288.3	2.0×10^{-3}	2×10^{-5}	2.6M	400	3.5×10^5	-	-	35
(Fig. 9) Sand stir	158.3	3.1×10^{-3}	1×10^{-4}	5.2M	400	3.5×10^5	-	-	10
(Fig. 14) Sand paddles	20.0	2×10^{-3}	1×10^{-4}	1.0M	400	3.5×10^5	-	-	30
(Fig. 16) Robot	112.4	3.3×10^{-3}	1×10^{-4}	3.4M	2000	1.8×10^6	-	-	45
(Fig. 16) Robot (reverse)	116.8	3.3×10^{-3}	1×10^{-4}	3.4M	2000	1.8×10^6	-	-	45

Table 3. Particle count and time per frame are provided as average values. All are measured on an Intel Core i7-7700K CPU with four cores at 4.2GHz. We use the Poisson's ratio $\nu = 0.3$ for all the examples. Elastic materials are simulated with the fixed Corotated hyperelasticity [Stomakhin et al. 2013]. Weakly compressible water is done as in [Tampubolon et al. 2017]. Plastic materials adopt St. Venant-Kirchhoff elasticity with von Mises plasticity [Gao et al. 2017]. Granular materials use St. Venant-Kirchhoff elasticity with Drucker-Prager plasticity [Klár et al. 2016].

boundary configurations such as sharp corners and narrow gaps as done by Azevedo et al. [2016] with cut-cells. Our compatibility condition between particles and grids nodes is a binary decision and essentially grid-aligned. One possible future direction for increasing the accuracy would be enforcing a smoother transition region based on sub-grid features. Also, it is hard to reconstruct sharp cutting surfaces from particles. Incorporating embedded meshes as in [Wojtan et al. 2009] would be worth investigating. Fully implicit rigid-MPM strong coupling is not formulated in this work and we leave that as future work.

MLS-MPM provides a new perspective for discretizing the governing equations that is consistent with other meshless approaches. We believe it also builds a foundation for devising higher order MPM schemes for enhanced accuracy and visual vividness. It is also a promising direction to look into reducing the cell-crossing error when multilinear kernels are used. While traditional MPM readily goes unstable in this case, MLS-MPM provides possible solutions due to its freedom in choosing the function space and weighting functions. Efficient spatial adaptivity with [Gao et al. 2017] would be another interesting topic for further study, since MLS-MPM does not need a regular grid or intricate discretization strategies on hanging nodes. Due to its robust nature, moving least squares would work on any unstructured (or even non-manifold) grid. For example it would be interesting to investigate applying MLS-MPM to power diagrams [de Goes et al. 2015] for solids and granular media. Furthermore, coupling MPM particles and mesh-based solvers (e.g. cloth) with CPIC would be a potential future direction, since the boundary particles can represent arbitrary codimension-1 manifolds.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for their valuable suggestions and comments. We thank Christopher Long from Los Alamos National Laboratory for useful discussions, Hannah Bollar from University of Pennsylvania for narrating the video, and Hangxin Liu from UCLA CS Department for assisting the robot experiments. The work is partially supported by Jiang's StartUp Grant from the University of Pennsylvania, NSF IIS-1755544, National Key Technology R&D Program of China (2017YFB1002701), a gift from Awowd Inc., a gift from NVIDIA Corporation, and a gift from SideFX.

REFERENCES

N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler, and M. Teschner. 2012. Versatile rigid-fluid coupling for incompressible SPH. *ACM Trans Graph* 31, 4, Article 62 (July 2012), 8 pages.

V. Azevedo, C. Batty, and M. Oliveira. 2016. Preserving geometry and topology for fluid flows with thin obstacles and narrow gaps. *ACM Trans Graph* 35, 4 (2016), 97.

S. Band, C. Gissler, and M. Teschner. 2017. Moving least squares boundaries for SPH fluids. *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (2017).

B. Banerjee, J. Guilkey, T. Harman, J. Schmidt, and P. McMurtry. 2012. Simulation of impact and fragmentation with the material point method. *arXiv preprint arXiv:1201.2452* (2012).

Z. Bao, J. Hong, J. Teran, and R. Fedkiw. 2007. Fracturing rigid materials. *IEEE Transactions on Visualization and Computer Graphics* 13, 2 (2007).

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

M. Becker, H. Tessendorf, and M. Teschner. 2009. Direct forcing for Lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics* 15, 3 (May 2009), 493–503.

T. Belytschko, Y. Lu, and L. Gu. 1994. Element-free Galerkin methods. *International journal for numerical methods in engineering* 37, 2 (1994), 229–256.

T. Belytschko and M. Tabbara. 1996. Dynamic fracture using element-free Galerkin methods. *Internat. J. Numer. Methods Engrg.* 39, 6 (1996), 923–938.

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

M. Carlson, P. Mucha, and G. Turk. 2004. Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans Graph* 23, 3 (2004), 377–384.

Z. Chen, M. Yao, R. Feng, and H. Wang. 2014. Physics-inspired adaptive fracture refinement. *ACM Trans Graph* 33, 4 (2014), 113.

N. Chentanez, T. Goktekin, B. Feldman, and J. O'Brien. 2006. Simultaneous coupling of fluids and deformable bodies. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim.* Eurographics Association, 83–89.

G. Daviet and F. Bertails-Descoubes. 2016. A semi-implicit material point method for the continuum simulation of granular materials. *ACM Trans Graph* 35, 4 (2016), 102:1–102:13.

F. de Goes, C. Wallez, J. Huang, D. Pavlov, and M. Desbrun. 2015. Power particles: an incompressible fluid solver based on power diagrams. *ACM Trans Graph* 34, 4 (2015), 50–1.

B. Feldman, J. O'Brien, and B. Klingner. 2005. Animating gases with hybrid meshes. In *ACM Trans Graph*, Vol. 24. ACM, 904–909.

S. Fleishman, D. Cohen-Or, and C. Silva. 2005. Robust moving least-squares fitting with sharp features. In *ACM Trans Graph*, Vol. 24. ACM, 544–552.

C. Fu, Q. Guo, T. Gast, C. Jiang, and J. Teran. 2017. A polynomial Particle-In-Cell method. *ACM Trans Graph* 36, 6, Article 222 (2017), 12 pages.

M. Gao, A. Pradhana Tampubolon, C. Jiang, and E. Sifakis. 2017. An adaptive Generalized Interpolation Material Point Method for simulating elastoplastic materials. *ACM Trans Graph* 36, 6 (2017).

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

E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw. 2005. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans Graph* 24, 3 (July 2005), 973–981.

D. Hahn and C. Wojtan. 2015. High-resolution brittle fracture simulation with boundary elements. *ACM Trans Graph* 34, 4 (2015), 151.

D. Hahn and C. Wojtan. 2016. Fast approximations for boundary element based brittle fracture simulation. *ACM Trans Graph* 35, 4 (2016), 104.

J. Hegemann, C. Jiang, C. Schroeder, and J. Teran. 2013. A level set method for ductile fracture. In *Proc ACM SIGGRAPH/Eurograp Symp Comp Anim.* 193–201.

Y. Hu. 2018. Taichi: An Open-Source Computer Graphics Library. *arXiv preprint arXiv:1804.09293* (2018).

Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, and C. Jiang. 2018. A moving least squares Material Point Method with displacement discontinuity and two-way rigid body coupling: supplementary document. (2018).

A. Huerta, T. Belytschko, S. Fernández-Méndez, and T. Rabczuk. 2004. Meshfree methods. (2004).

Thomas J.R. Hughes. 2012. *The finite element method: Linear static and dynamic finite element analysis.* Courier Corporation.

G. Irving, J. Teran, and R. Fedkiw. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim.* 131–140.

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

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

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

C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle. 2016. The material point method for simulating continuum materials. In *SIGGRAPH 2016 Course.* 24:1–24:52.

Y. Kanamori, N. Cuong, and T. Nishita. 2011. Local optimization of distortions in wide-angle images using moving least-squares. In *Proceedings of the 27th Spring Conference on Computer Graphics.* ACM, 51–56.

P. Kaufmann, S. Martin, M. Botsch, and M. Gross. 2009. Flexible simulation of deformable models using discontinuous Galerkin FEM. *Graphical Models* 71, 4 (2009), 153–167.

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

B. Klingner, B. Feldman, N. Chentanez, and J. O'Brien. 2006. Fluid animation with dynamic meshes. In *ACM Trans Graph*, Vol. 25. ACM, 820–825.

D. Koschier and J. Bender. 2017. Density maps for improved SPH boundary handling. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim.* ACM, 1.

D. Koschier, J. Bender, and N. Thuerey. 2017. Robust eXtended finite elements for complex cutting of deformables. *ACM Trans Graph* 36, 4 (2017), 55.

P. Lancaster and K. Salkauskas. 1981. Surfaces generated by moving least squares methods. *Mathematics of computation* 37, 155 (1981), 141–158.

T. Langlois, S. An, K. Jin, and D. James. 2014. Eigenmode compression for modal sound models. *ACM Trans Graph* 33, 4 (2014), 40.

D. Levin. 1998. The approximation power of moving least-squares. *Mathematics of Computation of the American Mathematical Society* 67, 224 (1998), 1517–1531.

D. Levin. 2004. Mesh-independent surface interpolation. In *Geometric modeling for scientific visualization.* Springer, 37–49.

C. Li, T. Zhang, and D. Goldman. 2013. A terradynamics of legged locomotion on granular media. *Science* 339, 6126 (2013), 1408–1412.

W. Liu, S. Jun, and Y. Zhang. 1995. Reproducing kernel particle methods. *International journal for numerical methods in fluids* 20, 8-9 (1995), 1081–1106.

F. Losasso, T. Shinar, A. Selle, and R. Fedkiw. 2006. Multiple interacting liquids. In *ACM Trans Graph*, Vol. 25. ACM, 812–819.

M. Macklin and M. Müller. 2013. Position based fluids. *ACM Trans Graph* 32, 4 (2013), 104:1–104:12.

M. Macklin, M. Müller, N. Chentanez, and T. Kim. 2014. Unified particle physics for real-time applications. *ACM Trans Graph* 33, 4 (2014), 153:1–153:12.

S. Martin, P. Kaufmann, M. Botsch, E. Grinspun, and M. Gross. 2010. Unified simulation of elastic rods, shells, and solids. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 39.

N. Mitchell, M. Aanjaneya, R. Setaluri, and E. Sifakis. 2015. Non-manifold level sets: A multivalued implicit surface representation with applications to self-collision processing. *ACM Trans Graph* 34, 6 (2015), 247.

N. Molino, Z. Bao, and R. Fedkiw. 2005. A virtual node algorithm for changing mesh topology during simulation. In *ACM SIGGRAPH 2005 Courses.* ACM, 4.

G. Moutsanidis, D. Kamensky, D.Z. Zhang, Y. Bazilevs, and C.C. Long. 2018. Modeling sub-grid scale discontinuities in the Material Point Method using a single velocity field. *Submitted, received via private communication* (2018).

M. Müller, D. Charypar, and M. Gross. 2003. Particle-based fluid simulation for interactive applications. In *Symp Comp Anim (SCA '03).* 154–159.

M. Müller and M. Gross. 2004. Interactive virtual materials. In *Proceedings of Graphics Interface 2004 (GI '04).* Canadian Human-Computer Commu, 239–246.

M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. 2007. Position based dynamics. *J Vis Comm Imag Repre* 18, 2 (2007), 109–118.

M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. 2004. Point based animation of elastic, plastic and melting objects. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim.* Eurographics Association, 141–151.

K. Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM Trans Graph* 32, 3 (2013), 27.

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

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

J. O'Brien, A. Bargteil, and J. Hodgins. 2002. Graphical modeling and animation of ductile fracture. In *Proc ACM SIGGRAPH 2002.* 291–294.

J. O'Brien and J. Hodgins. 1999. Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques.* ACM Press/Addison-Wesley Publishing Co., 137–146.

M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L. J Guibas. 2005. Meshless animation of fracturing solids. *ACM Trans Graph* 24, 3 (2005), 957–964.

T. Pfaff, R. Narain, J. de Joya, and J. O'Brien. 2014. Adaptive tearing and cracking of thin sheets. *ACM Trans Garph* 33, 4 (2014), 110.

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

A. Robinson-Mosher, R. English, and R. Fedkiw. 2009. Accurate tangential velocities for solid fluid coupling. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim.* ACM, 227–236.

A. Robinson-Mosher, T. Shinar, J. Gretarsson, J. Su, and R. Fedkiw. 2008. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans Graph* 27, 3 (2008), 46:1–46:9.

S. Sato, Y. Dobashi, K. Iwasaki, T. Yamamoto, and T. Nishita. 2014. Deformation of 2D flow fields using stream functions. In *SIGGRAPH Asia 2014 Technical Briefs.* ACM, 4.

S. Schaefer, T. McPhail, and J. Warren. 2006. Image deformation using moving least squares. In *ACM Trans Graph*, Vol. 25. ACM, 533–540.

R. Setaluri, M. Aanjaneya, S. Bauer, and E. Sifakis. 2014. SPGrid: A sparse paged grid structure applied to adaptive smoke simulation. *ACM Trans Graph* 33, 6 (2014), 205.

T. Shinar, C. Schroeder, and R. Fedkiw. 2008. Two-way coupling of rigid and deformable bodies. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* Eurographics Association, 95–103.

E. Sifakis, K. Der, and R. Fedkiw. 2007. Arbitrary cutting of deformable tetrahedralized objects. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim.* Eurographics Association, 73–80.

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

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

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

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

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

D. Terzopoulos and K. Fleischer. 1988. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. *SIGGRAPH Comp Graph* 22, 4 (1988), 269–278.

Y. Wang, C. Jiang, C. Schroeder, and J. Teran. 2014. An adaptive virtual node algorithm with robust mesh cutting. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim.* Eurographics Association, 77–85.

C. Wojtan, N. Thürey, M. Gross, and G. Turk. 2009. Deforming meshes that split and merge. In *ACM Trans Graph*, Vol. 28. ACM, 76.

J. Wretborn, R. Armiento, and K. Museth. 2017. Animation of crack propagation by means of an extended multi-body solver for the material point method. *Computers & Graphics* (2017).

J. Wu, R. Westermann, and C. Dick. 2015. A survey of physically based simulation of cuts in deformable bodies. In *Comp Graph Forum*, Vol. 34. Wiley Online Library, 161–187.

H. Xu and J. Barbič. 2014. Signed distance fields for polygon soup meshes. In *Proceedings of Graphics Interface 2014.* Canadian Information Processing Society, 35–41.

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

O. Zarifi and C. Batty. 2017. A positive-definite cut-cell method for strong two-way coupling between fluids and deformable bodies. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim.* ACM, 7.

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

Y. Zhu and S. Gortler. 2007. 3D deformation using moving least squares. (2007).